Programming Assignment 1
– Solving TSP by Searching

CMSC 421 Summer 2024
Due Date: 11:59 PM July 31st 2024

Welcome to your first programming assignment. In this assignment you are going to explore searching algorithms on the Traveling Salesman Problem (TSP). The goal of this assignment is for you to explore various mathematical and algorithmic trade offs. So you are encouraged to collaborate, discuss and compare results with your classmates. You may work in groups of three at most. The final code and lab report submitted should include the name of all the team members.

Stuff you will need to do:

1. Familiarize you you see fit. Feel free to adapt/c rces with proper citation. Hints aged to share on Piazza.
2. Represent a gr
3. Read an $N * N$ h with up to N^2 edges. Your c cy matrix in the following forma Figure 1. Take the TSP proble infile matrix. As graphs will be e full matrix will be provided alt
4. Write your out me. For example, for the hill-clim . The file should contain 4 entrie s expanded, CPU run time, and
5. Perform basic tc.
6. Use your outpu y using a GUI or other means.
7. Perform experiments and conduct some empirical experimentation to explore various mathematical and algorithmic trade offs.

Grading rubrics:

- Presentation of the results in graphical form. Several experiments are defined for each of which there is a capstone plot or chart showing tradeoffs in solution quality and accuracy and measures of time/space used to reach that solution (submitted via GRADESCOPE). Are these graphs illustrative of the phenomena expected?

```
n     /* integer # of rows/columns , all matrices will be square */
M11, M12, ..., M1n  /* each row of the matrix separated by a new line */
M21, M22, ..., M2n   /* you can assume these are integers */
...
Mn1, Mn2, ... ,Mnn
/* end of file */
```

Figure 1: Format of .txt

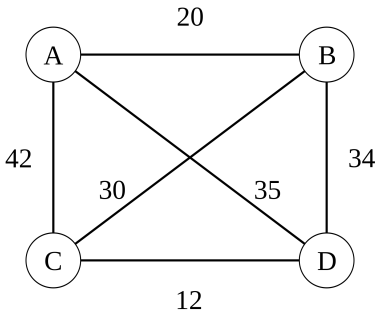


Figure 2: TSP Graph

```
4
0, 20, 42, 35
20, 0, 30, 34
42, 30, 0, 12
35, 34, 12, 0
```

Figure 3: Example of `infile.txt`

- Discussion of the experiments. For each of the experiments a short narrative explaining the phenomena observed will be submitted (via GRADESCOPE). Does this narrative accurately describe the mathematical and algorithmic phenomena behind the observations?

Part 1: Expl

Implement A^* on a r tics:

1. $h(n) = 0$. This
2. Grab edges ra nodes, start with grabbing n edg
3. Cheapest rema

For the experiment:

1. Randomly crea 5, 20, ...
2. Run the above
3. For each famil N/MAX of total cost, number o
4. Use those data and the other for CPU and real- hs/matrices (size 5, 10, 15, ...). nodes, and CPU and real-world etc. to represent each searching

Compare and discuss the results in a short paragraph. Questions you can explore are for example:

- Which algorithm provides solution with the lowest cost? What's the difference of their best solutions, and how that changes when the size of the graph increases?
- Which algorithm has the least runtime and how do their runtimes change with the size of the graph increases?
- Is there difference between GPU and real-world runtime?

What to submit:

Code: Your implementation of three functions: `A_uniformCost()`. `A_randomEdge()` and `A_cheapestEdge()`. Each function should first read in the graph/matrix from `infile.txt`. Then perform the algorithm and finally return the cost of the best solution.

Report: Two graph plots of your experiment results and your discussion.

Part 2: Explore Local Search Algorithm

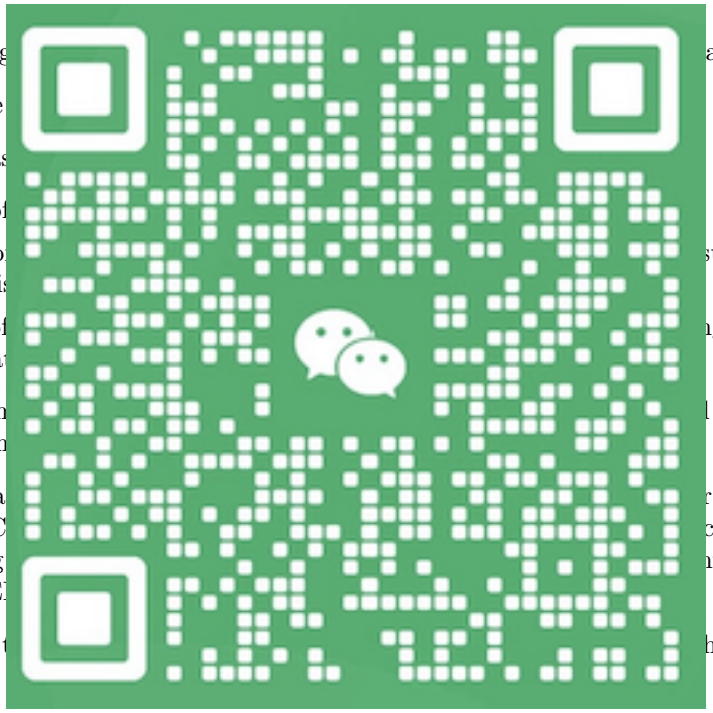
In this exercise, we explore the use of local search methods to approach TSPs.

1. Implement and test a hill-climbing method to approach TSPs.
2. Implement and test a simulated annealing method to approach TSPs.
3. Implement and test a genetic algorithm method to approach TSPs. (see section 4.3 of Larranaga et al. 1999)

Compare these results with each other and with the optimal solutions obtained from the A^* algorithm with the MST heuristic. Compare not just the quality of the results but the time it takes to obtain them. Give an assessment of the relationship between computation time and solution quality, i.e., if all I wanted was a 75% solution, which method is quickest? How about a 90% solution? Is there are general rule here?

For the experiment:

1. Randomly create a family of 30 TSP graphs/matrices for each size 30 (or you can experiment with the size).
2. Run the A^* alg al solutions.
3. Run the above
4. Try different as
   - number of
   - number o sume the cooling function is
   - number of gth of crossover, and muta
5. Fix your paran l cost, number of nodes, CPU an
6. Use those data each graph, the x-axis is the C cost between the local searching t shape/color to represent AVE
7. Maybe repeat t hs.



What to submit:
Code: Your implementation of three functions: `hillClimbing()`, `simuAnnealing()` and `genetic()`.
Report: At least three graph plots of your experiment results and a short paragraph of your discussion.

1 Extra Credit

Use your implementation from Part 2 to solve the Knapsack problem. See the 0-1 knapsack problem definition.

2 Submission Instructions

Submit a single zip file to Gradescope by 11:59 PM on July 31st. This should include your implementations of the following functions:

- A_uniformCost()
- A_randomEdge()

- A_cheapestEdge()
- hillClimbing()
- simuAnnealing()
- genetic()

It should also include any additional code necessary for running these functions.