Fernando da Silva Pereira Borges Afonso
Gonçalo de Abreu Matias
Tiago Tavares Simões

Artificial Intelligence

Optimization Methods
Meta-Heuristics

Assignment No. 1
Airport Landing

Work Specification

- **Problem Definition:**
  - Scenario: Manage a continuous stream of airplanes landing at an airport with three landing strips.
  - Key Challenge: Design an algorithm for efficient and safe landing management.

- **Objectives:**
  - Safety First: Ensure minimum one hour of fuel is left upon landing for each airplane.
  - Optimal Scheduling: Minimize delays while respecting expected landing times.

- **Constraints:**
  - Constant fuel consumption rate.
  - Adherence to expected landing times.
  - Minimize crash risks and optimize landing efficiency.
  - Landing strip occupation post-landing: 3 minutes.
  - Airport's max capacity: 60 landings per hour.

- **Critical Failures:**
  - Any plane crash during landing is considered a failure of the algorithm.

- **Objective Function:**
  - Minimize crash risks while optimizing for fuel efficiency and punctuality.

- **Input Data Generation:**
  - Script provided to simulate arrivals with variable fuel levels, consumption rates, and expected landing times.

Formulation of the Optimization Problem

- **Solution Representation:** An array with each element representing a time slot for a specific runway. Each time slot will have an aircraft assigned based on the optimization criteria.
- **Neighborhood/Mutation Functions:** Swapping two aircraft landing times to find a better schedule, or slightly altering the expected landing time within safe margins to explore time slot efficiencies. Inversion mutation where a subset of the landing sequence is reversed to prevent local optimum entrapment.
- **Crossover Functions:** A single-point crossover where one part of a schedule (parent) is combined with another schedule (parent) to create a new schedule (child), preserving essential sequences from both parents. Uniform crossover where each time slot for the landing is chosen randomly from one of the parents, ensuring variety in the offspring schedules.
- **Hard Constraints:** Safety First, Optimal Scheduling, Fuel Consumption Rate, Expected Landing Times, Landing Strip Occupation Time, Airport Capacity.
- **Evaluation Functions:** Maximize Fitness Function - Higher fitness values should correspond to fewer crashes and better adherence to landing times and fuelefficiency rules.

Work Implementation

Development Environment:
Languages & Tools: Python with VS Code and PyCharm for collaborative coding.

Data Structures:
Classes for airplane objects, lists for arrivals, pandas data frames for data management.

Algorithm Implementation:
Development of Hill Climbing, Tabu Search, Simulated Annealing underway.

Interactivity & Visualization:
Functions for user interaction, visualization of algorithm performance.

Approach

Solution Evaluation:
- For Each Airplane:
  - Time Deviation: Absolute difference between expected and actual landing times.
  - Urgency Consideration: Airplanes with low fuel or near their landing time are marked urgent.
  - Score Calculation: `Score = 1000 - (Time Deviation + Urgency Penalty)`
  - Total Score: Summation of individual scores, with a perfect score being 0.

Execution Flow:
- Initialization:
  - Users define the simulation parameters (e.g., fuel levels, expected arrival times).
  - A stream of airplanes is generated, each with a randomized fuel level and expected landing time.

- Scheduling:
  - Urgent airplanes are given priority in the scheduling process.
  - Landing times are assigned based on the availability of runways and the urgency status.

- Algorithm Execution:
  - Users select the optimization algorithm (Hill Climbing, Simulated Annealing, Tabu Search, Genetic Algorithm).
  - The selected algorithm processes the airplane stream and produces an optimized landing schedule.
  - Efficiency scores are calculated to evaluate the performance of the algorithm.

Key Metrics:
- Time Deviation: A lower deviation from expected landing times is better.
- Urgency Response: Ability to handle urgent landings efficiently.
- Overall Score: Lower scores indicate a more optimal landing schedule.

Implemented Algorithms: Search Algorithms and Metaheuristics

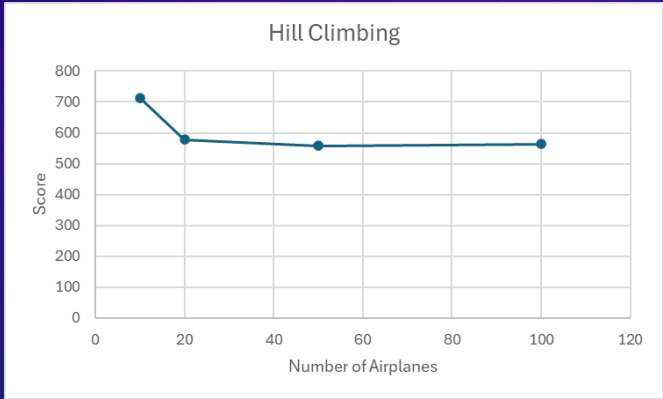
Hill Climbing (HC):
- Progressively refines the landing schedule using schedule_landings.
- Prioritizes aircraft by urgency derived from fuel metrics and expected times.
- Search halts when it encounters a plateau with no score improvement.

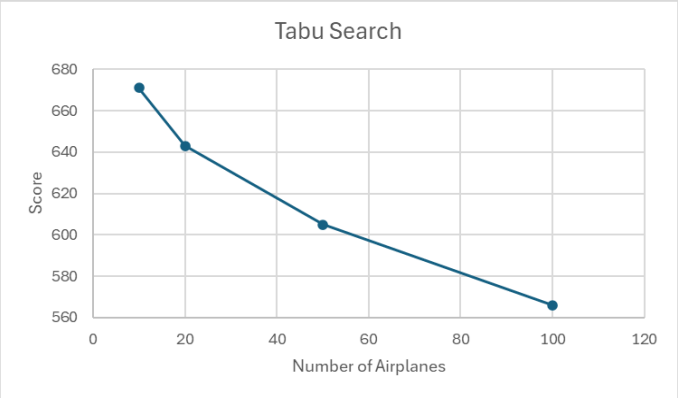
Simulated Annealing (SA):
- Introduces randomness to transition beyond local optima encountered by HC.
- Adopts a cooling schedule, mirroring metallurgical annealing, to minimize score.
- Alternates between exploration and refining, influenced by a thermal probability model.

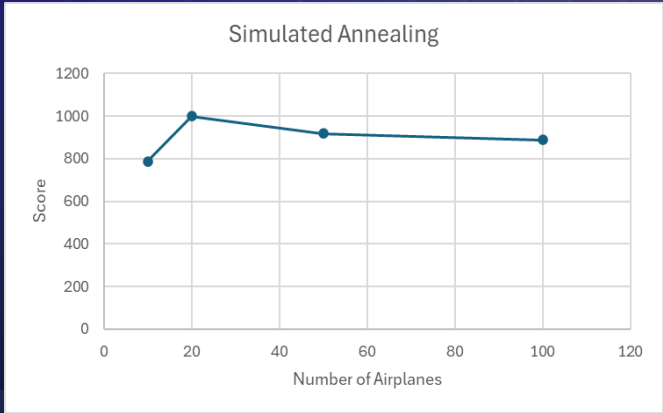
Tabu Search (TS):
- Leverages a tabu list to prevent cyclic revisits, propelling the search forward.
- Methodically searches for global optima, guided by past state avoidance.
- Persistence in the search is maintained up to a predefined iteration count.

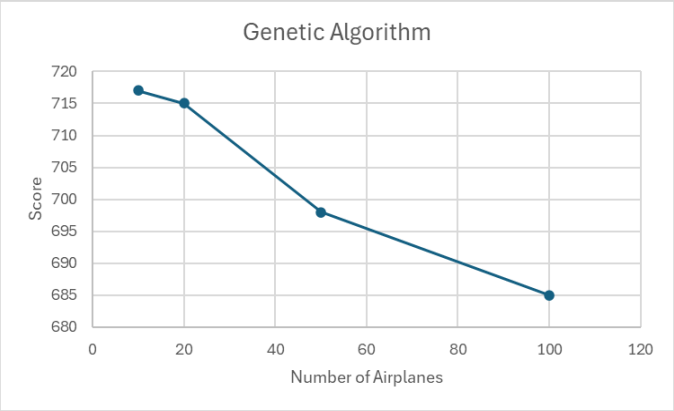
Genetic Algorithm (GA):
- Emulates evolutionary processes, iterating over a population of schedules.
- Genetic operations include crossover and mutation to foster schedule diversity.
- Selection process favors higher-scoring schedules for subsequent generations.

Experimental Results and Comparative Analysis (1/2)

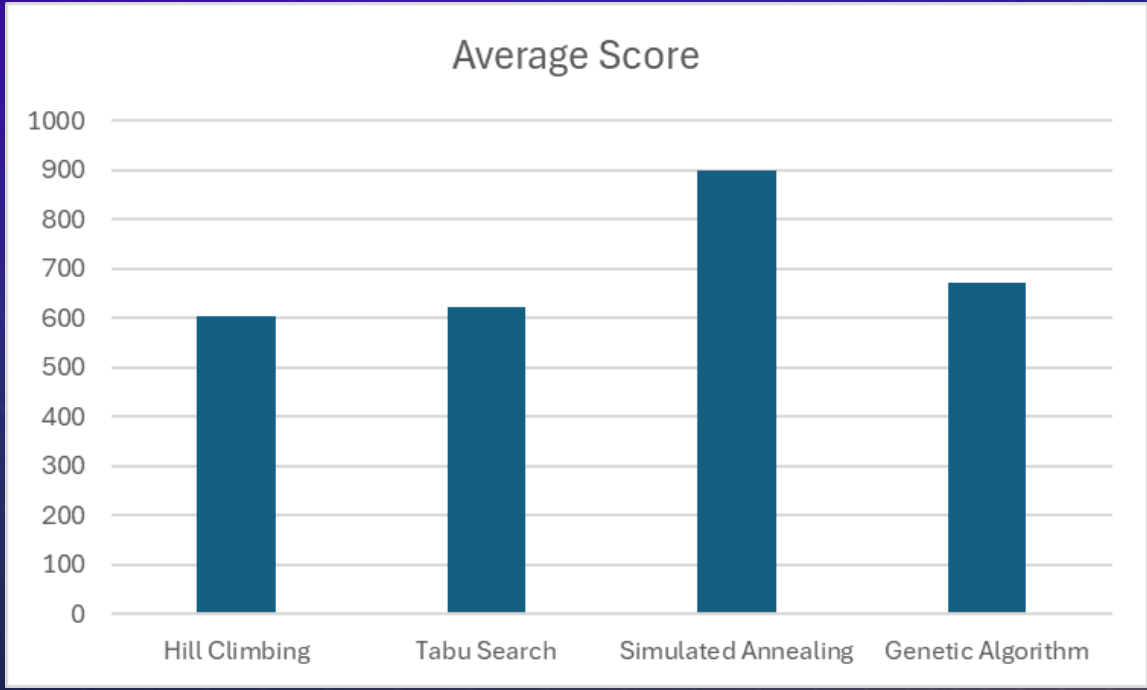








Experimental Results and Comparative Analysis (2/2)



References

During the investigation phase of our initiative, the team encountered multiple algorithmic solutions suitable for our endeavor, including Python-coded greedy algorithms, genetic algorithms, and numerous online articles detailing various methodologies.

https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

https://www.geeksforgeeks.org/what-is-tabu-search/

https://www.baeldung.com/cs/simulated-annealing

https://www.youtube.com/watch?v=rA3a8QDtYLs

https://github.com/jedrazb/python-tsp-simulated-annealing

https://github.com/topics/genetic-optimization-algorithm

Conclusion

We successfully implemented various optimization algorithms with relative smoothness. This project served as a valuable opportunity to delve deeper into the intricacies of Optimization Problems and the nuanced strategies of their corresponding algorithms. While the journey was challenging, the knowledge gained and the insights acquired have been immensely rewarding.

The algorithms—Hill Climbing, Simulated Annealing, Tabu Search, and Genetic Algorithms—each revealed unique strengths and limitations when faced with complex optimization scenarios. Our comparative analysis has not only highlighted the importance of algorithm selection based on specific problem constraints but also reinforced the need for adaptability and critical evaluation in problem-solving.

We believe we have met all the proposed objectives and presented our findings in a clear comprehensive, and objective manner. More importantly, this project has sharpened our problem-solving skills, which we are confident will serve us well in future endeavors.