CS 6601-A Artificial Intelligence Project 1 Report
A Genetic Algorithm for Symmetric TSP

1 Problem statement

I proposed to implement a genetic algorithm [5] to solve the Traveling Salesman Problem. TSP is a classic problem in combinatorial search and optimization that has been well studied. It is often used as a benchmark for combinatorial or black-box optimization algorithms, and is one of the best known problems in graph theory. Applications of TSP include routing (shipping and logistics), job scheduling, and even the analysis of crystal structure [1].

2 Related work

TSP can be formulated as a search problem and solved using standard tree-search algorithms, however the state space for such a problem is very large (on the order of $n!$ where n is the number of nodes in the graph). Evolutionary algorithms, like the GA, provide an efficient way to search large state spaces in parallel. Using a generic GA implementation to solve optimization problems tends to lead to poor performance, so a lot of work has gone into developing TSP specific genetic operators and other heuristic modifications of the standard GA, such as seeding the initial population [4], local optimization pre-processing of individuals [6], and specialized genetic operators [8], One approach [3] which implements several of these heuristics has reported some impressive results on instances from the TSPLIB data set [7].

3 Approach

The Genetic Algorithm is a meta-heuristic belonging to the class of Evolutionary Algorithms. In general, it is an global optimization technique that searches a large space of candidate solutions in parallel. The algorithm works by sampling the search space with a set of *individuals* (called the *population*), evaluating the *quality* of each of these individuals, and then sampling a new set of individuals based on combining favorable traits of individuals from the old population. In terms of the TSP, an individual represents one possible tour through the graph, and the quality of an individual is the tour's cost. New individuals can be created by combining low cost fragments of tours from individuals from the previous population. The insight of [3] had to do with clever was of performing this combination, and techniques for limiting the search space to local optima.

I originally planned to re-implement the algorithm given in [3] for Symmetric TSP, however due to time limitations I had to scale back my approach. The original algorithm improves upon the standard GA by introducing two heuristics. The first, a crossover operator which combines two different tours by incorporating their common sub-paths, or solution fragments, and then connecting the tour fragments using the lowest cost edges. This exploits a result from [2] that shows that the average distance from two distinct locally-optimal tours is equal to the distance to the globally optimal tour. By using this crossover technique, its more likely that the child created from the operation will be the global optimum. I implemented this crossover technique as it was described. The second heuristic used in [3] was the *Lin-Kernighan Heuristic*, which is a technique for improving sub-optimal tours. The original LKH algorithm is fairly complicated, so

problem	generations/pop	best	average
eli51.tsp	18/10	441.997 (3.7%)	441.997
kroA100.tsp	8/10	22539.4 (5.9%)	23470.8
att532.tsp	98/20	91949 (232.1%)	93308.3
rat783.tsp	128/20	9507.61 (7.9%)	9625.26

Figure 1: Best and average quality of tours found using the implemented algorithm. % deviation from optimal shown in parenthesis.

instead I implemented the much more straightforward *2-opt* heuristic, where a tour is iteratively improved by replacing two sub-optimal edges at a time. The LKH algorithm is actually a generalization and extension of the k-opt heuristic which considers more complicated sub-path replacements, and both are considered local search techniques for TSP. By exploiting these two heuristics, the GA can perform a focused parallel search among local optima.

4 Evaluation

The results of running the algorithm on 4 test problems from the TSPLIB data-set are given in Figure 1. The number of generations the GA was run for and the number of individuals in the population is given in the first column, which corresponds with the parameters used in [3]. The reported best and average results were out of either 20 runs (for eli51.tsp and kroA100.tsp) or 10 runs (for att532.tsp and rat783.tsp). On all 4 of these problems, the original algorithm from [3] finds the optimal tour length with the same parameters, so the % deviation reported is also the difference in performance between my implementation, and the one given in [3].

5 Discussion

Figure 1 shows that the two algorithms (the original LKH-based algorithm, and my implementation based on the 2-opt heuristic) perform comparably, except in one instance. This is understandable considering the LK heuristic is an extension of a more general class of local TSP search. Intuitively, whats happening here is that the LK heuristic narrows the search space to a smaller set of local optima, allowing the GA to more efficiently search among the candidate solutions. The large discrepancy on the att532.tsp problem might be explained by a graph structure that is not effectively exploited by the 2-opt heuristic, but which is by LKH. Further study of the particular structure of that graph (specifically, whether 2-opt yields pathological sub-optima, or whether the space of local optima is just too large for the GA to search effectively) would be an interesting area of investigation.

In my original proposal, I said wanted to see if the algorithm performed well on larger instances, now that computing power has increased drastically from the time the original paper was written. After some initial testing, it was clear that the running time for the larger instances was still taking a long time (much faster than the original paper's multiple hours per run, but slower than you would expect after a decade's worth of improvement in computer processing power). The reason for this is that even finding *local optima* on large combinatorial problems can approach intractability. The take-away message from this is that for this approach to work, you need to have an efficient method for finding locally optimal tours, and that the effectiveness of this algorithm is tied to how quickly and reliably the underlying heuristic can find these local optima.

References

[1] R. Bland and D. Shallcross. Large travelling salesman problems arising from experiments in X-ray crystallography: A preliminary report on computation* 1. *Operations Research Letters*, 8(3):125–128, 1989.

[2] K. Boese. *Cost versus distance in the traveling salesman problem.* Citeseer, 1995.

[3] B. Freisleben and P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 616–621. IEEE, 2002.

[4] J. Grefenstette. Incorporating problem specific knowledge into genetic algorithms. *Genetic algorithms and simulated annealing*, 4:42–60, 1987.

[5] J. Holland. *Hidden order: How adaptation builds complexity.* Basic Books, 1996.

[6] H. Muhlenbein. Evolution in time and space-the parallel genetic algorithm. In *Foundations of genetic algorithms.* Citeseer, 1991.

[7] G. Reinelt. TSPLIB–A traveling salesman problem library. *INFORMS Journal on Computing*, 3(4):376, 1991.

[8] N. Ulder, E. Aarts, H. Bandelt, P. van Laarhoven, and E. Pesch. Genetic local search algorithms for the traveling salesman problem. *Parallel problem solving from nature*, pages 109–116, 1991.