Spring 2023

AIX30006 Project – Genetic Algorithm

Due date

Monday, April 3rd, 2023.

Please note that no late submissions will be accepted.

Project description

In this team project, your team will design one of the well-known meta-heuristic algorithms, namely the genetic algorithm, to perform an optimization analysis. The objective of this project is to exercise the implementation of the genetic algorithm to tackle two tasks: 1) optimizing a continuous function using the algorithm and 2) solving the Traveling Salesman Problem (TSP) using the algorithm. The simulation code should be written in Python. The main deliverables for this project are 1) a project report documenting your team's efforts on this project, 2) a presentation file outlining the storylines (15 minutes long) used to nail down this project, and 3) simulation codes your team developed. All deliverables must be a joint effort, meaning that your team will need to distribute the different tasks amongst the members to share the workload fairly and effectively. However, please note that one student from your team should submit materials on behalf of the entire team.

Team

Students will work in a team of two or three students with the aim of developing teamwork skills while working with other students. The entire team will receive the same base grade; however, I will ask each of you to submit your own peer assessment in which you evaluate the contributions of your teammates; therefore, the grade may be changed as needed. The purpose of this assessment is to find ways to work well together and contribute equally to the overall product.

Data

Students will be required to use real-world operational datasets to complete the task 2. The datasets can be downloaded from the "Project" folder in the LMS system. It is important to note that the task 1 does not require the use of datasets to address the problem.

Citation

Please ensure that all codes and materials originating from other sources including ChatGPT must be clearly documented.

Task 1: Optimize a continuous function using the genetic algorithm

Unconstrained optimization

Suppose that you are asked to find the minimum point of the objective function:

$$f(x) = 2x^4 - 10x^3 + 8x^2 + 5x$$

Deliverables for this task are as follows:

- Create a plot showing the objective function within $-5 \leq x \leq 5$.
- Discuss how you end up choosing user-defined parameters (e.g., crossover rate).
- Explain how you design a stopping criterion and provide evidence of whether your optimum is converged or not.
- Report the optimal (unconstrained) design variable and solution.

Constrained optimization

Suppose that the objective function is subject to the constraint:

$$x \leq 2$$

Deliverables for this task are as follows:

- Create a plot showing both objective function and constraint within $-5 \leq x \leq 5$.
- Discuss how you consider the constraint in the process of the genetic algorithm.
- Report the optimal (constrained) design variable and solution.
- Compare the constrained and unconstrained optimal points and explain why they are different.

Please note:

It is possible to solve this task with various open source toolboxes or libraries. This is not the point. You are asked to write your own genetic algorithm code based on what you have learned in the class.

Task 2: Address the Traveling Salesman Problem (TSP) using the genetic algorithm

Problem statement

Imagine that you are currently working for the United Parcel Service (UPS) as an operations research analyst. Your job is to provide the shortest route, which visits each delivery point exactly once from the origin point, to the UPS truck drivers. Each driver is responsible for one city and you are asked to manage seven drivers for planning seven different cities (i.e., Atlanta, Boston, Cincinnati, Denver, New York, Philadelphia, and San Francisco).

Please note:

Please note that 1) all edge costs (i.e., great circle distance) are symmetric and need to be calculated by the Haversine formula, 2) you are only asked to choose three different cities for this task, meaning that you don't have to analyze all of the cities, and 3) you can feel free to use any open source toolboxes or libraries to tackle this task. As a final note, for your information, a near-optimal solution for the Atlanta city is approximately 180,000 meters. Please use this reference for a verification/validation purpose.

Deliverables for this task

- Generate a plot showing the delivery locations for the selected cities.
- Construct a distance adjacency matrix that represents great circle distances (you may need to consider using the Haversine formula to compute the distances) between cities i and j such that A_{ij} represents the distance between those two cities. The size of the matrix should be the number of cities by the number of cities (e.g., 20 by 20 matrix for 20 delivery locations).
- Find a solution using the greedy algorithm. Here, "greedy" means that it is designed to find a solution based on "nearest neighbor selection" from a starting point. You need to provide the shortest route generated by the greedy algorithm and a plot visualizing the optimized route.
- Find a solution using the genetic algorithm. You need to provide the shortest route generated by the algorithm, the optimal value, and a plot visualizing the optimized route.
- Explain the reason why the solution generated from the greedy algorithm looks different from the solution generated by the genetic algorithm.
- Create a plot of convergence history showing the optimum value generated by the genetic algorithm is converged as the number of iterations increases.
- Discuss with your teammates and answer the following question:
  - Do you think that the solution from the genetic algorithm is a global optimum or not?
- Discuss with your teammates and answer the following question:
  - Do you think that you will always get the same solution (e.g., the total travel distance is always identical) when you execute the algorithm multiple times?
- Discuss with your teammates and answer the following question:

- Suppose that your algorithm is performing slowly for some reasons. From the perspective of designing the algorithm, how would you be able to make it more computationally efficient? As this is an open question, I would like to specifically ask you to answer the question by mentioning the implementation of the aforementioned greedy algorithm.