ROB 537
Learning Based Control
Fall 2025
HW #2: Local Search and Evolution
Due 10/21

Implement two search algorithms:

1- Simulated annealing or Epsilon-greedy
2- An evolutionary algorithm

You will use your implementations of algorithm 1 and 2 to solve the Traveling Salesperson Problem (TSP) for a 10, 25, and 100 city case. Download 10-city.csv, 25-city.csv, and 100-city.csv from Assignments/Homework 2 on canvas. The lines in a file correspond to cities on a map where the left bottom is (0,0).

For your report:

Part A: Describe your implementation of algorithm 1 and 2. Pseudo-code and hyperparameters will suffice. Compute the city-to-city distance matrix for each city map (to help save on computation). Then run the following experiments.

1) Algorithm 1 on 10, 25, and 100 city case
2) Algorithm 2 on 10, 25, and 100 city case

For statistical significance, run each algorithm and city map combination at least 10 times so you can provide statistical results. (Ex: Run algorithm 1 on the 10-city map at least 10 times.)

For each algorithm and city map combination, plot the best score at each iteration and the total number of solutions searched (as a function of iterations). Include the average and standard error in each plot. Then plot a "map" of the final path taken for each algorithm/city map.

Tip: Standard error is standard deviation divided by the square root of the number of trials. (Ex: If you run your algorithms 10 times, then you divide the standard deviation by $\sqrt{10}$ to get standard error.)

(Continue reading to page 2)

Part B: Answer the following discussion questions:

1) How many solutions are there to the 10, 25, and 100-city TSP?
2) What percentage of these solutions (on average) did algorithm 1 search? What about algorithm 2?
3) Why do these algorithms not find an optimal solution every time? (Hint: Why do we care about the percentage of solutions searched?)
4) Based on your results, is your algorithm 1 or 2 better for solving the TSP? (Hint: "Better" is purposely ambiguous here. It can mean more reliable, faster convergence, higher final scores, interpretable paths, etc. Tell us what "better" means to you and reference your results to tell us why 1 or 2 fits your description of "better".)
5) Based on your implementation and results, what are the tradeoffs between writing a more complex (algorithm 2) vs simpler algorithm (algorithm 1)?

Cite outside information you reference, tools you use, and people you work with. (Ex: Referenced Wikipedia, used github copilot w. Claude 4, got help from TA.)

Submit your report as either a PDF with your code attached as an appendix, or a zip file including a PDF of your report and your code in separate file(s).