André Gonçalves Pinto
Luis Vieira Relvas
Rodrigo Moisés Baptista Ribeiro

Group_A1_48

Book Scanning

Introduction

- Given a description of libraries and books available, plan which books to scan from which library to maximize the total score of all scanned books, taking into account that each library needs to be signed up before it can ship books.
- Python
- Hill Climbing Best Neighbor/ First Accept
- Simulated Annealing Random
- Tabu Search
- Genetic Algorithm
- Ant Colony

Problem Statements

- L libraries;
- N books;
- T days required for signup;
- M Books that can be shipped per day (after the signup process is done);
- Objective: Maximize the number of books that can be scanned before Time is over;
- Constraints:
  - Each library can only be scanned/signup once;
  - The scan of the books can only be done after the library signup;
  - The number of books that we can ship per day is defined for each library;

State Representation and Initial State

- State Representation
  - [LE,BE,BS,LS,CD]
  - LE -> Libraries Explored : list
  - BE -> Books Explored : dict {key=library_id,value=[books_explored]}
  - BS -> Books Scores : dict {key = book_id, value=score}
  - LS ->Library Signing : list
  - CD -> Current Day : integer

- Initial State
  - [LE,BE,BS,LS,CD]
  - LE = []
  - BE = {}
  - BS = {Book Scores}
  - LS = []
  - CD = total_days

Heuristics

- 1) Library Score:
  - The books are first organized by descending order of their score;
  - Calculate the number of books that are possible to read in the D days;
  - Get the number of books that can be read in the possible days;
  - Calculate the total score of the books that can be read;
  - Calculate the average score per day it takes to sign up the library;
- 2) Signup:
  - Order the libraries by their signup time by ascending order;

Hill Climbing

- We initially implemented a Hill Climbing algorithm. This algorithm starts with a randomized starting solution and iteratively explores neighboring states, seeking the neighbor that maximizes the total score. Operations available to the algorithm:
  - Swap books explored, add/remove books from the books explored dict, add/remove libraries from the library explored list and add/remove libraries from the library signing list.
- After a comprehensive neighbor search, the algorithm identifies feasible neighbors and selects the one with the highest score to compare with the current solution.
- If a best neighbor is found, it becomes the new current state, and the process repeats.
- If no better neighbors exist, the algorithm, ends.
- Drawbacks:
  - Local Optima : Hill Climbing algorithms are susceptible to becoming trapped in local optima, which are solutions better than their immediate neighbors but not globally optimal.
  - Computational Cost: Evaluating all potential neighbors in each iteration can be computationally expensive, especially for large and complex problems.

Simulated Annealing

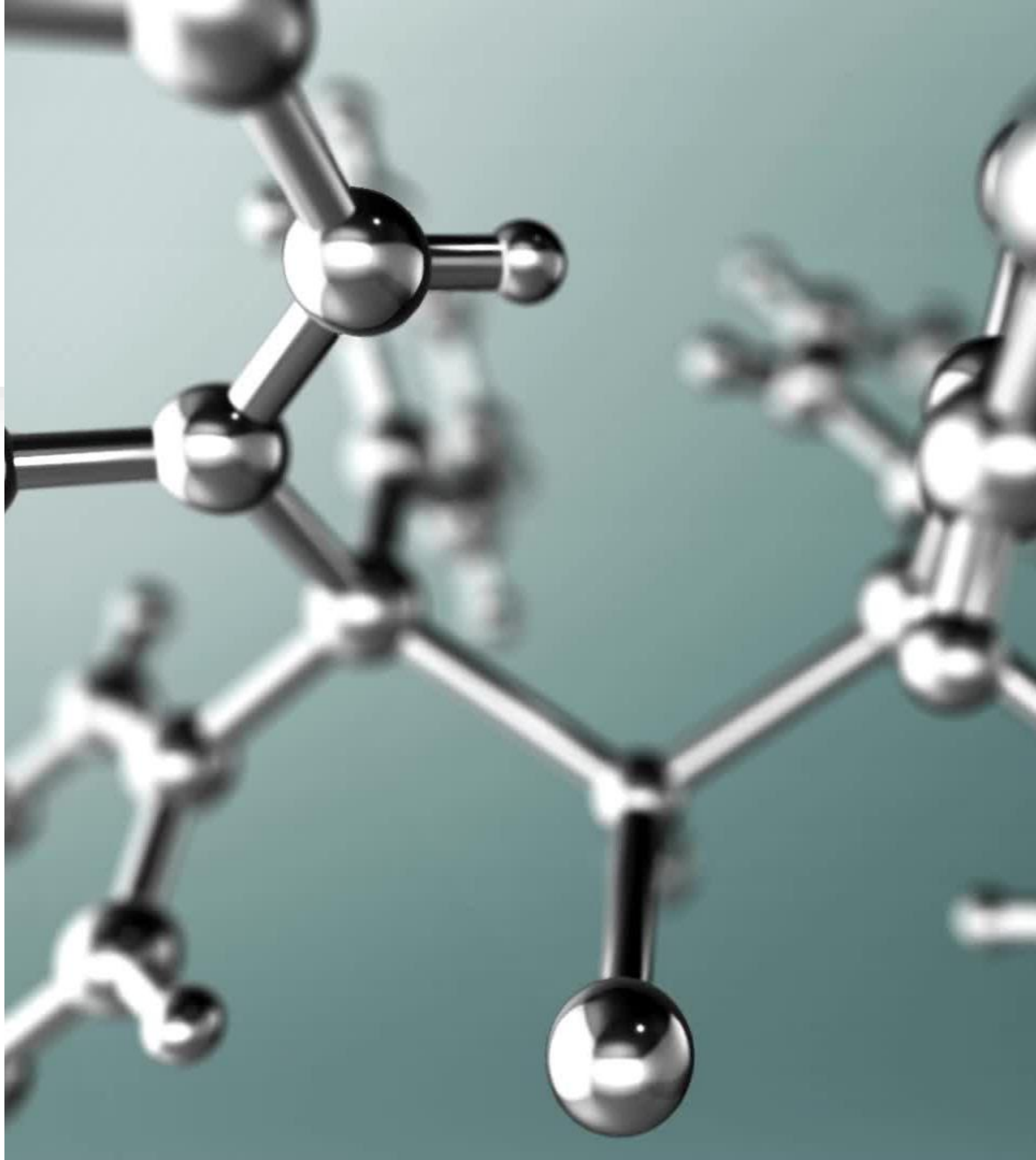
- Our Simulated Annealing algorithm aims to find better solutions by strategically accepting worse solutions during its search, increasing the chance of escaping local optima.
- Begins with an initial randomized solution, and calculated the initial temperature, the temperature is used to control the probability of accepting worse solutions.
- Randomly select a subset of operations (same as Hill Climbing) and applies them to the current state, generating multiple potential neighbors. After this, calculates the score of each neighbor and the current state.
- If a neighbor has a better score, it is automatically accepted as the new current state, although if it has a worse score, it might still be accepted based on a calculated probability that depends on the score difference and the current temperature.
- The temperature gradually decreases, reducing the likelihood of accepting worse solution over time.
- The algorithm stops when the temperature reaches a near-zero value or if there hasn't been an improvement within a specified number of iterations.
- Drawbacks:
    - Initial Temperature Calculation : If the initial neighbors are of poor quality, the algorithm mught start with na inappropriatly high or low temperature;
    - Neighbor Selection : The code chooses the neighbors randomly, so we can't predict the "path";

Tabu Search

- This algorithm begins with a randomized initial solution, and an empty list to store recently visited states.
- Starts by applying a set of heuristic functions (similar to the hill climbing) to the current solution and generates a list of potential neighboring states.
- Determines the best neighbor based on their total score and updated the tabu list by adding the next solution and removing the oldest entry if the list exceeds the tabu size.
- Sets the current solution to the next solution and updated the best solution if a better neighbor is found
- This algorithm is going to stop after a maximum number of iterations or if no neighbors are found.
- Drawbacks:
  - Tabu List Size : The fixed size might be too restrictive or too leniente depending on the problem.
  - Heuristic Reliance : The effectiveness of Tabu Search heavily depends on well-designed heuristics. Poor heuristics may lead to limited neighbor quality .

Genetic Algorithm

- This algorithm aims to find the State that yields the highest fitness score using the auxiliary function get_value.
- The Genetic Algorithm starts with a population of initial State objects. Over multiple generations, it mimics natural evolution:
  - **Elitism** : The very best solution is preserved each generation;
  - **Selection**: Parent States are chosen for reproduction, either using a roulette wheel method (where higher fitness gives better odds) or a tournament method (where groups compete and the fittest is selected);
  - **Crossover** : Parents produce childs through an ordered_crossover process. This carefully combines libraries from the parents, ensuring that the solutions(childs) are feasible;
  - **Mutation**: Introduces diversity to the population by applying mutations to the childs.



Ant Colony Algorithm

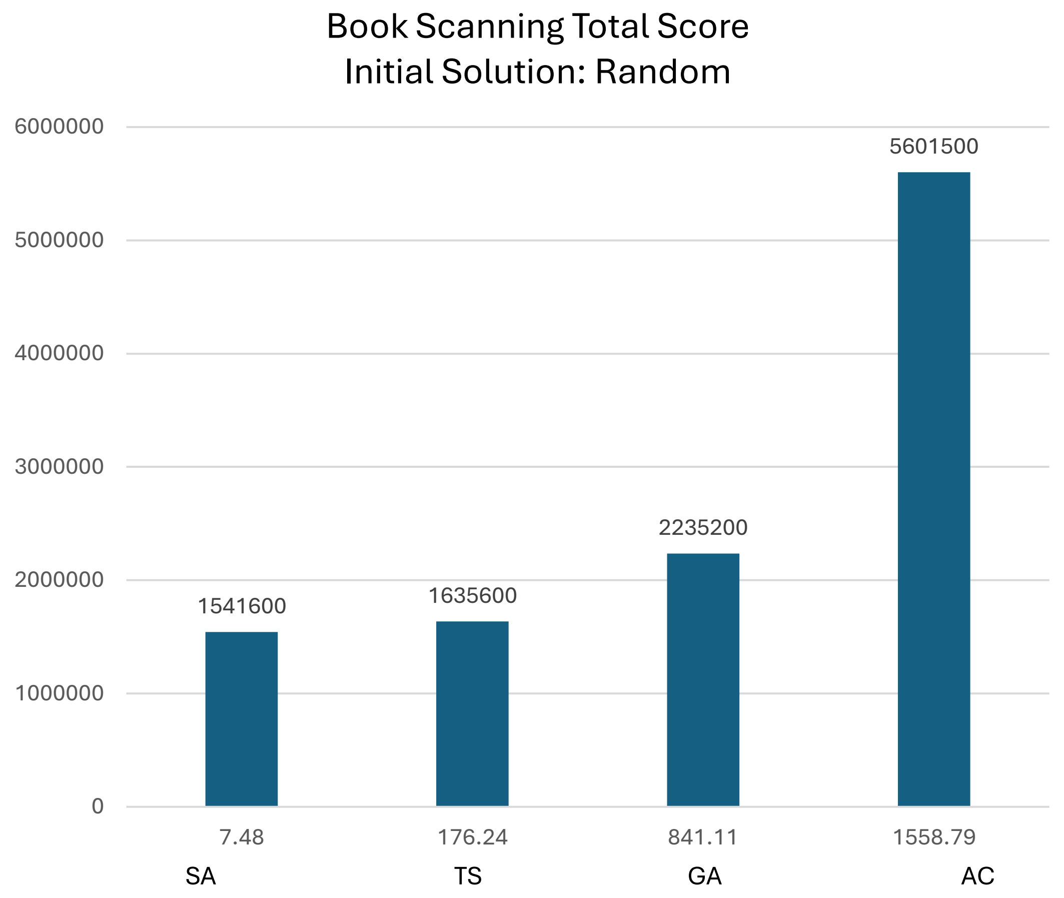
- This Algorithm draws inspiration from how ants find efficient paths to food sources by leaving pheromone trails. Strong trails attract more ants, reinforcing good paths.
- The algorithm provides a pheromones matrix, where each entry represents the strength of a trail between libraries.
- Each ant constructs a potential solution to the problem, the trails are then adjusted based on solutions and track the overall best solution found.
- Ants don’t blindly follow the strongest trails because there’s an element of randomness to encourage exploration.
- We have two parameters to control the importance of the heuristic versus the pheromones: alpha and beta.
- Our implementations avoids purely random trial-and-error, instead learns which choices tend to be part of good solutions.

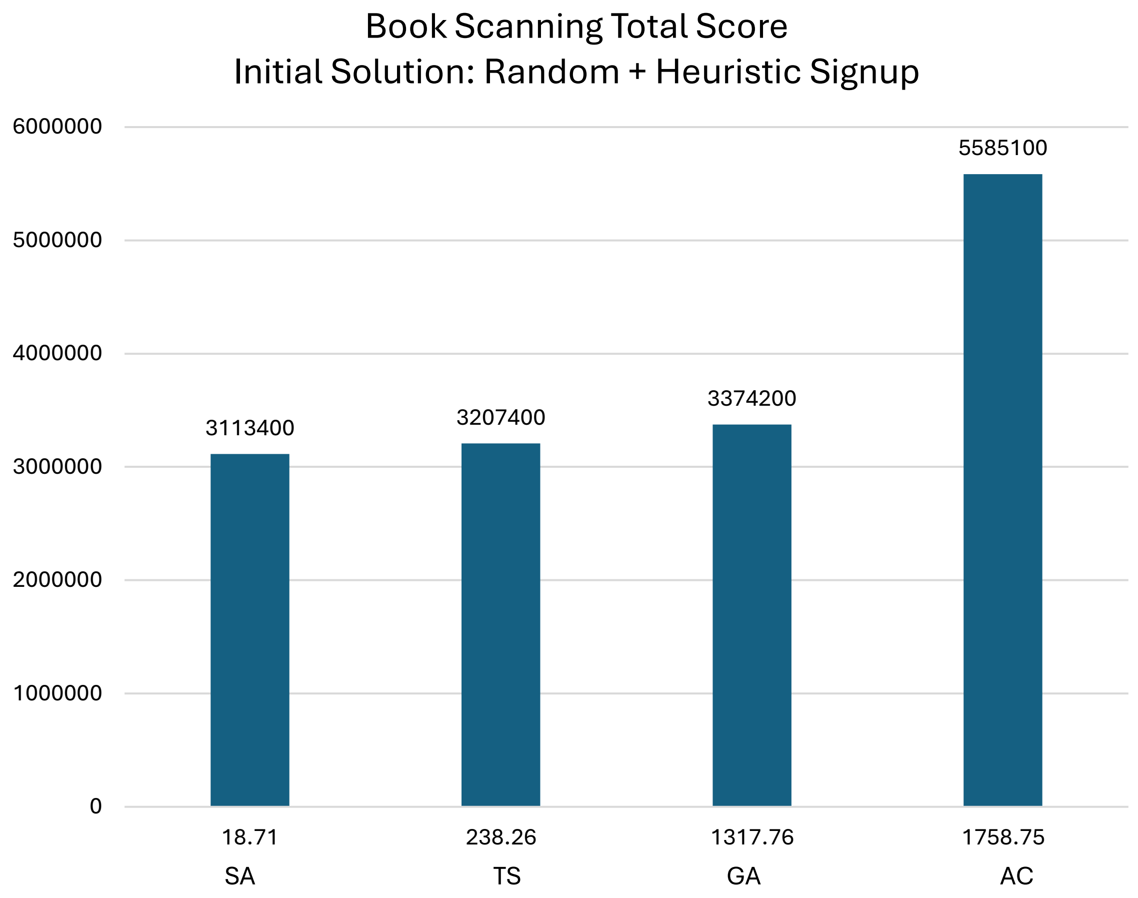
Analysis

- In optimization problems, it's essential to compare algorithm-generated solutions to determine the most suitable one for a given dataset.
- Experimenting with different initial solutions types can be beneficial:
    - Random: provides a baseline for comparison;
    - Random + Heuristic : Offers a guided starting point by prioritizing libraries with low signup times;
    - Greedy: Emphasizes maximizing the number of books explored by selecting libraries with high score.

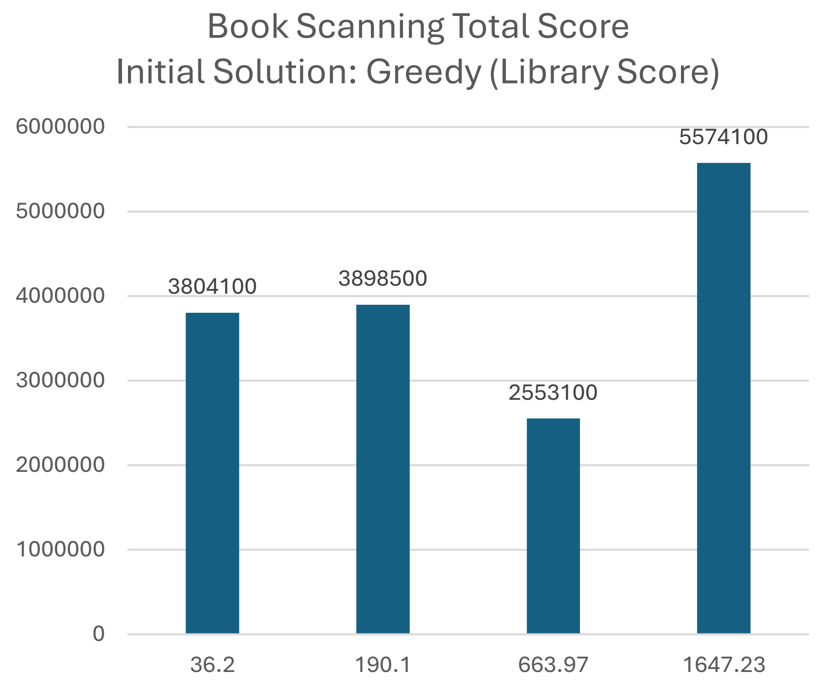
- To evaluate the solutions, we used the get_value function that prioritizes by the following order:
    - 1) Total Score;
    - 2) Number of Libraries Explored;
    - 3) Number of Books Explored;
    - 4) Give a bonus if it has something in the Library Signing List;

Analysis


Book Scanning Total Score
Initial Solution: Random
6000000
5000000
4000000
3000000
2000000
1000000
0
1541600
1635600
2235200
5601500
7.48
176.24
841.11
1558.79
SA
TS
GA
AC


Book Scanning Total Score
Initial Solution: Random + Heuristic Signup
6000000
5000000
4000000
3000000
2000000
1000000
0
3113400
3207400
3374200
5585100
18.71
238.26
1317.76
1758.75
SA
TS
GA
AC

Analysis



- The plots demonstrate superiority of the Greedy Heuristic (using library score) for initializing Simulated Annealing and Tabu Search.
- As expected, the Greedy approach hinders Genetic Algorithms, which thrive on diversity. In this case, a heuristic focused on library signup time fosters the needed diversity.
- Since Ant Colony Optimization constructs its own solutions, a provided initial solution is unnecessary. Therefore, solutions remain consistent for specific dataset.

Conclusion

- All five Algorithms – Hill Climbing, Simulated Annealing, Tabu Search, Genetic Algorithm and Ant Colony Optimization are influenced by their initial starting points.
- Randomized solutions introduce variability, potentially impacting final solution quality, although the optimal algorithm depends on your specific needs.
- **Speed** : Simulated Annealing;
- **Deep-Search**: Hill Climbing;
- **Diversity:** Genetic Algorithm;
- **Consistent**: Ant Colony.