Artificial Intelligence
-
Delivery Scheduling

•••

João Mendes Silva Belchior - up202108777
José Francisco Reis Pedreiras Neves Veiga- up202108753
Pedro Vidal Marcelino - up202108754

Problem Definition

This project’s objective is to optimize the delivery of packages from a starting point (0,0) to various destinations, optimizing the travel costs while adhering to each package type's special needs.

There are 3 different types of packages:

- **Fragile packages** have a chance of getting damaged by km traveled.
- **Urgent packages** have a penalty associated with delays.
- **Normal packages** have a cost by km traveled (like the other 2).

A good solution should:

- **Maintain a high reputation:** avoid breaking fragile packages and deliver urgent ones on time.
- **Reduce the total cost**.

Related Work

There are several optimization solving and scheduling APIs that can be found online, such as:

- Hexaly is the world's fastest optimization solver for Routing, Scheduling, Packing and more.)
- Mapbox Optimization API returns a duration-optimized route between provided input coordinates.
- Solvice powers up scheduling and planning tools with AI, through a unique fusion of Heuristics, Optimization and Machine Learning.
- Urbantz is designed to optimise, control, track and scale large-volume delivery operations.

Optimization Problem

Solution - Defined as an ordered list of packages, from the first to the last to be delivered.

Evaluation - A good solution should have a high reputation and a low total cost.

Constraints - Fixed costs and speed; only 1 vehicle is used.

Important Functions:

- **Local Search Heuristics**(Hill Climbing with Best/First/Random Neighbor Accept)
- **Metaheuristics** (Simulated Annealing, Tabu Search, ILS)
- **Genetic Algorithms** (Biological algorithms, such as Selection, Mutation, Crossover)

Implementation

Technologies: Python, NumPy, pandas

- We used the code template in the description, which contains the Package class implementation.
- Main useful functions have been implemented, such as calculating distances, probabilities for breaking fragile packages and total cost and reputation of a solution.
- Fully implemented local search heuristics, metaheuristics and genetic algorithms.
- Developed various auxiliary functions for each heuristic.

Approach

Evaluation Function:

- Travelled distance cost
- Reputation Cost
- Ratio (“reputation_weight”) to minimize both

Metaheuristics:

- Iterative Local Search (Hill Climbing based)
- Simulated Annealing
- Tabu Search
- Genetic Algorithms

Auxiliary functions:

- Neighbor Functions (First Accept, Best Accept, Random)
- Mutation Functions (Swap, Inversion, Scramble)
- Selection Functions (Tournament, Roulette-wheel, Random)

Metaheuristics

ILS with Hill Climbing

- Start with a random solution
- Chooses a neighbor, finds Local Optimum
- Repeat until finding the best Local Optimum (closer to Global Optimum)

Simulated Annealing

- Similar to hill climbing
- Dependant on a decreasing variable (temperature), may or may not accept worse solutions

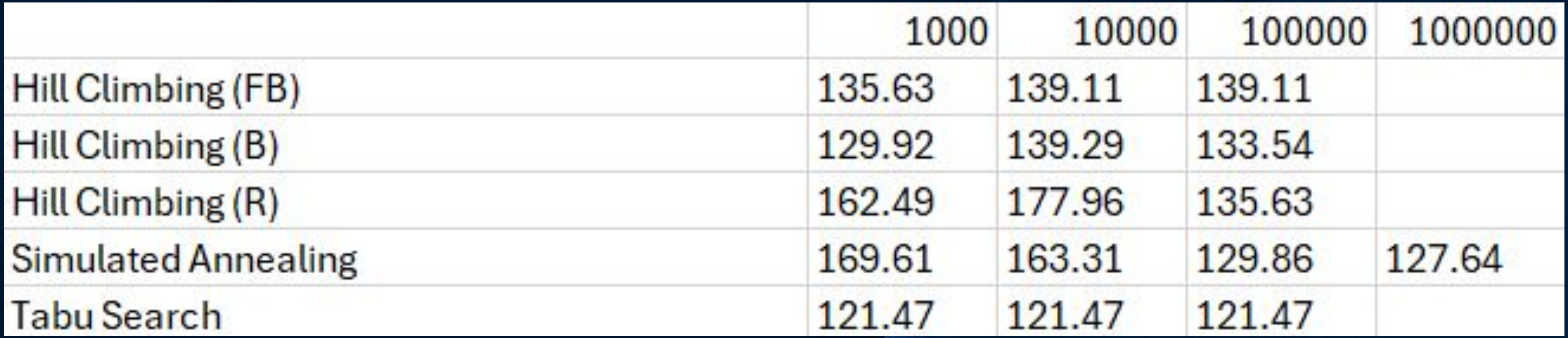
Genetic Algorithm

- Start with a random set of solutions
- Select parent algorithms to create next generation
- Keep record of best solutions so far

Tabu Search

- Keeps record of visited solutions
- Best Accept in a Neighborhood

Results



	1000	10000	100000	1000000
Hill Climbing (FB)	135.63	139.11	139.11	
Hill Climbing (B)	129.92	139.29	133.54	
Hill Climbing (R)	162.49	177.96	135.63	
Simulated Annealing	169.61	163.31	129.86	127.64
Tabu Search	121.47	121.47	121.47	

Conclusion

We implemented and explored several optimization heuristics studied in class. Each one offers a unique approach for finding the best possible solution. By comparing results, we understood their strengths, weaknesses and applicabilities.

Hill Climbing

- Simple to implement
- Needs ILS to overcome Local Optimum

Simulated Annealing

- Similar to Hill Climbing
- Needs more iterations

Genetic Algorithms

- Reaches high diversity of solutions
- Hard to tune (generations, mutation rate)

Tabu Search

- Probably the best (reaches good solutions with less iterations)
- Tabu list size not so important

References/ materials used

- Slides present in Moodle
- Several Computerphile videos