Simulated Annealing and Niching Genetic Algorithms for the Bottleneck Traveling Salesman Problem

Link to Code Link to Video

Daniella Arteaga Mendoza
Speed School of Engineering
University of Louisville
D0Arte01@louisville.edu

Matthew Poncini
Speed School of Engineering
University of Louisville
M0Ponc01@Louisville.edu

I. Introduction

The bottleneck Traveling Salesman Problem (BTSP) asks for a Hamiltonian tour whose largest edge cost is minimized. This objective is relevant in applications where the worst-case of a route dominates performance, such as latency-sensitive communication, evacuation planning or logistics under capacity constraints. Unlike the classical TSP, where the sum of edge lengths is minimized, the BTSP shifts the landscape toward plateaus and many solutions with similar bottleneck values, which makes it an interesting testbed for meta-heuristics.

Early work on BTSP focused on exact methods. Garfinkel and Gilbert analyzed the probabilistic structure of the bottleneck objective and established NP-hardness [1]. Carpaneto *et al.* developed assignment-based branch-and-bound algorithms and high-quality codes to solve small and medium-sized instances exactly [2]. Hochbaum and Shmoys later proposed a unified approximation framework for bottleneck optimization problems with polynomial-time guarantees [3]. More recently, Helsgaun showed that the Lin–Kernighan-Helsgaun (LKH) heuristic can be adapted to BTSP, obtaining high-quality solutions on large instances in reasonable time [4].

TABLE I
Survey of Literature on Bottleneck TSP

Reference	Algorithm	Category	Complexity
Probabilistic Analysis of a Bottleneck TSP, 1978 [1]	Threshold Graph, Hamiltonicity	Exact	NP-hard
Algorithms for the Bottleneck TSP, 1984 [2]	Assignment-Based B&B	Exact	NP-hard
A Unified Approach to Bottleneck Optimization, 1986 [3]	Bottleneck Approx. Framework	Approx.	Polynomial
Solving the Bottleneck TSP with LKH, 2014 [4]	LKH, BLKH Heuristic	Heuristic	Polynomial

In this project, we compare three families of approaches on Euclidean BTSP instances: (i) several cooling schedules for Simulated Annealing (SA), (ii) three variants of a niching Genetic Algorithm (GA) designed to preserve multiple local optima, and (iii) two configurations of the LKH heuristic

as a strong baseline. Our goal is to understand the trade-offs among solution quality, runtime, and diversity of solutions (niches). A brief survey of relevant BTSP work is summarized in Table I.

II. Optimization Problem

We consider the Euclidean BTSP on a complete graph $G = (V, E)$, where each vertex $i \in V$ corresponds to a city with coordinates in the plane and the cost c_{ij} of edge $(i, j) \in E$ is the Euclidean distance between cities i and j.

Let $x_{ij} \in \{0, 1\}$ indicate whether edge (i, j) is included in the tour, and let B denote the bottleneck value. A standard formulation is:

$$\min \; B \tag{1}$$

$$\text{s.t.} \sum_{j \in V, j \neq i} x_{ij} = 2, \quad \forall i \in V, \tag{2}$$

$$\sum_{i,j \in S, i \neq j} x_{ij} \leq |S| - 1, \quad \forall S \subset V, S \neq \emptyset, \tag{3}$$

$$c_{ij} x_{ij} \leq B, \quad \forall (i, j) \in E, \tag{4}$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in E. \tag{5}$$

Constraint (2) enforces that each city has degree two, constraint (3) eliminates subtours, and constraint (4) ties the decision variables to the bottleneck objective. The input size is dominated by $|V| = n$ and the $O(n^2)$ edge costs.

III. Search Space

We represent a tour as a permutation of the n cities. Thus the search space is the set of all permutations:

$$\Omega = \{(v_1, \ldots, v_n) : v_i \in V, v_i \neq v_j \text{ for } i \neq j\}, \tag{6}$$

with $|\Omega| = (n-1)!/2$ distinct Hamiltonian cycles in the undirected case. The dimensionality grows linearly with n, but the number of feasible solutions grows super-exponentially. This leads to a highly multi-modal search landscape where tours with similar bottleneck values may be very distant in permutation space.

IV. Difficulties

Solving BTSP exactly is NP-hard [1], so heuristic or meta-heuristic methods are required for large n.

- **Combinatorial explosion:** The number of tours grows as $(n-1)!/2$, so exhaustive search is infeasible beyond very small instances.
- **Plateaus and local minima:** Because the objective depends only on the maximum edge cost, many tours share the same bottleneck value. Small perturbations can leave the objective unchanged, producing large plateaus that are challenging for local search.
- **Deceptive structure:** Shortening one long edge may expose another as the new bottleneck, which can mislead greedy improvements.
- **Diversity vs. intensification:** Meta-heuristics must balance exploiting promising regions and maintaining diversity to discover multiple high-quality tours.

V. Related Problems

BTSP is closely related to the classical TSP, which minimizes the sum of edge costs, as well as to other bottleneck problems such as the Bottleneck Assignment Problem and Bottleneck Minimum Spanning Tree. Unlike MST or shortest path problems, which are solvable in polynomial time, both TSP and BTSP are NP-hard in general graphs. BTSP can be transformed into TSP by introducing a lexicographic objective, and conversely TSP variants can be approximated via bottleneck relaxations.

VI. Optimization Methods

We compare the following optimization methods:

1) **Simulated Annealing (SA):** A single-solution meta-heuristic that performs local moves (2-opt edge swaps) and accepts worse moves with a probability controlled by a temperature parameter. We implement three cooling schedules: (i) *slow cooling* (exploratory), (ii) *fast cooling*, and (iii) an *aggressive exploration* schedule with higher initial temperature and more frequent reheating.
2) **Niching Genetic Algorithm (GA):** A population-based meta-heuristic that evolves

a set of tours using selection, crossover, and mutation. We combine standard GA operators with a niching mechanism based on distance in permutation space to encourage multiple local optima. We evaluate three variants: (i) a *standard* configuration, (ii) *high-exploration* with stronger mutation and weaker selection, and (iii) *low-diversity* with stronger selection and weaker niching.

3) **LKH baseline:** We adapt the LKH heuristic for BTSP following [4] with two parameter settings ($k = 15$ and $k = 50$) to represent a strong high-performance baseline.

VII. Algorithms' Pseudocode

A. Simulated Annealing for BTSP

Algorithm 1: Simulated Annealing for BTSP

Generate initial tour π (e.g., random permutation)
$B \leftarrow \text{bottleneck}(\pi)$, $\pi^\star \leftarrow \pi$, $B^\star \leftarrow B$
$T \leftarrow T_0$
for $t = 1$ to MaxIter **do**
  $\pi' \leftarrow \text{2OptNeighbor}(\pi)$
  $B' \leftarrow \text{bottleneck}(\pi')$
  if $B' \leq B$ **then**
    $\pi \leftarrow \pi'$, $B \leftarrow B'$
    if $B' < B^\star$ **then**
      $\pi^\star \leftarrow \pi'$, $B^\star \leftarrow B'$
    end if
  else if $\text{rand}() < \exp(-(B' - B)/T)$ **then**
    $\pi \leftarrow \pi'$, $B \leftarrow B'$
  end if
  $T \leftarrow \text{UpdateTemperature}(T)$
end for
return $\pi^\star, B^\star$

B. Niching Genetic Algorithm for BTSP

Algorithm 2: Niching Genetic Algorithm for BTSP

Initialize population P with P random tours
Evaluate bottleneck of each individual in P
$\pi^\star \leftarrow$ best individual in P, $B^\star \leftarrow \text{bottleneck}(\pi^\star)$
for $g = 1$ to G **do**
  Compute niches using distance between tours
  Assign niche-penalized fitness values
  Select parents from P according to fitness
  Apply crossover with probability p_c to create offspring
  Apply mutation (swap/2-opt) with probability p_m
  Evaluate offspring and merge with current population
  Apply niching/survivor selection to form new P
  Update $\pi^\star$, $B^\star$ if a better individual is found
end for
return $\pi^\star, B^\star$

VIII. Algorithms Relation to Existing Methods

Both SA and the niching GA are *approximate* meta-heuristics relying on partial evaluation of the search space.

- **Simulated Annealing:** Single-solution, stochastic local search. It performs partial evaluation and is an approximate method.
- **Niching GA:** Population-based evolutionary algorithm with explicit diversity preservation. It performs partial evaluation via sampling and recombination and is also approximate.
- **LKH:** A deterministic (but multi-start) local-improvement heuristic that applies sophisticated search moves. It is also an approximate method without guaranteed optimality in general.

IX. Complexity

Let n be the number of cities. A single 2-opt move can be evaluated in $O(1)$ incremental time if implemented carefully, though naive implementations are $O(n)$.

- **SA:** Let I be the number of iterations. Each iteration performs a constant number of local moves and bottleneck updates, so the complexity is $O(I)$ in terms of objective evaluations. If we consider naive 2-opt evaluation, the cost becomes $O(In)$.
- **Niching GA:** Let P be the population size and G the number of generations. Selection, crossover, and mutation are $O(P)$ per generation, while computing niche distances is $O(P^2)$ in the worst case. Thus the overall complexity is $O(GP^2)$ for naive niching.
- **LKH:** The complexity per run is instance-dependent and dominated by repeated local search moves. Empirically, it scales super-linearly in n but remains highly efficient in practice.

X. Experimental Setup

We evaluate all algorithms on randomly generated Euclidean BTSP instances with $n \in \{5, 15, 25, 50, 100\}$ cities. Coordinates are sampled uniformly in a bounded 2D region, and edge costs are Euclidean distances. For each combination of algorithm and problem size, we perform 10 independent runs with different random seeds.

The following configurations are compared:

- **SA variants:**
  1) Slow cooling (exploratory)
  2) Fast cooling
  3) Aggressive exploration (higher initial temperature / reheating)
- **Niching GA variants:**
  1) Standard configuration
  2) High-exploration (higher mutation, weaker selection)
  3) Low-diversity (stronger selection, weaker niching)
- **LKH:** Two parameter settings with $k = 15$ and $k = 50$.

We record three performance metrics:

1) The bottleneck edge length of the best tour found.
2) The runtime in seconds.
3) For GA variants, the number of distinct niches maintained at the end of the run.

XI. Results and Discussion

A. Solution Quality vs. Problem Size

Figure 1 reports the average bottleneck edge length as a function of the number of cities. Each point corresponds to the mean over 10 runs, and error bars indicate one standard deviation.

For small instances ($n = 5$ and $n = 15$), all methods quickly reach similar bottleneck values, with minor differences among SA, GA, and LKH. As the number of cities increases, LKH and the best-performing SA schedules tend to achieve slightly smaller bottlenecks than the GA variants, reflecting the strong local improvement mechanisms of LKH and the ability of SA to intensify search around good tours. The high-exploration GA sacrifices some final bottleneck quality for increased diversity.

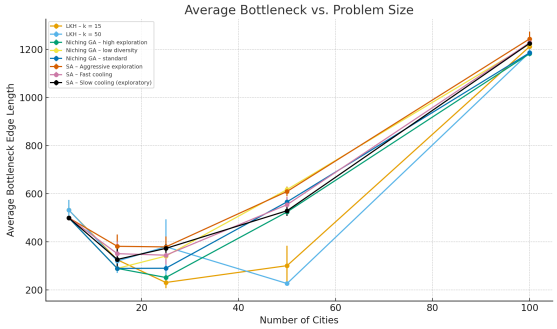

Fig. 1. Average bottleneck edge length as a function of the number of cities for all simulated annealing (SA), niching GA, and LKH configurations. Each point is averaged over 10 runs; error bars indicate one standard deviation.

B. Runtime vs. Problem Size

Figure 2 shows the average runtime on a logarithmic scale.

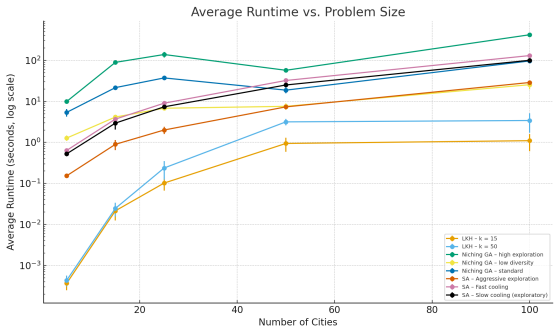

Fig. 2. Average runtime (log scale) versus number of cities. Niching GA variants are substantially more expensive than simulated annealing and LKH, especially on larger instances. LKH remains the fastest heuristic across all problem sizes.

LKH is consistently the fastest algorithm across all problem sizes, often finishing in under a second even for $n = 100$ in our implementation. SA requires more time than LKH but remains manageable, scaling roughly polynomially with n given the fixed number of iterations. In contrast, the niching GA variants are one to two orders of magnitude slower on larger instances, mainly due to $O(P^2)$ niche computations and population-level

operations. This highlights a clear trade-off: GA provides richer information about multiple optima but at significantly higher computational cost.

C. Niching Behavior of GA Variants

Figure 3 focuses on the niching GA variants and plots the average number of niches as a function of problem size.

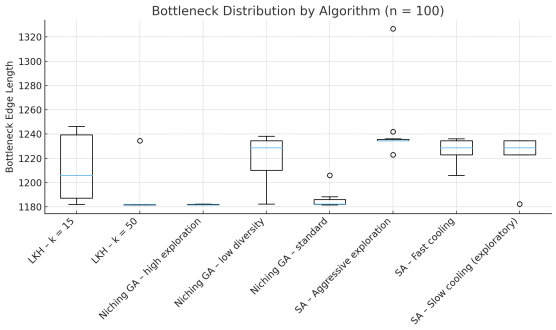

Fig. 3. Average number of niches discovered by the niching GA variants as a function of the number of cities. The high-exploration configuration generally finds more distinct niches, while the low-diversity variant collapses to fewer niches for medium-sized instances.

The high-exploration variant maintains the largest number of niches on most instance sizes, confirming that stronger mutation and weaker selection promote diversity. The standard configuration balances diversity and convergence, while the low-diversity variant tends to collapse to fewer niches (particularly for $n = 25$ and $n = 50$), behaving more like a conventional GA with a single dominant cluster. For $n = 100$, the standard GA recovers a larger number of niches, indicating that the increased combinatorial complexity naturally supports multiple local optima.

D. Robustness on the Largest Instance

To analyze robustness, Figure 4 presents boxplots of the bottleneck values obtained on the largest instance size ($n = 100$).

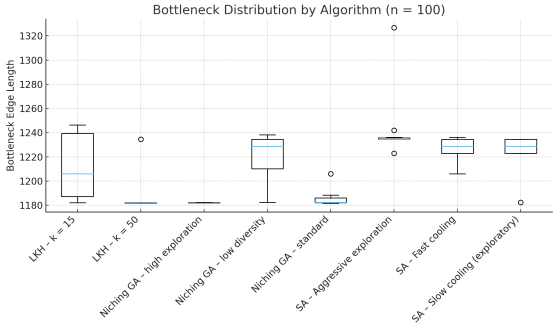

Fig. 4. Distribution of bottleneck edge lengths for all algorithms on the 100-city instance. Boxplots summarize 10 independent runs per algorithm, highlighting both central tendency and variability in solution quality.

The LKH configurations display both strong median performance and relatively low variance, which is expected from a powerful deterministic local-search heuristic. SA variants exhibit slightly higher variance, especially under aggressive exploration, but remain competitive in terms of median bottleneck values. The niching GA variants show wider spreads: while they sometimes match or outperform SA on individual runs, they also produce worse tours in others, reflecting the stochasticity and the emphasis on diversity rather than pure exploitation. This supports the view of GA as a tool for exploring multiple good tours rather than optimizing a single one.

REFERENCES

[1] R. S. Garfinkel and K. C. Gilbert, "Probabilistic Analysis of a Bottleneck Traveling Salesman Problem," *Journal of the ACM*, vol. 25, no. 1, pp. 1–14, 1978.

[2] G. Carpaneto, S. Martello, and P. Toth, "Algorithms and Codes for the Bottleneck Traveling Salesman Problem," *Operations Research*, vol. 32, no. 3, pp. 528–542, 1984.

[3] D. S. Hochbaum and D. B. Shmoys, "A Unified Approach to Bottleneck Optimization Problems," *Journal of the ACM*, vol. 33, no. 3, pp. 533–550, 1986.

[4] K. Helsgaun, "Solving the Bottleneck Traveling Salesman Problem Using the Lin–Kernighan–Helsgaun Algorithm (LKH)," Technical Report, Roskilde University, 2014. (Associated BLKH code available from author's LKH distribution.)

APPENDIX A: SIMULATED ANNEALING RESULTS

TABLE II: Simulated Annealing Results Across All City Sizes

Cities	**SA Slow**	**Runtime** (s)	**Iter**	**SA Fast**	**Runtime** (s)	**Iter**	**SA Agg**	**Runtime** (s)	**Iter**
5	499.66	0.4300	2575	499.66	0.69	3372	499.66	0.16	774
5	499.66	0.5500	2575	499.66	0.66	3372	499.66	0.15	774
5	499.66	0.5300	2575	499.66	0.66	3372	499.66	0.15	774
5	499.66	0.5300	2575	499.66	0.61	3372	499.66	0.13	774
5	499.66	0.5200	2575	499.66	0.59	3372	499.66	0.15	774
5	499.66	0.6400	2575	499.66	0.61	3372	499.66	0.16	774
5	499.66	0.5600	2575	499.66	0.53	3372	499.66	0.16	774
5	499.66	0.5400	2575	499.66	0.70	3372	499.66	0.17	774
5	499.66	0.5400	2575	499.66	0.62	3372	499.66	0.15	774
5	499.66	0.4100	2575	499.66	0.69	3372	499.66	0.15	774
15	295.8	3.0500	2575	295.8	3.12	3372	379.28	0.72	774
15	379.28	2.3900	2575	381.31	4.77	3372	428.04	0.72	774
15	289.63	2.7200	2575	381.31	3.24	3372	428.04	0.72	774
15	295.8	2.4400	2575	379.28	3.19	3372	379.28	0.72	774
15	289.63	4.2000	2575	295.8	3.18	3372	382.8	0.71	774
15	379.28	2.5200	2575	289.63	4.89	3372	295.8	0.71	774
15	381.31	2.4300	2575	295.8	3.17	3372	379.28	1.16	774
15	289.63	2.4100	2575	379.28	3.16	3372	428.04	1.28	774
15	289.63	2.4000	2575	382.8	3.23	3372	297.9	1.28	774
15	379.28	4.9400	2575	428.04	4.93	3372	416.48	0.94	774
25	380.81	9.3700	2575	290.72	9.25	3372	416.48	1.83	774
25	398.69	6.2000	2575	382.8	8.42	3372	342.97	1.82	774
25	379.28	9.0700	2575	375.11	9.87	3372	398.69	1.81	774
25	380.81	6.1900	2575	342.97	8.83	3372	342.97	1.82	774
25	340.18	8.0600	2575	294.85	8.90	3372	428.04	1.83	774
25	340.18	6.1800	2575	389.19	9.71	3372	374.54	3.00	774
25	379.28	7.2800	2575	374.54	8.26	3372	379.28	2.44	774
25	374.54	6.9500	2575	340.18	9.35	3372	294.85	1.82	774
25	379.28	8.0500	2575	360.35	9.70	3372	374.54	1.82	774
25	374.54	6.2200	2575	294.85	7.85	3372	437.27	1.82	774
50	518.46	27.46	2575	529.77	31.51	3372	624.90	8.24	774
50	537.93	25.30	2575	529.77	31.58	3372	624.92	6.43	774
50	503.29	25.22	2575	581.75	34.30	3372	619.72	8.24	774
50	503.29	25.06	2575	531.47	31.22	3372	617.02	6.41	774
50	535.38	23.63	2575	557.77	32.10	3372	616.26	8.24	774
50	535.38	25.37	2575	583.57	33.85	3372	566.88	6.47	774
50	579.6	25.14	2575	594.77	31.91	3372	577.24	8.20	774
50	537.93	24.99	2575	551.02	31.42	3372	612.34	6.44	774
50	520.63	23.49	2575	494.77	33.05	3372	614.76	8.24	774
50	521.29	25.20	2575	588.93	31.93	3372	619.29	6.44	774
100	1222.99	98.40	2575	1234.61	125.09	3372	1234.61	28.84	774
100	1222.99	96.72	2575	1234.60	126.51	3372	1234.61	28.99	774
100	1234.61	102.66	2575	1206.01	133.07	3372	1326.80	28.96	774
100	1234.61	97.71	2575	1222.99	131.29	3372	1242.00	29.03	774
100	1222.99	106.04	2575	1234.61	139.85	3372	1234.61	29.06	774
100	1234.61	105.83	2575	1236.22	137.79	3372	1234.61	28.53	774
100	1234.61	102.16	2575	1222.99	134.14	3372	1236.22	28.59	774
100	1182.35	98.11	2575	1222.99	126.13	3372	1234.61	28.63	774
100	1222.99	98.32	2575	1222.99	125.53	3372	1222.99	28.74	774
100	1234.61	96.24	2575	1235.49	126.28	3372	1234.61	28.60	774

APPENDIX B: NICHING GENETIC ALGORITHM RESULTS

TABLE III: Niching Genetic Algorithm Results Across All City Sizes

Cities	**GA Std**	**Runtime** (s)	**Niches**	**GA High**	**Runtime** (s)	**Niches**	**GA Low**	**Runtime** (s)	**Niches**
5	499.66	5.74	2	499.66	10.44	1	499.66	1.21	2
5	499.66	3.87	1	499.66	8.73	1	499.66	1.16	2
5	499.66	5.70	2	499.66	10.64	1	499.66	1.20	2
5	499.66	6.11	1	499.66	10.43	1	499.66	1.23	2
5	499.66	5.41	1	499.66	10.43	1	499.66	1.21	2
5	499.66	6.90	1	499.66	9.04	1	499.66	1.23	2
5	499.66	3.72	1	499.66	10.40	1	499.66	1.74	2
5	499.66	3.86	1	499.66	10.01	1	499.66	1.48	2
5	499.66	6.13	2	499.66	8.97	1	499.66	1.16	2
5	499.66	6.38	2	499.66	10.08	1	499.66	1.11	2
15	289.63	23.33	7	289.63	89.67	8	289.63	3.82	5
15	289.63	20.08	7	289.63	88.39	7	289.63	4.39	6
15	289.63	21.06	7	289.63	87.79	7	289.63	4.08	5
15	289.63	19.41	4	289.63	89.92	8	289.63	4.06	5
15	289.63	21.24	7	289.63	87.22	6	289.63	4.51	3
15	289.63	21.16	7	289.63	91.54	6	289.63	3.96	6
15	289.63	20.80	7	289.63	90.45	7	289.63	3.81	6
15	289.63	27.05	7	289.63	91.19	7	289.63	4.73	7
15	289.63	20.21	7	289.63	86.44	8	289.63	3.92	5
15	289.63	21.96	7	289.63	90.21	6	289.63	4.15	7
25	292.76	38.33	8	229.12	130.43	12	330.11	7.03	10
25	292.02	39.49	7	243.62	141.62	6	374.54	6.38	7
25	309.57	38.14	5	259.74	150.59	8	340.18	6.56	8
25	294.85	34.84	6	256.37	147.55	7	345.27	7.12	5
25	292.02	33.73	7	252.58	150.65	7	351.08	6.32	8
25	269.72	35.31	8	240.72	146.52	7	330.11	7.25	7
25	292.76	33.00	6	254.28	154.43	6	269.72	6.30	11
25	290.72	39.99	7	254.12	137.92	6	351.08	7.47	8
25	292.02	41.90	9	276.42	145.33	8	342.97	7.18	6
25	276.42	37.96	10	252.58	82.16	6	374.54	6.35	6
50	550.63	23.31	2	511.04	59.28	2	624.46	5.94	1
50	553.35	15.96	4	522.14	50.83	2	622.44	8.86	2
50	588.93	19.35	2	529.77	48.77	2	598.83	6.26	2
50	591.85	20.88	2	522.14	56.42	1	610.21	8.77	1
50	559.41	23.35	3	529.77	68.85	2	618.99	6.81	2
50	554.12	15.63	2	543.56	57.67	1	618.99	8.50	2
50	581.14	15.60	1	522.14	55.66	2	624.46	9.20	2
50	550.63	16.99	2	529.77	64.35	2	618.99	5.63	1
50	550.63	16.42	3	504.39	58.16	3	619.20	8.75	2
50	581.75	20.21	2	528.50	52.54	2	617.02	6.90	1
100	1182.35	94.85	7	1182.35	408.68	6	1222.99	36.84	9
100	1188.48	99.88	8	1182.17	433.58	6	1206.01	20.09	4
100	1182.17	106.11	8	1181.73	424.28	6	1234.60	26.93	8
100	1182.35	90.34	5	1181.73	408.83	4	1222.99	20.17	3
100	1181.73	102.05	9	1181.73	376.90	5	1238.31	20.78	3
100	1187.33	98.85	7	1181.73	455.95	4	1234.61	19.44	8
100	1182.35	96.73	6	1182.17	416.26	5	1234.60	31.53	5
100	1206.01	89.88	5	1181.73	426.08	7	1206.01	28.48	5
100	1182.17	95.37	8	1181.73	460.48	6	1234.44	24.34	2
100	1181.73	92.20	4	1181.73	391.15	5	1182.35	26.10	5

APPENDIX C: LKH HEURISTIC RESULTS

TABLE IV: LKH Heuristic Results Across All City Sizes

Cities	**LKH 1**	**Runtime** (s)	**LKH 2**	**Runtime** (s)
5	581.03	0.00051	499.66	0.00044
5	499.66	0.00044	499.66	0.00037
5	499.66	0.00033	581.03	0.00064
5	499.66	0.00037	499.66	0.00041
5	499.66	0.00029	581.03	0.00034
5	499.66	0.00037	581.03	0.00036
5	581.03	0.00024	581.03	0.00029
5	499.66	0.00025	499.66	0.00030
5	499.66	0.00027	581.03	0.00056
5	581.03	0.00059	499.66	0.00061
15	379.28	0.00941	289.63	0.03462
15	289.63	0.02944	379.28	0.02426
15	289.63	0.02586	289.63	0.03401
15	379.28	0.01674	289.63	0.02598
15	289.63	0.01296	289.63	0.01867
15	379.28	0.00849	416.48	0.01702
15	379.28	0.03419	289.63	0.01464
15	289.63	0.02831	289.63	0.04084
15	289.63	0.02502	379.28	0.01821
15	379.28	0.02394	289.63	0.01713
25	220.14	0.07664	340.18	0.20218
25	220.14	0.11538	624.37	0.27634
25	220.14	0.18453	374.54	0.38800
25	220.14	0.07631	374.54	0.29641
25	254.12	0.09420	374.54	0.10844
25	220.14	0.07028	340.18	0.30117
25	220.14	0.11387	410.37	0.18855
25	220.14	0.09719	252.58	0.40807
25	290.72	0.06599	485.12	0.09506
25	226.09	0.12302	220.14	0.10738
50	239.06	1.6717	223.90	2.66634
50	336.47	0.89744	233.85	3.08273
50	236.29	1.21293	223.90	3.87311
50	473.82	0.48876	223.90	2.64723
50	259.47	0.95540	233.85	2.84006
50	237.02	1.11584	233.85	2.55444
50	256.10	1.01142	223.90	3.66561
50	400.43	0.46545	223.90	4.23636
50	244.53	0.79711	223.90	3.09152
50	327.37	0.79711	223.90	2.95481
100	1206.01	0.88525	1181.73	1.52499
100	1240.93	0.73358	1234.60	2.68339
100	1234.49	0.81373	1181.73	3.51893
100	1182.17	1.04915	1182.17	2.31622
100	1187.33	0.79943	1182.17	5.71876
100	1246.37	1.90931	1182.17	2.85293
100	1187.33	1.99386	1181.73	1.72814
100	1182.17	1.33426	1181.73	6.31659
100	1240.95	0.65428	1181.73	2.39877
100	1206.01	0.82165	1182.17	5.07079