IE801B Homework Assignment 3

1 Implementations and Strategies

Implement the 2-opt algorithm for solving TSP.

1. Compare the following two different implementations:
   (a) To calculate the tour length after creating a new tour.
   (b) To calculate the improvement directly without creating a new tour.
2. Using the implementation without creating a new tour, compare the following two strategies in each iteration:
   (i) Among all possible 2-opt swaps, choose the best one.
   (ii) Accept the first improving 2-opt swap.
3. Based on the 2-opt swap operations, develop a probabilistic search method with reasonably good performance. A *probabilistic* method means a method that can produce a different result for each run. The original 2-opt algorithms are deterministic; i.e., they will produce the exact same result for multiple runs. Your probabilistic search method does not need to be a complicated one. It can be an existing method such as simulated annealing, or any other small "probabilistic" ideas combined with 2-opt.

For 2-(i), if you are using Python or Julia, visualize how tours are changing and save the images as a GIF animation file to show progress; `matplotlib` and `Plots.jl` can save GIF animations. If you are using Java or C++, the visualization part is optional.

2 Evaluation

When you evaluate each implementation or strategy, generate random instances as follows:

- Use $[0, 1] \times [0, 1]$ square space to generate random x-y coordinates and use the Euclidean distance to create distance matrices. Generate 100 to 1,000 instances for each run and report the average tour length obtained.
- Use the Concorde solver to obtain exact optimal solutions to report the optimality gap of each implementation or strategy. Recall that the Concorde solver can only accept integer values for coordinates and distance matrices. You may multiply the coordinates or the distance matrices by 1,000 and round the numbers, then divide the tour length obtained by Concorde by 1,000.
- Observe performances for various numbers of cities, say $N = 20, 50, 100$. To see the impact of two different implementations, you may consider $N = 1,000$ or more.

3 Submission

Submit the following files for this assignment:

1. A PDF report that summarizes your code, experiments, and findings. LaTeX is recommended but not required. Using a Jupyter notebook is fine. Describe your experimental settings: CPU, RAM, OS, language version, package version, computational time, etc. In most cases, this is the only file that I will read. I will read your source code if necessary.

2. Describe how you used AI tools. This is not for grading but for my own education on how students are using AI tools.

3. GIF animation.

4. Your code files.

   - Do NOT submit your algorithm code as a Jupyter Notebook. You can use Jupyter while developing your code but not in the submission. (But for submission your Jupyter needs to be converted to PDF.)
   - You can write your main code as `main.py` and import it to your Jupyter notebook to create the final report is fine.
   - In your report, specify which source file is the file that I need to run to reproduce the results. If you choose to use C/C++/Java, describe how I can compile and run the code. For C/C++, `cmake` is recommended.

5. Please upload each file (PDF and source codes) separately without zipping unless you have too many separate files or use special directory structures.

6. If you prefer to submit your code via GitHub repo, that is okay too. You need to mention the repo URL and the specific commit SHA. Please make sure that I have access to the repo. My GitHub account name is `chkwon`.