Traveling Salesman Problem

Shaobing Yuan

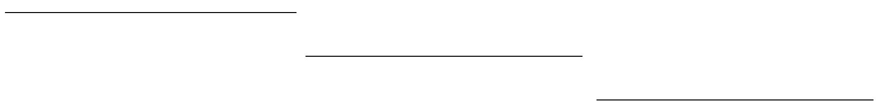
CHM 152 Start Date: 08/03/2024
Prof. Salvatore Torquato Report Date: 08/14/2024

Problem Statement

Briefly, for the traveling salesman problem, you randomly place N cites in a square box. The problem is the following:

Given N cities and the costs (distances) of traveling from any city to any other city, what is the cheapest (shortest) round-trip route that visits each city exactly once and then returns to the starting city?

The "energy" (objective function) E is simply the total distance traveled. You begin with some initial guess for a particular route. You then will swap two city routes. This will change the energy. If the energy change is negative, you always accept the move. If it's positive, you accept the move with a Boltzmann factor probability, i.e., $\exp(-E/T)$, where T is a fictitious temperature. You go through all of the cities this way. You need to begin with large T (meaning you will accept most moves independent of the energy change) and then you gradually lower the temperature after each equilibration at a particular temperature. This is the "cooling schedule." For simplicity, one might use the following cooling schedule:



where the vertical axis is T and the horizontal axis is time. The time interval before dropping in temperature is enough to "equilibrate" at that particular T. As T gets smaller, you will accept less and less uphill moves, until you achieve the "ground state" (the lowest energy). A criterion for determining when you are in the ground state is when the energy does not change after some number of iterations (which you can determine from experience). You should keep track of the lowest energy solutions for different initial conditions and then determine the frequency of the lowest energy configurations.

Begin with 5 cities to debug your program because in such an instance you can enumerate all possible solutions and determine the true ground state exactly. Once you are satisfied the program is working, try 25, 50, and 100 cities. Determine the fraction of times you achieve what you believe to be the global minimum starting from 100 different initial conditions for 5, 25, 50, and 100 cities. You should also plot the distribution of final energies as a function of N. Finally, plot your solutions for the minimal path for 5, 25, 50, and 100 cities.

Methods

The problem can be roughly divided into three major parts: generate the cities, find an initial guess, and optimize the initial route. The first two parts are easy; I set up the function generate_cities(N) to generate city coordinates based on numpy.random() and the function calculate_distances(cities_coordinates, N) to calculate the distance matrix, i.e. the distances between every two cities. Also, the function generate_initial_route(distances, N, i) finds an initial guess by keeping tracking the next nearest neighbour of a current city. Then, the third part can be further divided and conquered with the function swap_route_segment(route, N, i, j), should_swap(distances, route, N, i, j, T), calculate_initial_temperature(distances, N), current_temperature(initial_temperature, step, N), search_for_shortest_route(distances, N, initial_temperature, route, step), calculate_route_length(distances, route, N), and find_shortest_route(distances, N, initial_route); all written by simply following the instructions in the problem statement.

After that, the function try_all_initial_guesses(distances, N) tries all different initial guesses that generate_initial_route(distances, N, i) can get, find the shortest route and its length among all results, and calculate the accuracy rate as the fraction of times it shows up. There's no guarantee that what try_all_initial_guesses(distances, N) finds is the true global minimum, and consequently, the estimated accuracy rate is actually an upper bound; that seems to be the best we can do.

The function trend_of_accuracy_and_length(N_list, number_of_init) basically calculates the average shortest route length and accuracy rate with roughly a given number of different initial conditions, e.g. 100. For example, with the number of initial conditions being 100 and N being 5, it runs try_all_initial_guesses(distances, N) 20 times and calculates the average.

Finally, we have functions that plot figures. Nothing noteworthy.

Results

The distribution of final energies, i.e. the shortest route lengths, as well as the estimated accuracy rate of the algorithm are plotted in Fig. 1. As shown in the figure, the accuracy decreases rapidly with increasing city number N; estimated accuracy rates for $N = 5$, 25, 50, and 100 appear to be 100%, 54%, 10%, and 1%. On a second run, they go as 98%, 73%, 20%, and 1%. That is to say, a certain level of fluctuation remains at 100 initial conditions, yet that's good enough for us to see the trend. The algorithm here may be able to find the global minimum for up to $N = 50$, more or less, but not a chance for $N = 100$. The shortest route length grows with increasing N at a decreasing speed.

Shortest routes for 5, 25, 50, and 100 cities are plotted in Fig. 2.

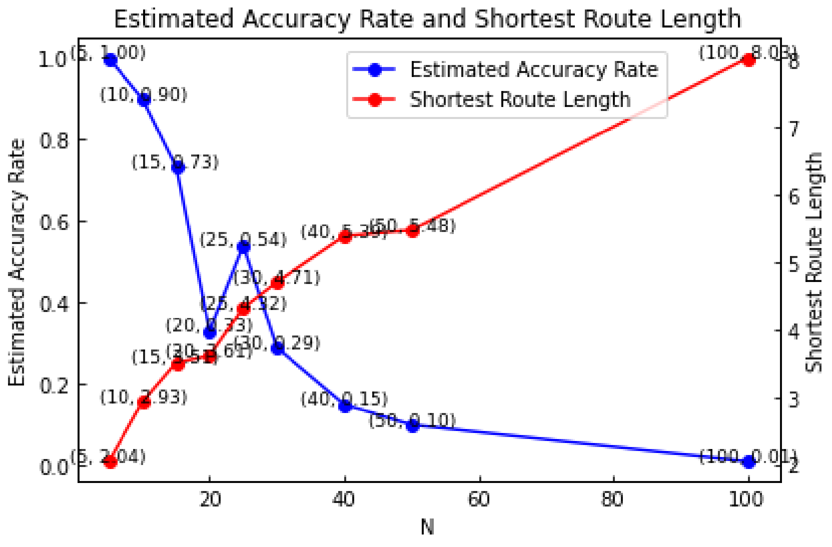


Figure 1: The estimated accuracy rates of the algorithm and the shortest route lengths (final energies) of different city number Ns.

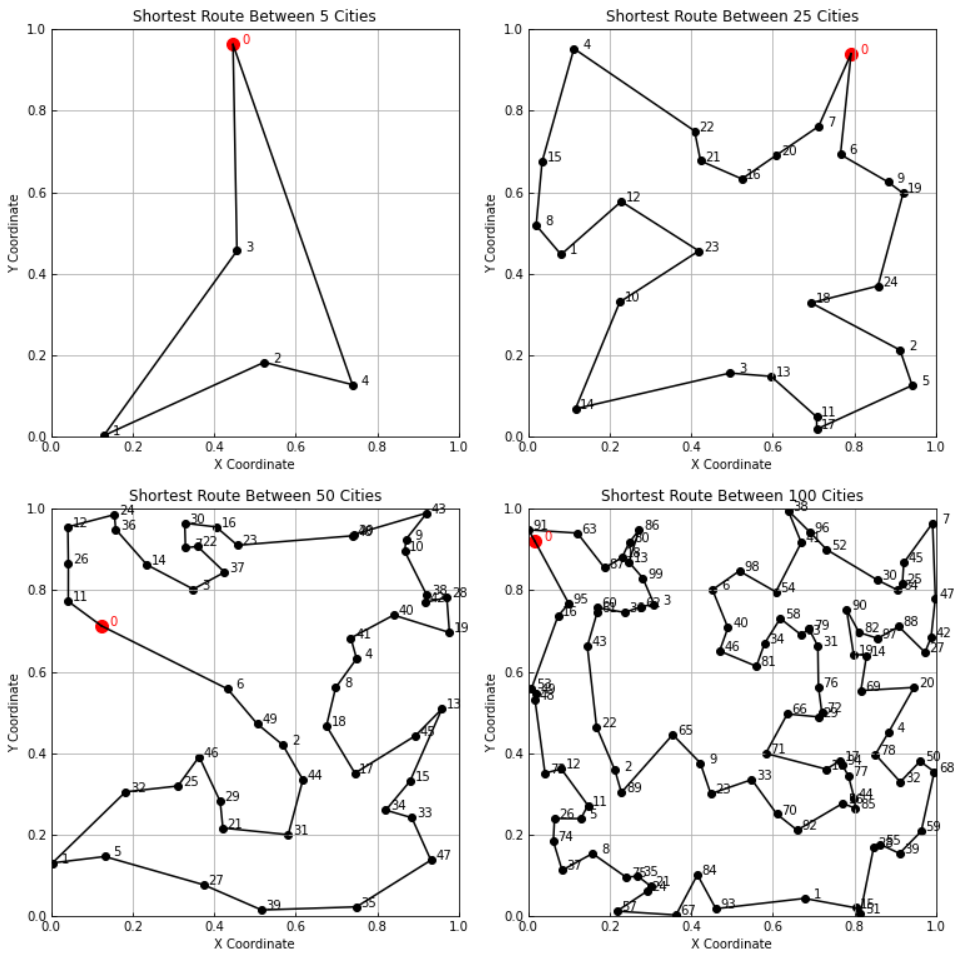


Figure 2: Shortest routes for 5, 25, 50, and 100 cities.