FUNSEARCH ON TRAVELLING SALESMAN PROBLEM
CS5491 AI PROJECT MILESTONE

1 Group Members

MACheng 59046021
SuZehao 58818678
ZhouJiayu 58909853

2 Problem Description and Motivation of the Approach

The Traveling Salesman Problem (TSP) is a well-known NP-hard combinatorial optimization problem that requires searching for the lowest cost to complete a given set of visits to a set of city targets and return to the starting point. The complexity of the routes grows exponentially as the number of cities increases, making computationally exact solutions infeasible for larger problem instances.

Traditional heuristic and meta-heuristic approaches, such as Christofides algorithm, nearest neighbor heuristics, and genetic algorithms, have been widely explored in the present. However, while the problem of access has been solved so far, there is still no search to find a more efficient and generalized solution. In this project, FunSearch, an evolutionary search framework driven by Large Language Models (LLMs), is used to enable the automatic generation, evaluation, and refinement of optimization algorithms for solving instances of the TSP problem based on the exploration of new algorithmic constructs that may be overlooked by human-designed heuristics.

3 Design of the Approach

The approach combines FunSearch with an evaluation-driven optimization framework for solving TSPs, aiming at a more efficient route planning problem.

3.1 FunSearch framework

FunSearch consists of two main components.

LLM (Code Generator): Generates multiple feasible candidate algorithms for solving the TSP.

Evaluator (performance filter): Tests and evaluates the generated algorithms, filtering out low performing solutions.

3.2 Implementation details

Data processing: parse the contents of the dataset, extract city coordinates, `parse_display_coords(filepath)` function reads and stores TSP city coordinates for further processing.

Algorithm execution and evaluation: Sandbox environment ensures safe execution of the algorithms generated by LLM.

3.3 Implemented Core Functions

solve(): skeleton function populated by LLM to generate TSP solutions.
evaluate(): evaluates the quality of the solution generated by LLM.
priority(): defines heuristics to guide FunSearch to evolve to a better solution.
Visualization: Use matplotlib to plot city coordinates and verify the integrity of the input data.
Benchmarking: Benchmark performance against traditional TSP solvers.

4 Preliminary results

4.1 Partially Implemented

LLM-Based Algorithm Generation: After data processing was complete, initial FunSearch runs produced valid Python implementations of the heuristic solver, and sandbox execution verified the safety and controllability of the code execution.

Evaluation and Optimization: Basic performance filtering was implemented using evaluate(), and early FunSearch-generated heuristic solvers showed potential but required further refinement to compete with mature algorithms.

4.2 Next Steps

Improve the LLM-generated algorithms using an iterative improvement strategy and implement evaluation metrics tailored to TSP.