





















Inteligência Artificial

Eduardo Oliveira - 202108690
João Francisco Alves - 202006281
José Miguel Isidro -202006485

One Pizza - Optimization Problem

- Opening a small pizzeria offering only one type of pizza.
- Need to determine ingredients to include based on client preferences.
- Each client has likes and dislikes for ingredients.
- Clients will visit if all liked ingredients are on the pizza and none of the disliked ingredients are.
- Objective: Maximize the number of clients visiting the pizzeria.
- Scoring: One point for each client satisfied with the pizza ingredients.

What do you like on pizza?	What do you like on pizza?	What do you like on pizza?
 <input checked="" type="checkbox"/>	 <input type="checkbox"/>	 <input type="checkbox"/>
 <input type="checkbox"/>	 <input checked="" type="checkbox"/>	 <input checked="" type="checkbox"/>
 <input type="checkbox"/>	 <input type="checkbox"/>	 <input checked="" type="checkbox"/>
 <input checked="" type="checkbox"/>	 <input type="checkbox"/>	 <input type="checkbox"/>
 <input type="checkbox"/>	 <input type="checkbox"/>	 <input checked="" type="checkbox"/>
 <input type="checkbox"/>	 <input checked="" type="checkbox"/>	 <input type="checkbox"/>

Related Work

- The optimization problem "One Pizza" was featured in a competition hosted by Google called "Practice Round - Hash Code 2022". The competition was won by the programmer who achieved the highest score.
- Theoretical Maximum Score: < **3900** (using the given test files)
- In our research, we found some source code with attempts to solve the problem:
 - Using Google OR-Tools, a competent was able to achieve a score of 3863;
 - A solution using the Hill Climbing algorithm achieved a score of 3895;

Formulation as a Search Problem

State Representation:

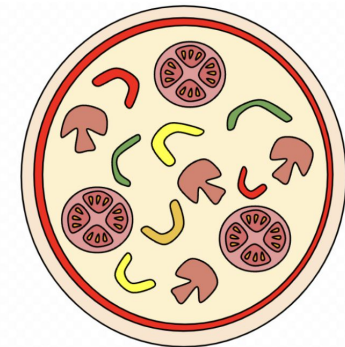
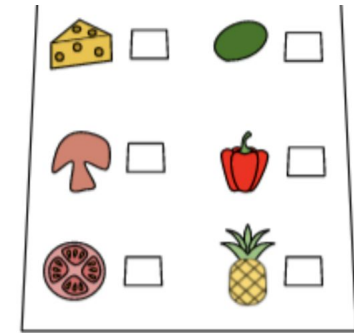
- The state in this search problem can be represented as a set of ingredients currently chosen for the pizza.

Initial State:

- The initial state is an empty set representing no ingredients chosen for the pizza yet.

Objective Test:

- If all the ingredients **liked** by a client are present and none of the ingredients **disliked** by that client are present in the set of chosen ingredients, then the objective is achieved.



Operators:

- AddIngredient**(ingredient)
- RemoveIngredient**(ingredient)

Effects:

- Add the specified ingredient to the set of chosen ingredients.

Preconditions:

- The ingredient to be added must not already be present in the set of chosen ingredients.

Costs:

- In this problem, the cost of adding an ingredient can be considered constant, for example, 1.

Work Already Carried Out

Programming Language:

- Utilizing Python for its simplicity.

Development Environment:

- Using Visual Studio Code.

Key Components:

- Objective Function: Determines satisfaction level of clients based on pizza ingredients.
- Operators: Add and remove ingredient operations to change pizza state.

Interface:

- Implemented CLI to select and algorithm and test file.

Data Structures:

- PizzaState: class for representing the state of a pizza with ingredients.
- Client: class that contains client's ingredient preferences (likes and dislikes).
- Set to store all ingredients mentioned by clients

Algorithms & Heuristics

Hill Climbing:

- Local Search: Iteratively explores neighboring solutions by making **small modifications** to the current solution.
- Greedy Selection (Best accept): Selects the neighboring solution that **maximally improves** the objective function value at each step, moving towards the peak (or valley) of the local optimum.

Simulated Annealing:

- Exploration and Exploitation: Simulated Annealing balances exploration and exploitation by accepting moves that lead to worse solutions with a certain probability, allowing the algorithm to **escape local optima** and explore diverse regions of the solution space.
- Temperature Schedule: It uses a temperature schedule to **control the acceptance probability** of worse solutions. Initially, the temperature is high, allowing for more exploration, and gradually decreases over time to focus on exploitation.
- Boltzmann Distribution: It uses the Boltzmann distribution to **probabilistically accept worse solutions** based on the temperature and the difference in objective function values.

Algorithms & Heuristics

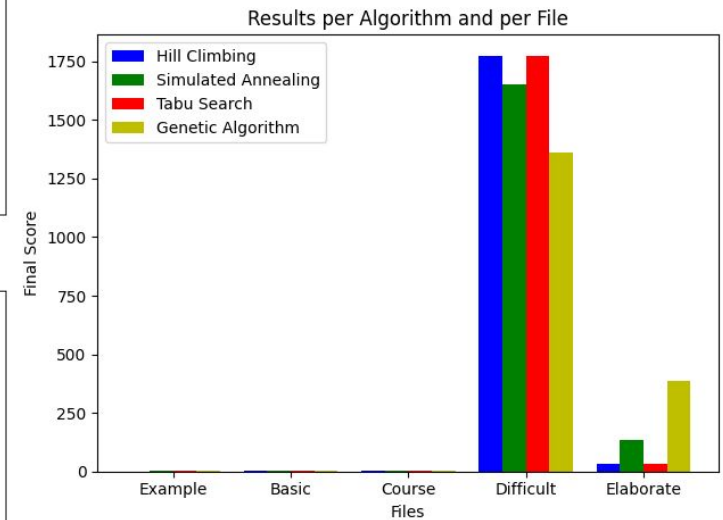
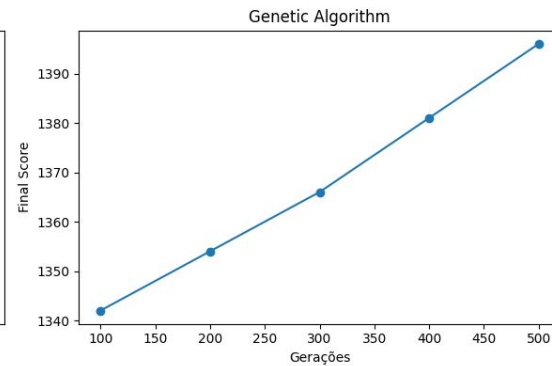
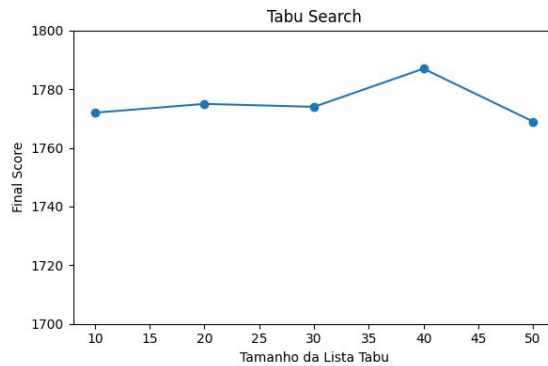
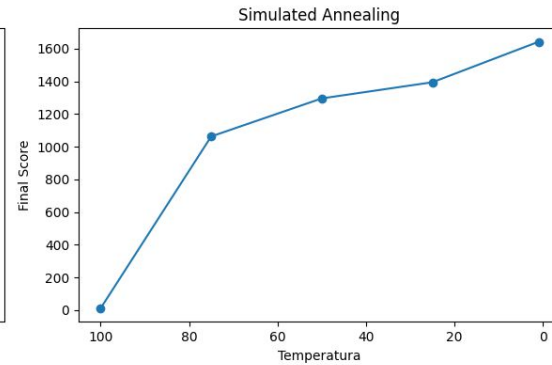
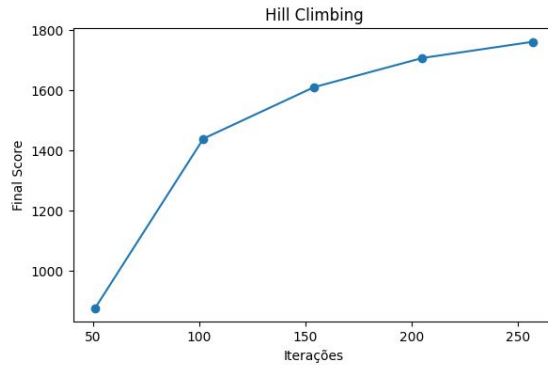
Genetic:

- Population-based Search: Genetic Algorithm maintains a **population** of candidate solutions and evolves them over generations through selection, crossover, and mutation operations.
- Selection Pressure: It uses selection mechanisms, such as **roulette wheel selection** or **tournament selection**, to bias the selection of solutions towards higher fitness values, promoting the propagation of promising solutions.
- Crossover and Mutation: It employs crossover and mutation operators to **generate diverse offspring solutions**, facilitating exploration of the solution space and introducing new genetic material into the population.
- Survival of the Fittest: GA often incorporates elitism, preserving the **best solutions** from one generation to the next, ensuring that promising solutions are not lost during the evolution process.

Tabu Search:

- Memory-based Search: Tabu search maintains a tabu list (*tenure*) to keep track of **previously visited solutions** and restricts the search from revisiting them in subsequent iterations.
- Exploration and Exploitation: It balances between exploration and exploitation by allowing **exploration of non-tabu moves** while **avoiding cycling through previously visited solutions**.
- Aspiration Criteria: It uses aspiration criteria to **override the tabu status** of moves that lead to superior solutions, allowing promising moves to be explored even if they are on the tabu list.

Experimental Results



Comparison of Algorithms Final Scores on all Data Files

Comparison of Algorithm Evolution on The Difficult Data File

Adicional Notes:

- In **Example**, **Basic** and **Course**, the maximum results (2, 5, 5, respectively), were obtained in all algorithms, except for Hill Climbing.
- In **Elaborate**, results were limited to a running time of 10 minutes.

Conclusion

In this project, we explored various algorithms and meta-heuristics to tackle the optimization problem presented by **One Pizza**. Our objective was to find optimal solutions that maximize customer satisfaction within the constraints of a single pizza's ingredients.

Through the implementation of different algorithms, we gained valuable insights into the intricacies of optimization and the diverse approaches available to address such problems. Each algorithm pursued the goal of maximizing customer satisfaction with the given ingredients, resulting in a range of outcomes.

Notably, we observed significant variations in the results produced by each algorithm, including differences in efficiency and the quality of solutions generated. These disparities provided valuable insights into the strengths and weaknesses of each approach.

In conclusion, this project has been an enlightening journey, offering valuable lessons and practical experience in algorithmic optimization. Despite the challenges encountered, we view our efforts as a success, with our findings contributing to a deeper understanding of optimization techniques and their applications in real-world scenarios.

References

- Solution obtained using Google OR-Tools
- Google OR-Tools
- Near Perfect Solution Achieved with Hill-Climbing in C#
- Work Assignment and Input Files
- CustomTkinter by Tom Schimansky
- Artificial Intelligence: A modern Approach by Stuart Russel and Peter Norvig