Class 10 - Project Summary and Methodology

Guiding Genetic Algorithms with Language Models for Combinatorial Optimization

MISE Research Program - July 2025

Pedro Paredes

Summarizing the Project

Problems as Big Puzzles

A ***combinatorial problem*** can be viewed as an exceptionally large puzzle:

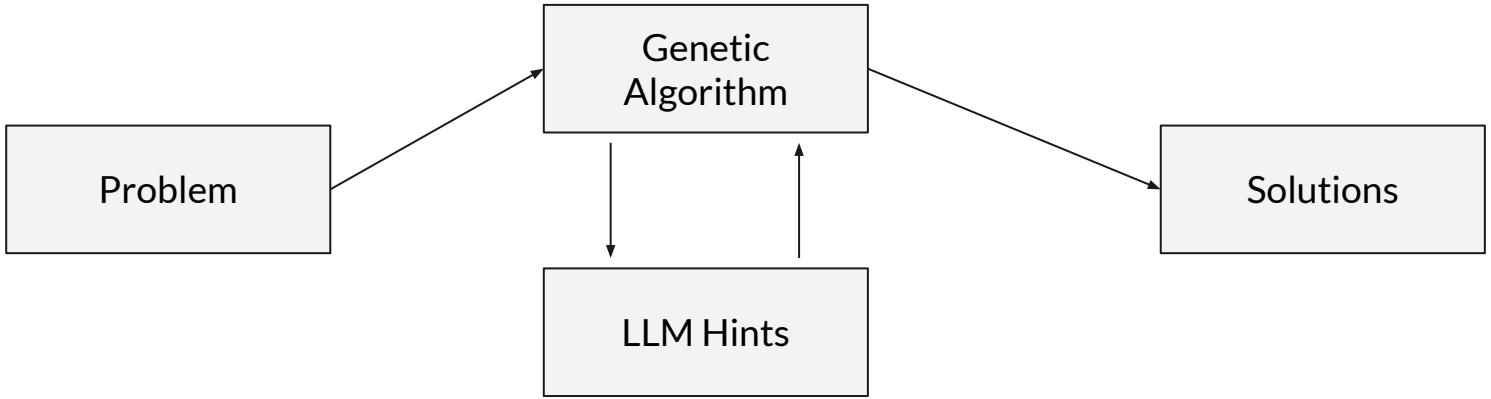
- We are given many discrete pieces and clear rules for how they may be arranged
- Our goal is to discover a valid arrangement that is best according to some measure of quality

Combinatorial problems appear everywhere: packing delivery trucks, arranging playlists, balancing sports teams, or routing thousands of packages through a warehouse, …

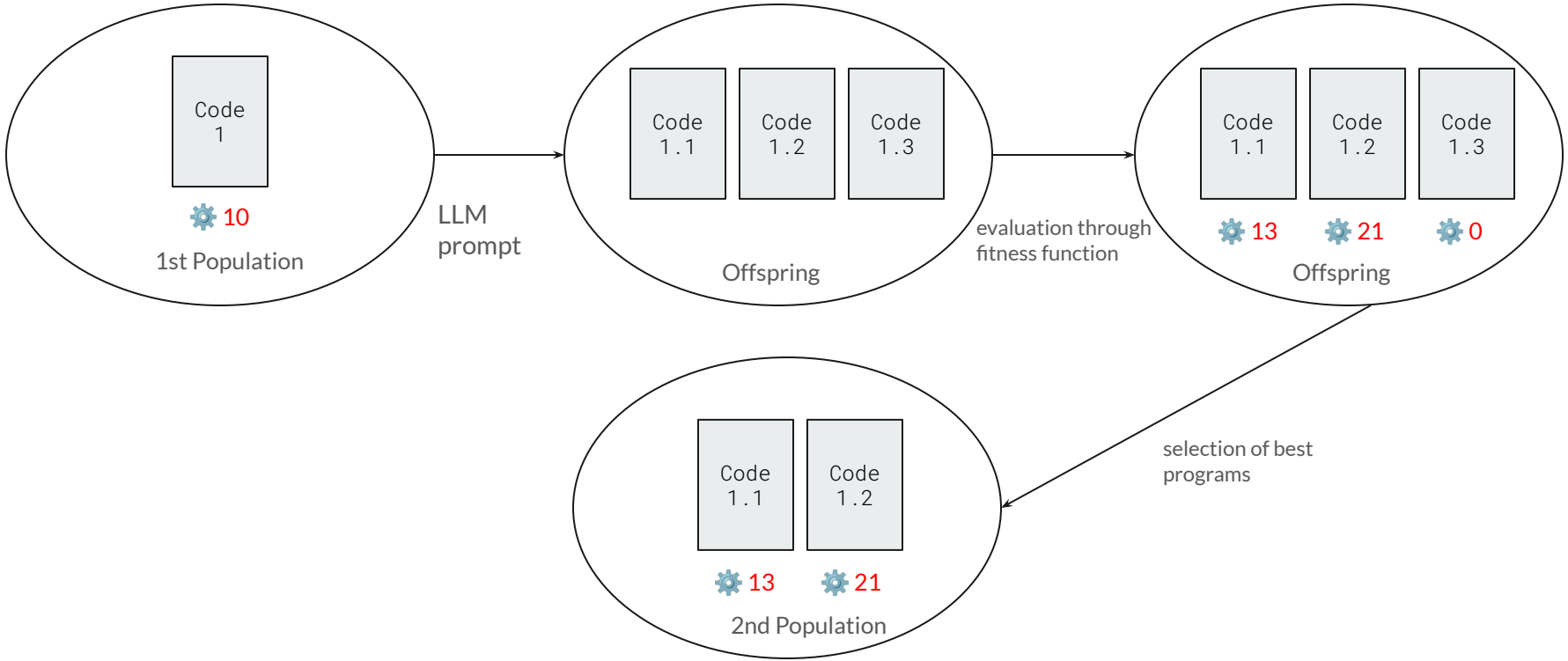
What Did We Build?

We will create an algorithm that ***searches*** through arrangements of pieces to solve these “puzzles”

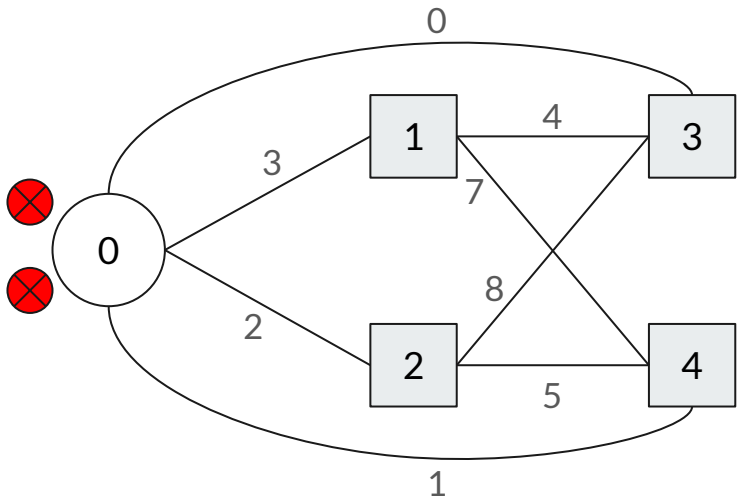
Our algorithm will use large language models to get advice on where to search next



Visual Representation of Project



CARR Application



Methodology Review

Topics We Learned

- Exhaustive Searching
- Informed Search
- Heuristics
- Hill Climbing
- Genetic Algorithms

Exhaustive Search

Search by at every step you choosing one unexplored option, follow it all the way until you can't continue, then backtrack to the last branching point and pick the next option.

You always finish one complete branch before moving to its sibling branch

When is it useful?

- Generating or solving puzzles (Sudoku, mazes, etc)
- Enumerating every possible arrangement or path

Informed Search - Dijkstra's Algorithm

The simple exhaustive search solution is pretty inefficient since we repeatedly search the same paths and we search paths that are not very promising

We can make our search more efficient using ***informed search*** (also known as ***Dijkstra's Algorithm***):

- Instead of going depth-first, let's look at more promising paths first, i.e., shorter
- Instead of keeping track of visited cells, we will keep track of the shortest path to get to each of them. If we visit a cell and the current distance is greater than or equal to the previous best, ignore this path.

Heuristics

A ***heuristic*** is a non-optimal approximation to a problem

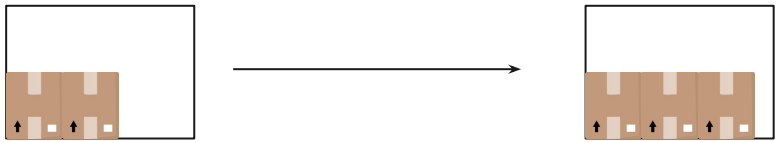
Often we use intuition about a problem to come up with heuristics

We can use heuristics in several ways:

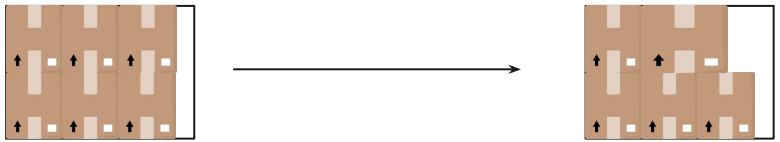
- To pick which states to explore first
- To discard non promising solutions

Search or Optimization

In **search algorithms** (e.g. Exhaustive Search, Informed Search, A*) states are partial solutions that we gradually build up into complete solutions



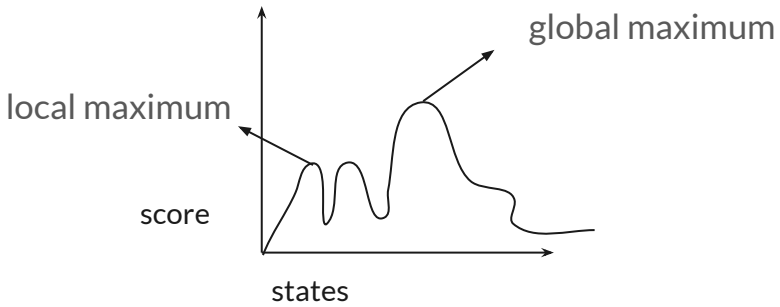
In **optimization algorithms** (e.g. Hill Climbing, Genetic Algorithms) states are full solutions that we move between



Finding Maximums in Functions

The ***state space*** of a problem is the set of all possible states, e.g., for N-Queens are boards with one queen per column

The problems we have been thinking about can be modeled as finding the **maximum of a score function** over a state space



Hill Climbing

Hill climbing is a method to find a solution to a problem by always moving in the direction of a “better” solution

- Start from a random state s
- Move to a neighbor with better score (break ties randomly)
- If we get stuck and didn’t find a solution, restart

A neighbor of a state is a state obtained by a local change: e.g., moving one queen in the n-queens example

Genetic Algorithm

A ***Genetic Algorithm (GA)*** is a method inspired by the evolutionary process of “survival of the fittest”

1. Generate **P** random states -> call them your population
2. Repeat until optimal found: (***evolution step***)
    a. Pick **C** pairs of states and randomly mix them (***crossover***)
    b. For each new state, with some small probability **p** apply a local change (***mutation***)
    c. Discard all but the **P** states with best score (***selection***)

What's next?

Next class Friday (3 pm) - last class

Assignment sheet on website soon

Class 11: ??