1 Introduction

The traveling salesman problem (TSP) is one of the most intensively studied problems in optimization. The implementation of the TSP problem in this paper is formulated as: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once."

The traveling salesman problem is NP-hard, and the solution space scales factorially with the number of cities. The time complexity of the brute force approach is $O(n!)$, which makes a TSP with fairly large number of cities infeasible to solve with modern computers.

A promising approach for solving the TSP problem is local search algorithms. Local search algorithms start with a random solution, which is far from the optimal solution, and perform incremental perturbations on the solution to gradually improve it over many iterations. The TSP has been tackled with three different algorithms in this paper: Hill Climbing, Simulated Annealing, and an Evolutionary Algorithm.

2 Algorithm Description

2.1 Hill Climbing

The hill climbing algorithm is one of the most naive approaches in solving the TSP. A random solution, a random sequence of cities, is generated. Then, all successor states of the solution is evaluated, where a successor state is obtained by switching the ordering of two cities adjacent in the solution. Finally, the best successor is chosen as the new state, and its successors are evaluated, and this continues until a satisfactory solution is obtained. The pseudocode for the algorithm is given in Algorithm 1.

2.2 Simulated Annealing

The simulated annealing algorithm promotes aggressively searching for good solutions in the early iterations, and gradually stabilizes and continues search locally. The idea is to avoid local minima in the first iterations, and find the

Algorithm 1 Hill Climbing

```
1: Start from a random state (random order of cities)
2: Generate all successors (all orderings obtained with switching any two adjacent cities)
3: Select successor with lowest total cost
4: Go to step 2
```

best solution in the vicinity in the later iterations. The pseudocode for the algorithm is given in Algorithm 2.

Algorithm 2 Simulated Annealing

```
1: Start from a random state (random order of cities)
2: Perturb solution by changing the order of two adjacent cities
3: if new solution is superior then
4:     Replace the previous state
5: else
6:     Replace the previous state with probability $e^{\Delta L/T}$
7: $T = T - \Delta T$
8: Go to step 2
```

2.3 Evolutionary Algorithm

The evolutionary algorithm implemented in this work starts with a population P of n random solutions. m of the solutions are selected and mutated. The selection is based on the quality of the solutions, where the probability distribution is inversely proportional to the squared errors. The mutated solutions are added to the population, and the worst m are eliminated to bring the population count back to n. The mutations, similar to the other algorithms, are performed by switching the order of two cities. The pseudocode for the algorithm is given in Algorithm 3.

Algorithm 3 Evolutionary Algorithm

```
1: Generate population $P$ with $n$ random solutions
2: Select and mutate $m$ solutions, based on their squared performance
3: Add the mutated solutions to the population
4: Evaluate fitness (negative of total cost) for each member
5: Eliminate $m$ worst solutions
6: Go to step 2
```

Table 1: Experimental Results. Smallest costs are in bold.

Algorithm	**TSP**	**Time (s)**	**Cost (m)**
Hill Climbing	15 Cities	0.003	8755
	25 Cities	0.013	12541
	25 Cities A	0.014	17567
	100 Cities	0.179	61802
Simulated Annealing	15 Cities	0.105	**6200**
	25 Cities	0.188	**9221**
	25 Cities A	0.190	13362
	100 Cities	0.671	**47231**
Evolutionary Algorithm	15 Cities	0.116	6567
	25 Cities	0.335	10138
	25 Cities A	0.328	**12044**
	100 Cities	1.268	52499

3 Experiments

3.1 TSP with 15 cities

A TSP with 15 cities has been solved with the three algorithms. The performance curves of the algorithms are given in Figure 2. Each algorithm has been tested 25 times, and the mean performance is plotted with standard errors. The parameters T and ΔT have been experimentally selected as 500 and 0.075, respectively. Parameters n and m have been selected as 10 and 5, respectively. The layout of the 15 cities in this TSP is given in **??**(a). Results on solution quality and time elapsed is given in table Table 1.

The hill climbing algorithm underperformed compared to the other two algorithms, which performed similarly. It took under 10 iterations for the hill climbing algorithm to reach a local minimum, which makes it the fastest algorithm due to its greedy nature, but the solution quality is much lower than the other two algorithms. Simulated Annealing algorithm has a slightly better solution compared to the evolutionary algorithm, but requires a higher number of iterations to compute. Despite the higher iteration count, the time elapsed is actually smaller, as in each iteration of the evolutionary algorithm, 10 solutions are evaluated and mutated. The early stages of the simulated annealing algorithm demonstrates a more erratic behavior. The T value is high in the early iterations, which results in a noisy graph, even with averaging over 25 times. However, as the T value decreases, the performance becomes much more stable.

3.2 TSP with 25 cities

A TSP with 25 cities has been solved with the three algorithms. The performance curves of the algorithms are given in Figure 3. Each algorithm has been tested 25 times, and the mean performance is plotted with standard errors. The

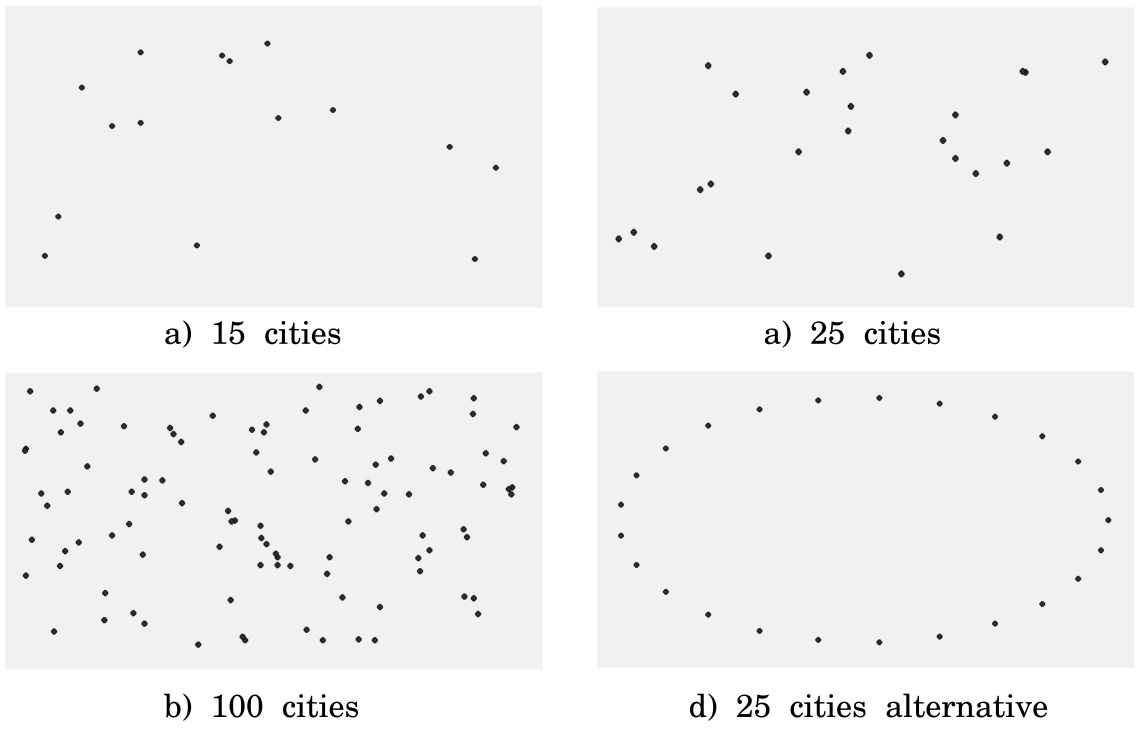

Figure 1: The city layouts in the TSPs.

parameters T and ΔT have been experimentally selected as 500 and 0.075, respectively. Parameters n and m have been selected as 10 and 5, respectively. Once again, simulated annealing reports the best results, and hill climbing takes the least amount of time to find a solution. Hill climbing finds a solution in an order of magnitude shorter time compared to other algorithms, but the solution quality is around %25 worse than the other algorithms. Results on solution quality and time elapsed is given in table Table 1.

3.3 TSP with 25 cities, alternative layout

In this TSP, 25 cities with a deterministic layout is tested. In all other examples, the cities are sampled from what seems to be a bounded uniform random distribution. However, in this example, the cities are placed on an elliptical shape. This unique layout makes the optimal solution very obvious to a human eye. Tracing the ellipse will obviously yield the best result. However, the local search algorithms discussed in this paper does not have access to this intuition. Another characteristic of this layout is the fact that finding near optimal solutions are more difficult compared to random layouts, as there exists many near optimal solutions in other layouts, whereas in this layout the optimal solution seems to be isolated from solutions. Performance curves are reported in Figure 4, and results on solution quality and time elapsed is given in table Table 1.

In this problem, the evolutionary algorithm outperforms the other two in terms of solution quality. This is because of its strength against local minima, as searching with a population is less susceptible to local minima compared to iterating only one solution. In TSPs with randomly distributed cities, it is easier to find a solution that is good enough, but in this example, the optimal solution is harder to find, and population based search algorithms have a greater chance locating it.

3.4 TSP with 100 cities

A TSP with 100 cities has been solved with the three algorithms. The performance curves of the algorithms are given in Figure 5. Each algorithm has been tested 25 times, and the mean performance is plotted with standard errors. The parameters T and ΔT have been experimentally selected as 500 and 0.075, respectively. Parameters n and m have been selected as 10 and 5, respectively. Results are similar to other randomly distributed city samples. Simulated annealing has the best final results, but requires the largest number of iterations.

3.5 Explored Solutions

The results on the unique solutions explored for each TSP, the number of total solutions, and the percentage of solutions for each algorithm are reported in Table 2. The number of explored solutions is drastically insignificant compared to the number of total solutions. For the TSPs with 15 cities, there exists about 10^{12} solutions. To put things into perspective, 10^{12} is larger than the number of

Table 2: Results on solutions searched.

Algorithm	**TSP**	**Explored**	**# Solutions**	**% Explored**
Hill Climbing	15 Cities	6.6	$\sim 10^{12}$	$\sim 10^{-12}$
	25 Cities	10.4	$\sim 10^{25}$	$\sim 10^{-24}$
	25 Cities A	11.2	$\sim 10^{25}$	$\sim 10^{-24}$
	100 Cities	45.6	$\sim 10^{157}$	$\sim 10^{-156}$
Sim. Annealing	15 Cities	1676	$\sim 10^{12}$	$\sim 10^{-9}$
	25 Cities	2612	$\sim 10^{25}$	$\sim 10^{-23}$
	25 Cities A	1619	$\sim 10^{25}$	$\sim 10^{-23}$
	100 Cities	26473	$\sim 10^{157}$	$\sim 10^{-153}$
Evo. Algorithm	15 Cities	9376	$\sim 10^{12}$	$\sim 10^{-9}$
	25 Cities	20937	$\sim 10^{25}$	$\sim 10^{-22}$
	25 Cities A	18240	$\sim 10^{25}$	$\sim 10^{-22}$
	100 Cities	25623	$\sim 10^{157}$	$\sim 10^{-153}$

trees on earth. Even though the number of explored solutions is small, the more intelligent local search algorithms, like simulated annealing, can find satisfactory solutions in reasonable time. As the number of cities increase, the total number of solutions increase factorially, which is faster than exponential increase. With 25 cities, there are about 10^{25} solutions, which is roughly around the number of stars in the observable universe. The evolutionary algorithm searches more solutions compared to simulated annealing. The local search only searches a small number of solutions before it hits a local minima. The number of solutions for a TSP with 100 cities is 10^157, which is much larger than the number of atoms in the observable universe. The percentage of solutions explored gets smaller as the number of cities increase, as the increase in number of solutions explored is insignificant compared to the growth of the solution space.

3.6 Bonus Figures

Figure 6 demonstrates a decent solution found by the simulated annealing algorithm for 25 cities. Figure 7 demonstrates the poor solution found by the hill climbing algorithm, for the same problem.

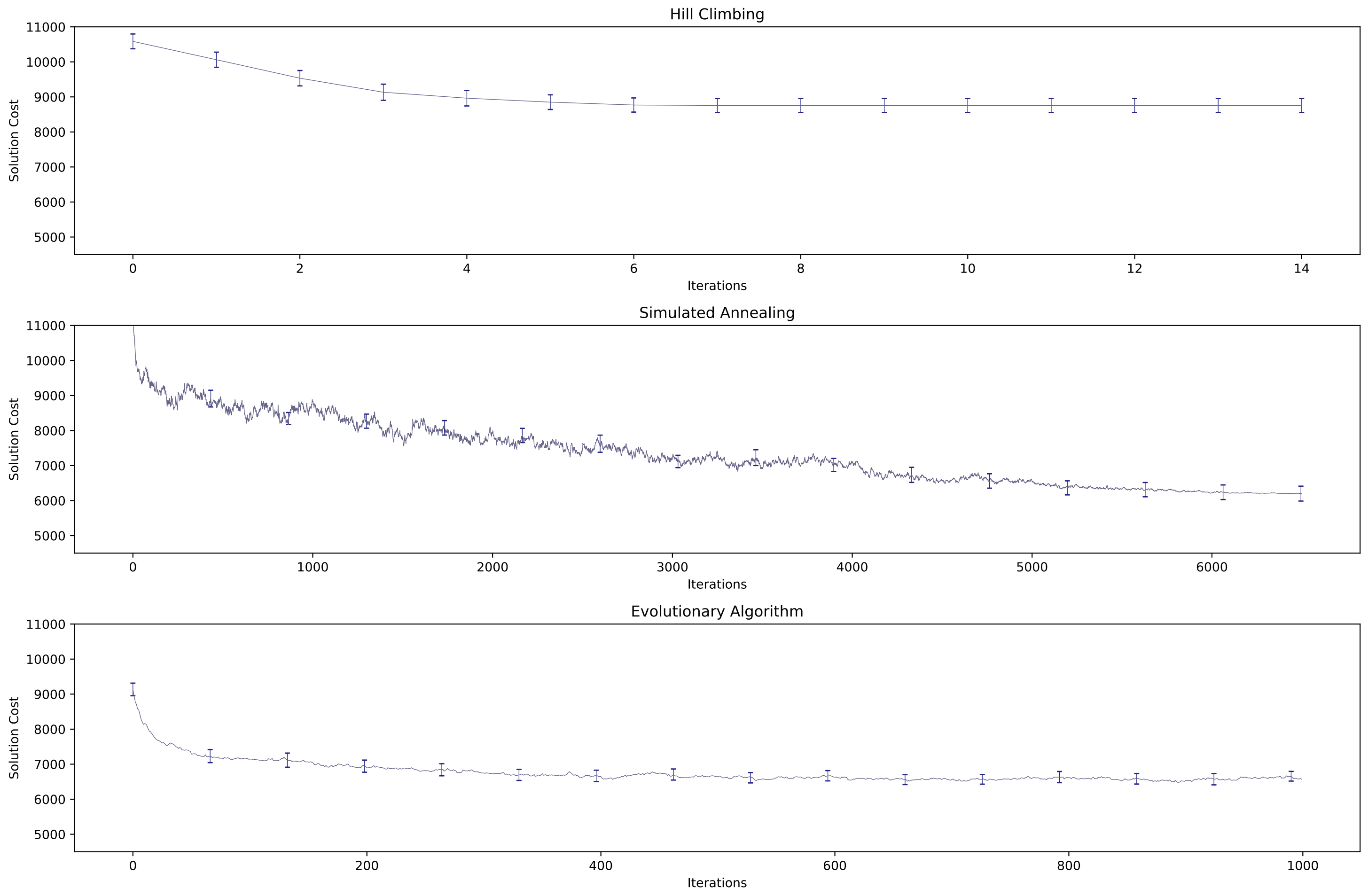

Figure 2: Performance curves for the three algorithms for a TSP with 15 cities.

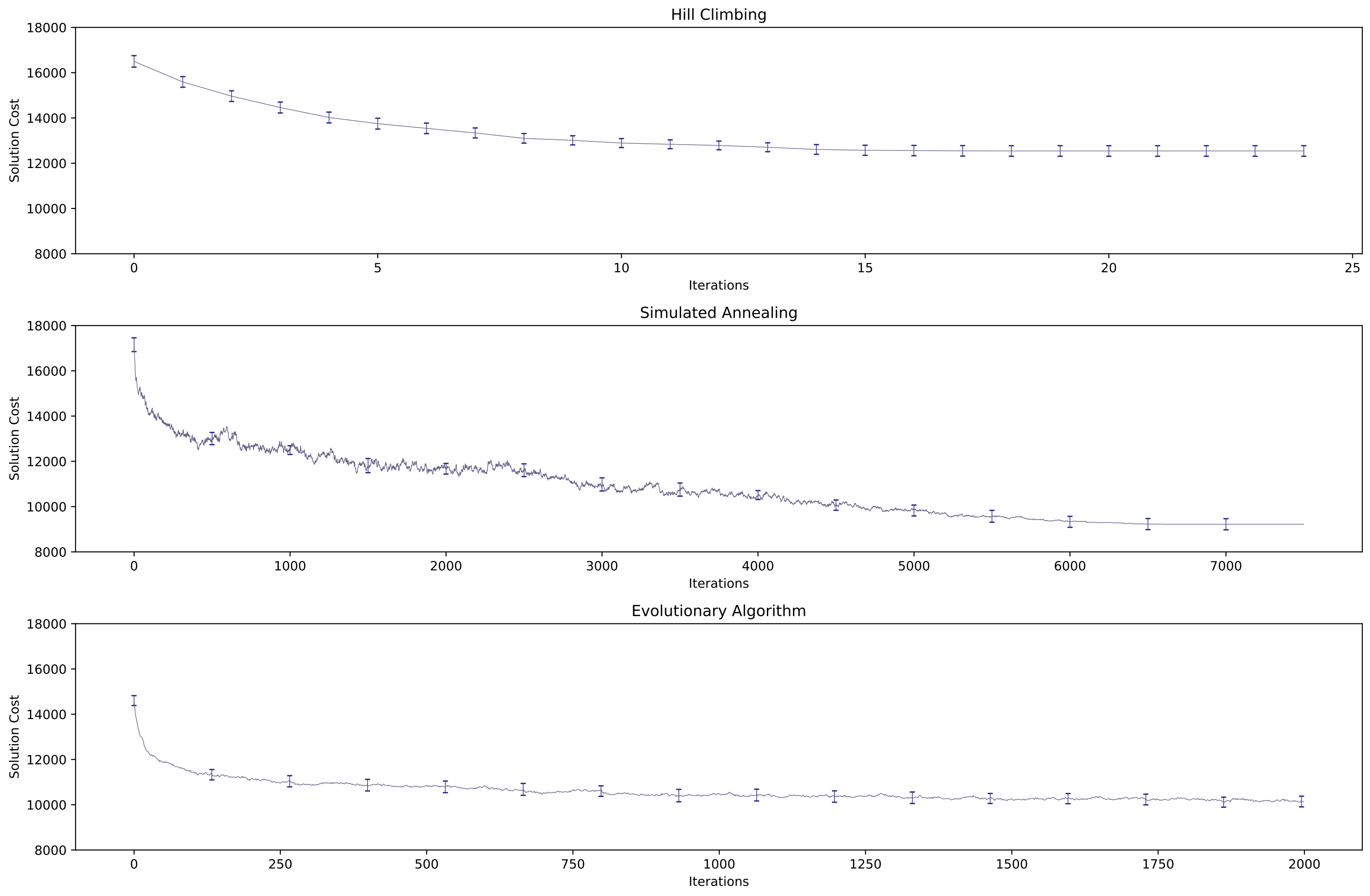

Figure 3: Performance curves for the three algorithms for a TSP with 25 cities.

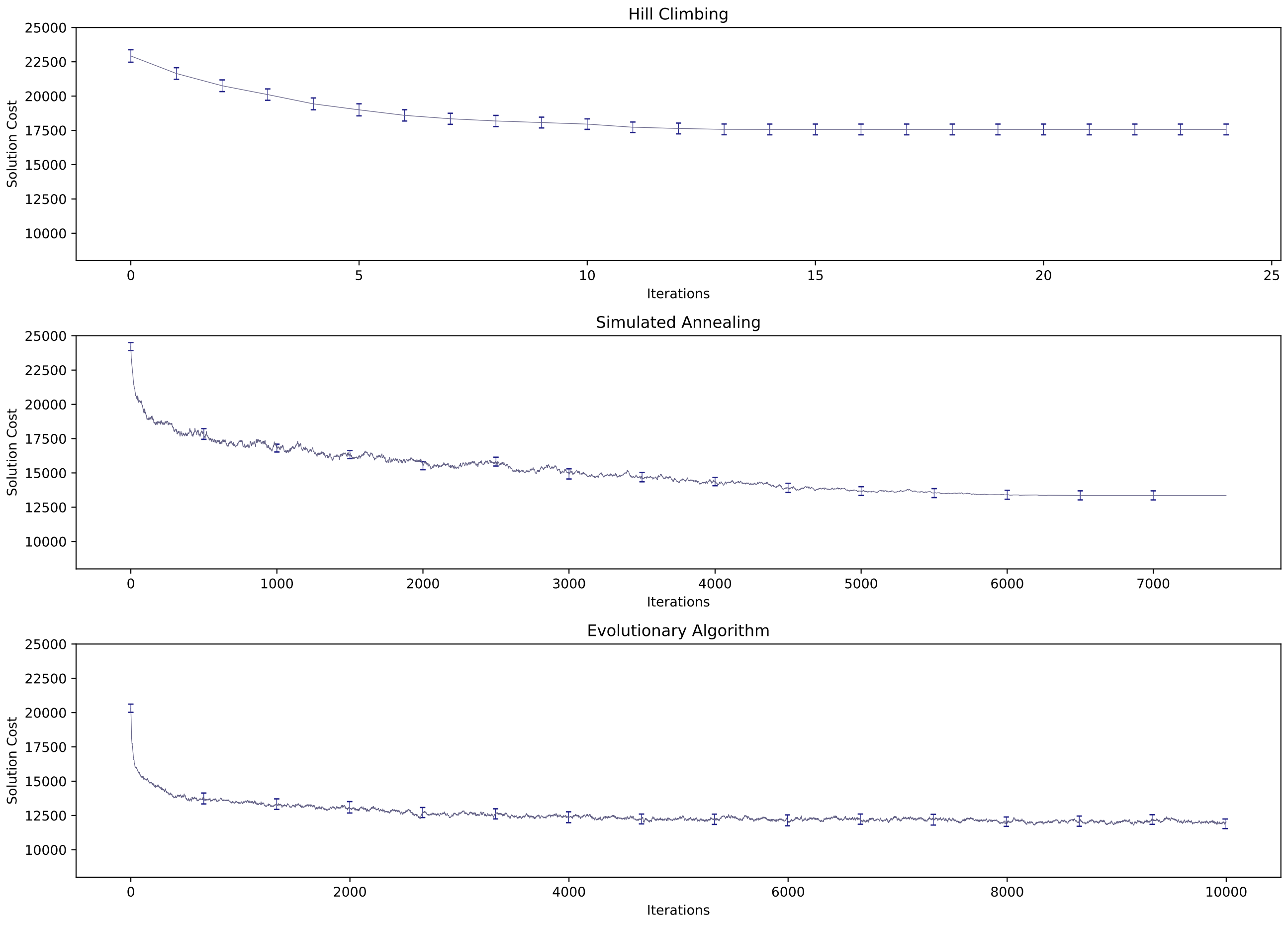

Figure 4: Performance curves for the three algorithms for a TSP with 25 cities, with an alternative layout.

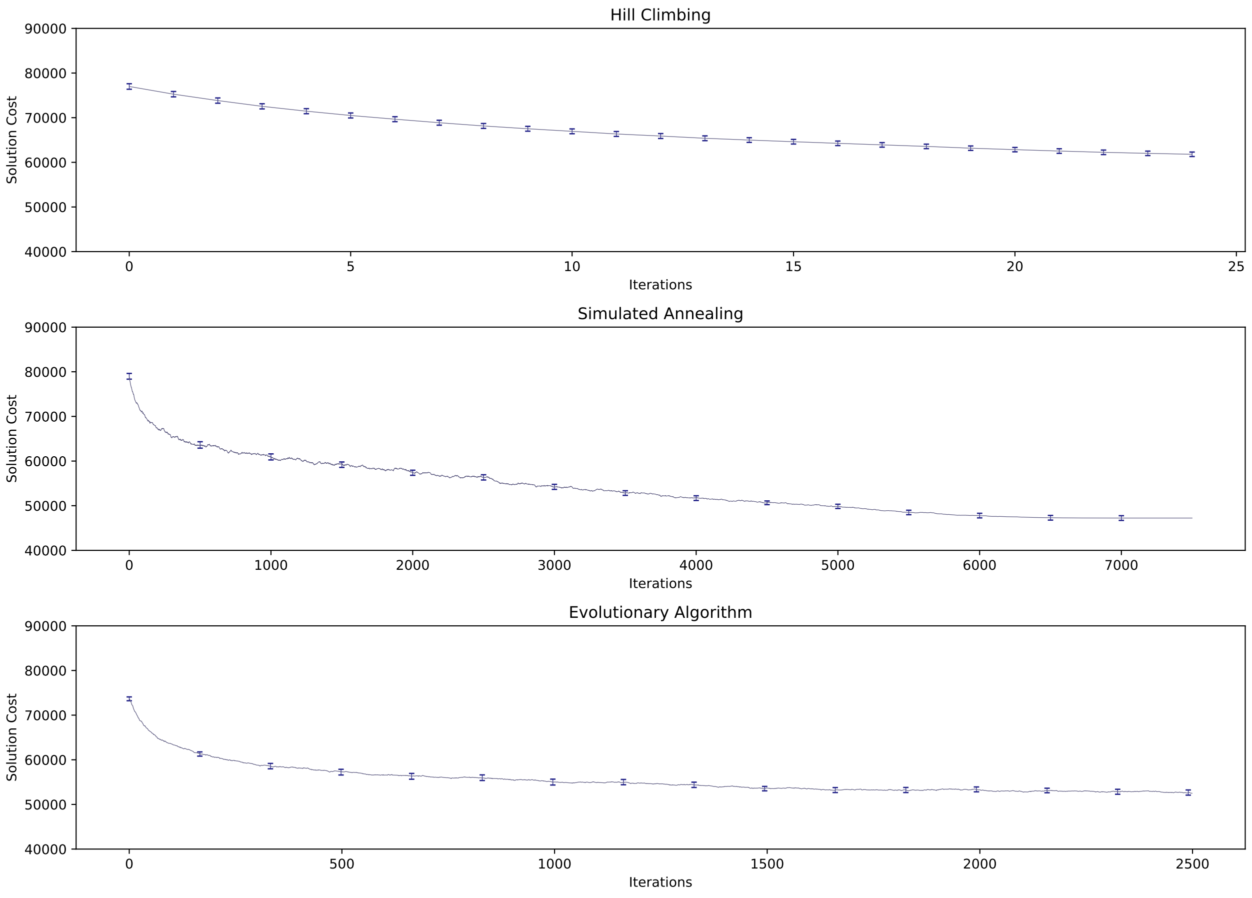

Figure 5: Performance curves for the three algorithms for a TSP with 100 cities.

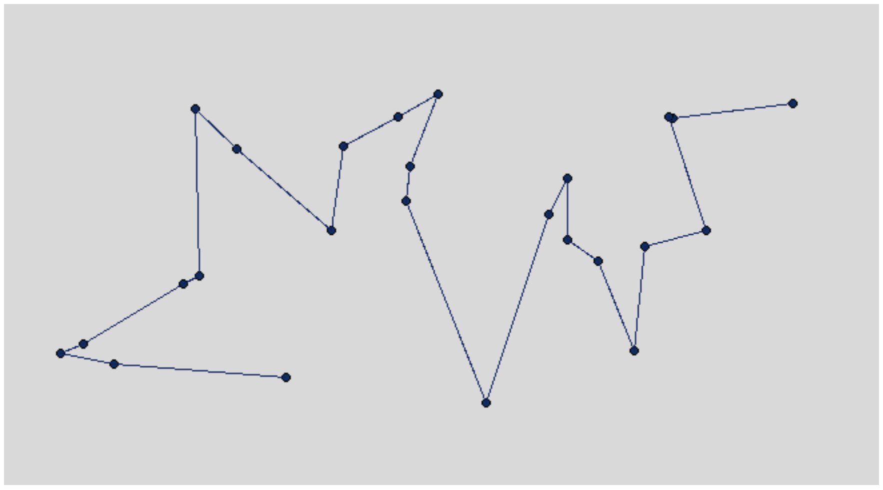

Figure 6: A good solution found by the simulated annealing algorithm for TSP with 25 cities.

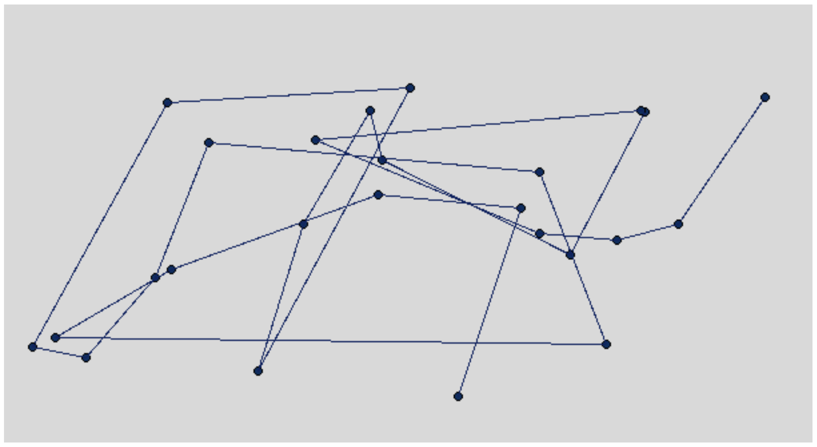

Figure 7: A poor solution found by the hill climbing algorithm for TSP with 25 cities.