CSE 6140 / CX 4140 Project - TSP Report

Haoyang Zeng, Jiayi Qian, Kaiwen Gong, Zheng Wang, Zhikai Xu
Georgia Institute of Technology

Abstract

The Traveling Salesman Problem (TSP) is encountered in a wide range of areas including vehicle routing, circuit board drilling, and VLSI design. In this project, we solve TSP by using four different algorithms, that is, brute-force, approximate algorithm based on MST, genetic algorithm, and hill climbing algorithm. It is found that genetic algorithm (local search) has the best balance between solution quality and computational time. Moreover, by changing the cutoff time for each four algorithm, we found that the obtained total cost decreases along with the increase of cutoff-time, which meets the expectation.

1 Description of Algorithms

1.1 Brute Force Algorithm

Brute Force Algorithm is implemented to find all the permutations (paths) of the points and calculate the minimum distances among the permutations. To optimize the brute force algorithm, backtracking is utilized to search all the permutations recursively. The core of the backtracking algorithm consists of two key components:

1. The check for the time limit is performed at the start of the backtracking method, ensuring the method exits if the time limit is reached.
2. Recursively exploring all possible routes and updating the best path and minimum distance when a shorter route is found.

1.2 Approximate Algorithm

For NP-complete problems, we can't find optimal solutions in polynomial time. Therefore, the Approximate Algorithm is implemented to find a near-optimal solution in polynomial time. For TSP, the approximation is achieved using the Preorder tree walk on the Minimum spanning tree of the graph, which is proved to be a 2-approximation algorithm. Key principles of Approximate Algorithm include:

1. Implementing Prim or kruskal algorithm to get the minimum spanning tree of the graph.
2. performing a preorder tree walk on generated the minimum spanning tree, with the triangle inequality, we can prove that it's a 2-approximation.

1.3 Hill Climbing Algorithm

Hill Climbing is a probabilistic technique that allows the acceptance of worse solutions with a certain probability, potentially escaping local optima to find a global optimum. In SA, a move from a local optimum, say B, to a suboptimal point D, can occur with a specific likelihood, enabling the algorithm to progress towards a global optimum C.

Key principles of Hill Climbing include:

1. Accepting better solutions unconditionally.
2. Accepting worse solutions with a probability decreasing over time, which is controlled by a decay parameter t. The likelihood of overcoming a long slope is low, but with appropriate t settings, it can be possible.

1.4 Genetic Algorithm

Evolutionary Algorithm (EA): Evolutionary Algorithms use mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection.

Key principles of Evolutionary Algorithm include:

1. Accepting Better Solutions Unconditionally: In EA, if a newly generated solution (or individual) is better than existing ones in the population (for example, has a shorter route in the Traveling Salesman Problem), it is accepted into the next generation. This is analogous to the survival of the fittest in natural evolution.
2. Mutation and Crossover: These are key mechanisms in EAs. Mutation introduces random changes to some individuals, while crossover combines parts of two or more individuals to create new offspring. These processes introduce variability in the population, encouraging exploration of the search space.
3. Selection Process: The selection process determines which individuals are chosen to reproduce and thus contribute to the next generation. Various selection strategies exist, like roulette-wheel selection or tournament selection, each with its advantages in maintaining diversity and driving the population towards optimal solutions.
4. Convergence towards Optimal Solutions: Over successive generations, the population of solutions in an EA tends to converge towards optimal or near-optimal solutions. This process is guided by the combined effect of selection, mutation, and crossover, along with the algorithm's acceptance of both superior and occasionally inferior solutions.

2 Results and Analysis

The experiments' results of addressing TSP problem via different algorithm are listed in Table 1. For approximation, hill climbing, and genetic, $Sol.Quality$ represents the average value of the ten running cases for each city with different random seed, and $RelError$ represent the difference between the best or optimal solution and the obtained average value.

Table 1. Experiments' results of solving TSP via brute force, approximation, hill climbing, and genetic algorithms.

	Brute Force			Approximation			Hill Climbing			Genetic		
Dataset	Time(s)	Sol.Quality	Full Tour	Time(s)	Sol.Quality	RelError	Time(s)	Sol.Quality	RelError	Time(s)	Sol.Quality	RelError
Atlanta	300	3805207	No	0.0002	2340581.6	123625.6	0.0235	2274020.3	243344.2	2.2762	2046907.6	23129.2
Berlin	300	19728	No	0.0013	10022.4	434.4	1.1075	10521.7	1134.7	5.9739	8200.9	836.5
Boston	300	2246134	No	0.0007	1066217.0	38422.0	0.4229	1103069.5	91890.3	4.5277	974761.9	43669.5
Champaign	300	218823	No	0.0014	67415.5	863.5	1.5951	73791.1	9235.6	6.4154	62812.6	5163.8
Cincinnati	127.12	277952	Yes	5.596e-5	306118.1	6836.1	0.0014	278903.7	951.1	1.1665	278639.5	686.9
Denver	300	563620	No	0.0031	129060.8	1503.8	8.8323	165243.8	20419.6	10.1500	144058.9	16780.8
NYC	300	7244933	No	0.0020	1905305.1	84401.1	4.0206	2235795.0	196538.3	8.0098	1970956.4	62100.6
Philadelphia	300	3710782	No	0.0004	1702961.7	17678.7	0.1217	1767073.0	184976.2	3.3414	1423985.9	19985.0
Roanoke	300	6857992	No	0.0261	799513.9	5222.9	627.1105	1574503.7	140145.6	33.1678	1884984.4	237294.8
SanFrancisco	300	5604793	No	0.0044	1086371.7	18263.7	15.5640	1574753.4	175682.2	12.4575	1383940.9	82577.7
Toronto	300	9219353	No	0.0054	1664101.1	42498.1	23.0488	2377742.9	271858.5	13.5241	2121122.0	179347.0
UKansasState	134.82	62962	Yes	5.589e-5	70640.1	3004.1	0.0013	62962.3	0.0	1.2147	62962.3	0.0
UMissouri	300	668983	No	0.0066	169077.6	4677.6	19.7563	212318.4	15649.2	13.0700	212258.1	17919.9

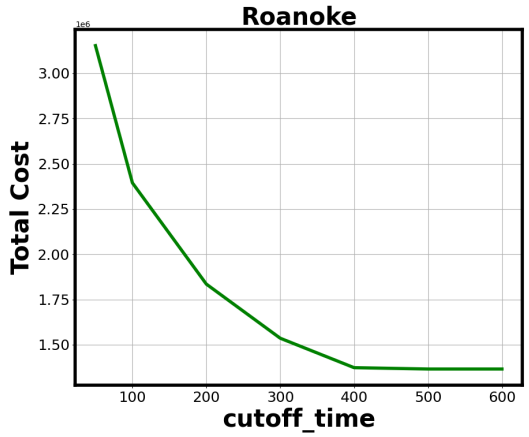

Figure 1. The impact of different cutoff-times on Hill Climbing algorithm for Roanoke.

Specifically, brute force guarantees the best solution but is often impractical due to high computational time. Moreover, aside from Cincinnati and UKansasState, all the remaining cities are not found the full tour by brute force algorithm.Therefore, if the time limited and full tour guaranteed, brute force is not the first choice though it can find the best solution. In terms of the computation time, approximation is the fastest method among the four but with significant error. Significantly, Hill Climbing cannot lead to good solutions for most cities, and it is much more inconsistent compared with approximation and genetic as well. Its performance can degrade depending on the landscape of the problem. For genetic algorithms, it offer a balance between solution quality and computational time, often providing better solutions than Approximation and outperforming Hill Climbing, but with higher computation time than Approximation. However, the computation time could be neglected compared to that of brute force algorithm. Note that local search algorithms including genetic and hill climbing algorithm even find the best solution for UMissouri the same as the brute force but with a dramatic decrease of computation time.

Additionally, the impact of different cutoff time on brute force is explored as well, and the corresponding results are shown in Figure 2. Six different cutoff-times are tested, that is, 50s, 100s, 200s, 300s, 400s, 500s, and 600s. The TSP solution (total route cost) for most cities decrease along with the increasing cutoff-time. While for Cincinnati and UKansasState, the solution does not change because the optimal solution is found. Besides, for hill climbing and genetic algorithms, we only test the impact of different cutoff time on the city with the final running time greater than 600s, that is, the Roanoke city with hill climbing, and the corresponding result is shown in Figure 1. It can be seen that the total route cost of Roanoke city is also dropping along with the cutoff-time rising via Hill Climbing algorithm. When the cutoff time reaches to 400s, the algorithm can find the best solution for this city.

3 Conclusion

In this experiment, four algorithms are tested to solve TSP problem, that is brute-force, approximate algorithm based on MST, genetic algorithm, and hill climbing algorithm. It is found that though brute-force can find the best solution, its high computation cost makes it unfeasible to use in practice. On the contrary, Genetic algorithm balance computation cost and solution quality the best, which is a more feasible choice in real world use. Moreover, even though both Hill Climbing and Genetic are local search algorithms, overall the latter one performs much better than the former one in terms of the experiments' results.

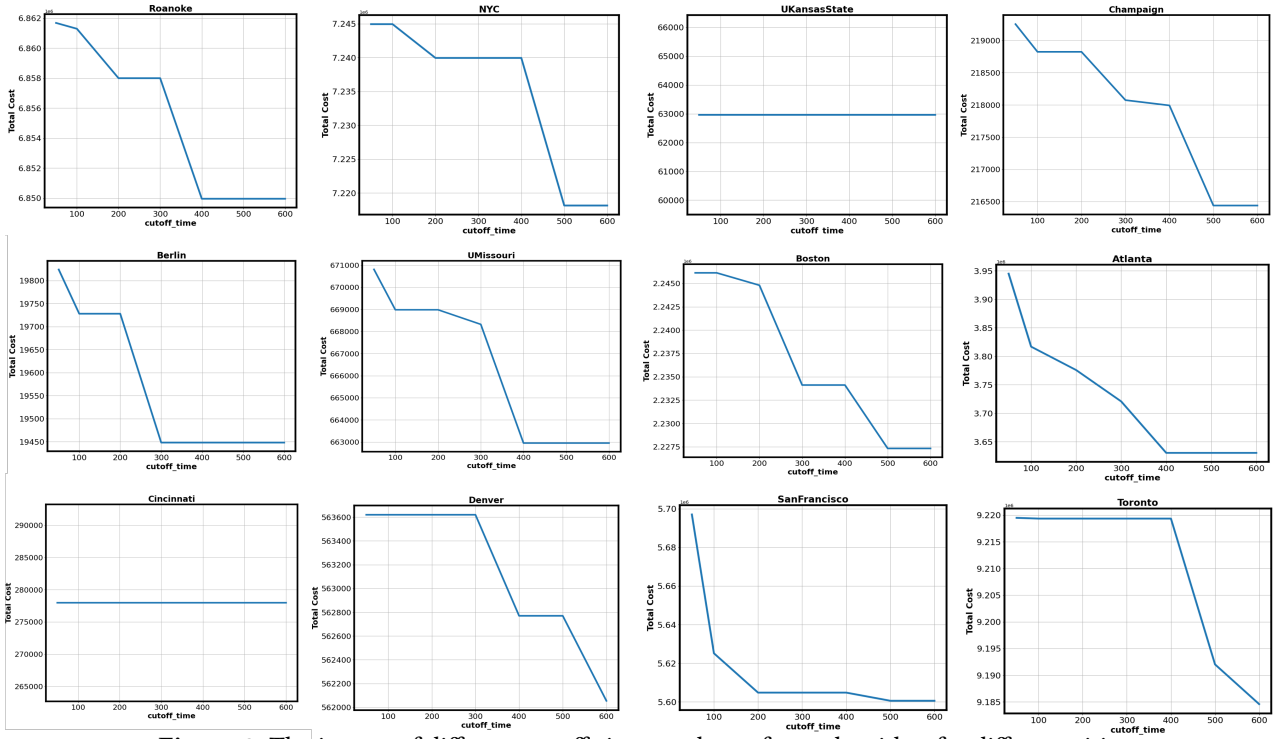

Figure 2. The impact of different cutoff-times on brute force algorithm for different cities.