Group Project Proposal: Travelling Salesperson Problem in Heuristic Methods

Team members:

Ke Xia, Yiyuan Fu, Ruiyu Zhang, Jingxuan Chen

1. Motivation

The Traveling Salesperson Problem (TSP) probably represents the most intensive area of research within the wide range of combinatorial optimization problems: given a set of N points in the plane, the goal of a traveling salesperson is to visit all of them (and arrive back home) while keeping the total distance traveled as short as possible. An application of the TSP is in the drilling problem of printed circuit boards (PCBs). To connect a conductor on one layer with one on another, or to position the pins of integrated circuits, holes have to be drilled through the board. The holes may be of different sizes. To drill two holes of different diameters consecutively, the head of the machine has to move to a tool box and change the drilling equipment. This is quite time consuming. Thus it is clear that one has to choose some diameter, drill all holes of the same diameter, change the drill, drill the holes of the next diameter, etc. This drilling problem can be viewed as a series of TSPs, one for each hole diameter, where the 'cities' are the initial position and the set of all holes that can be drilled with one and the same drill. The 'distance' between two cities is given by the time it takes to move the drilling head from one position to the other. The aim is to minimize the travel time for the machine head.

In principle, one can enumerate all possible tours to get the best answer, but in practice the number of tours is so staggeringly large (roughly N factorial) that this approach becomes useless. However, many methods have been studied that seem to work well in practice, even though they are not guaranteed to produce the best possible tour. Such methods are called *heuristics*.

We choose four heuristic algorithms (nearest neighbor heuristic, smallest increase heuristic, Tabu-search and genetic algorithms). The analysis part will focus on: 1) compare the path length of these algorithms; 2) estimate the running time of our programs and do time complexities analysis.

2. Algorithms

2.1 Nearest Neighbor Heuristic Algorithm

The key to this algorithm is to always visit the nearest node. The method of nearest neighbor heuristic algorithm is to Read in the next point, and add it to the current tour after the point to which it is closest. (If there is more than one point to which it is closest, insert it after the first such point you discover). Comparing with non-heuristic method, perhaps the path is not the best, but the time complexity is $O(N^2)$ and much smaller than enumerate method.

2.2 Smallest Increase Heuristic Algorithm

The method of smallest increase heuristic algorithm is to read in the next point, and add it to the current tour after the point where it results in the least possible increase in the tour length. (If there is more than one point, insert it after the first such point you discover). Need to calculate every possible increase in order to know which is smallest. The time complexity is also $O(N^2)$.

2.3 Tabu-Search Algorithm

Different from all the neighborhood searching algorithms above, tabu search adopts a tabu list to keep illegal search to avoid running in circles while allowing moves with negative gain. Because it allows more possibilities, the result would be better than the algorithms mentioned above. The sacrifice though, is time consumption, tabu search shows a time complexity of $O(N^3)$, making it far slower than a typical 2-opt local search. This algorithm is chosen to form comparison with other algorithms and trigger a gain-loss weigh.

2.4 Genetic Algorithm (GA)

GAs work in a way similar to nature, and it may achieve a path that is most similar with the best path. A basic GA starts out with a randomly generated population of candidate solutions. Some (or all) candidates are then mated to produce offspring and some go through a mutating process. Each candidate has a fitness value telling us how good they are. By selecting the most fit candidates for mating and mutation the overall fitness of the population will increase.

3. Future Plan

We are going to analyze the performance of the algorithms introduced in TSP modeled for chip-design, compare time complexities with data visualized (tables and graphs). We decide to choose a N of 1000, which is the lower bound of node quantity for a typical LSI chip. C++ is chosen as the primary programming language to implement algorithms. We are to test algorithms on our laptops running either Windows or MacOS.