Background

The travelling salesman problem (TSP) is a classic optimization problem.[1] Given a number of cities with associated distances, the goal is to find the shortest route that visits each city once. There are a number of excellent algorithms to solve this problem, here we consider a simple heuristic based on evolutionary algorithms.

Task

We provide a number of files, each one specifying the locations of a collection of "cities" as (x, y) tuples. Each line describes the location of one city. For simplicity, we consider cities arranged on a two-dimensional grid and restrict movement to horizontal and vertical, i.e., distances should be measured in Manhattan distance.[2]

Your program should parse a file containing a variable number of cities, and use a simple[3] $1 + 1$ evolution strategy[4] to find an approximately shortest path. This path should visit each city once, always starting from the city defined on the first line of the file. Since we consider a strategy with only one parent, skip the recombination/crossover step and only use (one or more) mutation operators. Visualize the fitness of the best solution in each generation. Additionally, visualize the location of cities overlaid with your best solution.

Implement your solution in Python. You can use any built-in modules, but you cannot use any code from external sources, e.g., libraries. One exception is `matplotlib` which you may use for visualization. Please use Git for version control and create repository on GitHub to share your solution with us.

Further instructions

- For your own sake allocate at most 4h for this task.
- Due to the limited time focus on writing clean code that works before you worry about optimizations.
- Keep in mind that you are implementing a *heuristic*. Do not expect it to discover optimal solutions for sufficiently large/complex problems.
- Be ready to explain your code in detail, including several design decisions you will need to make along the way. In this context, we may also discuss algorithmic limitations/improvements and opportunities for optimizing the implementation, e.g., to reduce runtime.
- It is OK to use AI tools, but do not forget the previous point.

[1] https://en.wikipedia.org/wiki/Travelling_salesman_problem

[2] https://en.wikipedia.org/wiki/Taxicab_geometry

[3] No need to include advanced features like self adaptation or support optimization problems other than TSP.

[4] http://www.scholarpedia.org/article/Evolution_strategies#Example:_A_Simple_.5C.28.28.5Cmu.2B.5Clambda.29.5C.29-ES_for_Combinatorial_Ordering_Problems