Book Scanning

César Nogueira
Maria Baía
Tomás Mendes

IART - Artificial Intelligence

Problem Specification

Problem Statement: Google has a book scanning program for libraries. Given a list of libraries, their sign up times and their books, choose the order in which they will sign up for the program, as well as the order in which their books will be scanned after sign up. Only one library may sign up at a time but the scanning process can occur in parallel for several libraries.

Objective Function: Each book is described by a score and we aim to maximize the sum of the scores of scanned books, in the alloted time, with no repeated books.

Problem Formulation

Solution representation: Array where the index is the order of sign up and the value is the library ID

Constraints:

- Only books scanned before the deadline are accounted
- Each book should only be scanned once
- Each library can only start its sign up process after all the libraries before it have finished their sign up process
- Each library can only start scanning books after finishing its sign up process

Evaluation function: Sum of all the scores of the scanned books within the allotted time (if the same book is scanned twice, the score is only awarded once)

Problem Formulation

Hill Climbing / Simulated Annealing

Initial solution:

- Random
- Greedy, ordered by libraries with highest total score first
- Greedy, ordered by libraries with smallest sign up time first

Neighborhood: Switch the order of two consecutive libraries in the solution array

Genetic Algorithm

Initial solution:

- Random

Mutation: Randomly switch the order of two libraries in the solution array

Crossover: Applies the order-based crossover, that given two parent solutions, it will generate a new solution

Project Structure

- **Programming language**: Python 3;
- **Development environment**: Visual Studio Code;
- **Data structures:**
  - Lists;
  - Library class;
  - Solver class.
- **File structure:**
  - input: Folder that contains the input data files;
  - output: Folder that contains the output data files;
  - main.py: File that contains the developed code.

Approach

Evaluation Function (Score)

For this optimization problem, we started by adopting a heuristic that obtained the real score, as described by Google. However, this approach consumed too much time. So we decided to try other approaches for the heuristic, only using this function to check the real score of the final solution.

Heuristics

We implemented three different heuristics, to check the fitness of a solution:

- Real score: obtained by going through the libraries in order of sign up, going through each book in the library and summing their score
- Real score (duplicates): obtained by calculating the sum of the N best books in each library at the beginning (saved in an array of size N) and only going through each library and getting the score directly from the array (duplicates are counted twice). Very close to the real score
- Simple Heuristic: heuristic based on a solution from a team that participated in Google Hash Code. Score obtained using this heuristic is not close to the real score, only a representation of the fitness of the solution

Algorithms

Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. This way, it finishes when it reaches a peak value where no neighbour has a higher value than the current best solution.

First Accept Hill Climbing

First accept hill climbing is the simplest way to implement a hill climbing algorithm. Basically, it examines the neighbouring nodes one by one and selects the first neighbouring node that has a better score, setting it as the current state. Furthermore, this algorithm consumes less time, but, on the other hand, the optimal solution is not guaranteed.

Steepest Hill Climbing

The steepest hill climbing is a variation of first accept hill climbing. This algorithm examines all the neighbouring nodes of the current state and selects the neighbour node that has the higher score. This way, consumes more time as it searches for multiple neighbours.

Algorithms

Simulated Annealing

Knowing that the Hill Climbing algorithm tends to get stuck in local maximas, the Simulated Annealing algorithm aims to overcome this problem, accepting bad solutions with a given probability.
It starts by establishing an initial temperature and solution, based on the chosen initial approach.
At each iteration, a list of solutions neighbouring the current solution is generated, given by the neighbourhood operator, and a solution is chosen randomly from the list. If the given solution is better than the current solution, it becomes the current solution, otherwise, it can be with a given probability. Also, if the current solution is better than the best solution, that must be updated. At the end of each iteration the temperature must be decreased at a random rate (between 0.9 and 0.99). The cycle ends when the temperature reaches 0.

Genetic Algorithm

The algorithm starts by generating an initial population with solutions organized by the IDs of the libraries randomly. Based on the mechanics of biological evolution, each generation creates a new population applying the order-based crossover operator, that given two parents selected using the tournament selection method, they will generate a new solution and, with a probability of 0.05%, the mutation operator can be applied to the new solution.
It was also applied an elitist approach, that with each new generation, the solution with the best fitness of the current population with size N is maintained, being necessary to generate N-1 new solutions.

Results Analysis

All input files were tested, with the exception of *a_example.txt*, due to its simplicity, and *d_tough_choices.txt* due to the excessive processing time.

All algorithms were tested with three different initial approaches, with the exception of the genetic algorithm, which is always imposed a random initial approach.

This way, we noticed that, for the **random solution**, real score (duplicates) - our first heuristic -, generally, gets better results in terms of final score (especially for the hill climbing algorithm), but consumes more time, with file *c_incunabula.txt* being the exception in terms of time and file *e_so_many_books.txt* in terms of score for the simulated annealing algorithm.

Secondly, for the **greedy score** initial approach, we had a better maximum score in files *b_read_on.txt* and *e_so_many_books.txt* for real score (duplicates). The other two registered similar scores. Once again, file *c_incunabula.txt* was the exception in terms of time consumption, being simple heuristic faster generally.

Results Analysis

For our last approach (**greedy sign up**), real score (duplicates) registered better results for the input files *e_so_many_books.txt* and *f_libraries_of_the_world.txt*. The other two had similar results. In what concerns to time consumption, file *c_incunabula.txt* worked faster with real score (duplicates) and the others needed basically the same time to run the algorithm (the only exception was file *e_so_many_books.txt* for the hill climbing algorithm which was faster with simple heuristic).

Finally, talking about the genetic algorithm, simple heuristic had better maximum scores. On the other hand, real score (duplicates) worked faster, consuming less time.

All tests are contained in the attachment slide, for a more detailed assessment.

Conclusion

For the development of this project, it was necessary to apply all the concepts given in the course of Artificial Intelligence.

We initially adopted a heuristic approach that aims to get the best possible score. However, it had an excessive processing time because it doesn't count the score of duplicated books, which led us to look for other heuristics. To overcome this problem, we implemented a heuristic that didn't take into account duplicated books. Also, we tried a simpler approach, which calculates a score for each library that is not the real one but approximated, which proved to be less effective but much faster than the others.

The project was successfully completed, covering all topics requested.

References

- Problem Statement
- Python 3 Documentation
- Stuart, R., 2010. *Artificial Intelligence: A Modern Approach.* Prentice Hall
- IART Slides on Optimization and Local Search
- IART Slides on Simulated Annealing and Tabu Search
- IART Slides on Optimization and Genetic Algorithms
- Universitat Autònoma de Barcelona, Master's degree in Modelling for Science and Engineering, Lluís Alsedà on Genetic Operations
- Google Hash Code 2020 Possible Solution

Appendix

input = b_read_on

heuristic = Real score (duplicates)

initial solution = random solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	5764700	5357070.0	29.494
Steepest Hill Climbing	5625800	5469800.0	31.355
Simulated Annealing	4624400	4410000.0	10.553
Genetic Algorithm	4032300	3906980.0	7.987

initial solution = greedy score solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	5537400	5537400.0	32.19
Steepest Hill Climbing	5537400	5537400.0	33.26
Simulated Annealing	4349600	4309010.0	9.484
Genetic Algorithm	4032300	3906980.0	7.987

initial solution = greedy signup solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	5822900	5822900.0	23.23
Steepest Hill Climbing	5822900	5822900.0	17.87
Simulated Annealing	5822900	5822900.0	16.901
Genetic Algorithm	4032300	3906980.0	7.987

heuristic = Simple Heuristic

initial solution = random solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	4376400	4193420.0	10.673
Steepest Hill Climbing	4533200	4177900.0	8.976
Simulated Annealing	4396600	4225700.0	9.115
Genetic Algorithm	4476300	4284650.0	9.951

initial solution = greedy score solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	4126100	4126100.0	9.56
Steepest Hill Climbing	4126100	4126100.0	8.34
Simulated Annealing	4126100	4126100.0	8.806
Genetic Algorithm	4476300	4284650.0	9.951

initial solution = greedy signup solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	5822900	5822900.0	19.43
Steepest Hill Climbing	5822900	5822900.0	17.98
Simulated Annealing	5822900	5822900.0	16.615
Genetic Algorithm	4476300	4284650.0	9.951

input = c_incunabula

heuristic = Real score (duplicates)

initial solution = random solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	1006166	885993.1	11.88
Steepest Hill Climbing	976136	886677.0	11.92
Simulated Annealing	930544	877429.8	322.02
Genetic Algorithm	820605	795581.1	9.53

initial solution = greedy score solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	1413940	1413940.0	9.22
Steepest Hill Climbing	1413940	1413940.0	9.18
Simulated Annealing	1413940	1413940.0	356.71
Genetic Algorithm	820605	795581.1	9.53

initial solution = greedy signup solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	5469473	5469473.0	44.26
Steepest Hill Climbing	5469473	5469473.0	41.27
Simulated Annealing	5467966	5467966.0	372.75
Genetic Algorithm	820605	795581.1	9.53

heuristic = Simple Heuristic

initial solution = random solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	955502	890977.4	302.7
Steepest Hill Climbing	954544	895913.2	176.41
Simulated Annealing	925427	875936.0	321.32
Genetic Algorithm	935661	885344.5	143.18

initial solution = greedy score solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	1413940	1413940.0	210.74
Steepest Hill Climbing	1413940	1413940.0	152.01
Simulated Annealing	1413940	1413940.0	399.25
Genetic Algorithm	935661	885344.5	143.18

initial solution = greedy signup solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	5467966	5467966.0	232.64
Steepest Hill Climbing	5467966	5467966.0	158.44
Simulated Annealing	5467966	5467966.0	409.0
Genetic Algorithm	935661	885344.5	143.18

input = e_so_many_books

heuristic = Real score (duplicates)

initial solution = random solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	1396152	1129930.6	12.899
Steepest Hill Climbing	1464404	1264100.4	25.23
Simulated Annealing	1019314	871254.2	70.58
Genetic Algorithm	795214	685237.3	01.05

initial solution = greedy score solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	1580357	1580357.0	19.05
Steepest Hill Climbing	1580374	1580374.0	36.74
Simulated Annealing	1144497	1133548.3	75.12
Genetic Algorithm	795214	685237.3	01.05

initial solution = greedy signup solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	4835131	4835131.0	255.65
Steepest Hill Climbing	4834590	4834590.0	723.86
Simulated Annealing	4272471	4268649.0	77.63
Genetic Algorithm	795214	685237.3	01.05

heuristic = Simple Heuristic

initial solution = random solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	1035610	879255.6	1.47
Steepest Hill Climbing	1198217	862533.0	1.44
Simulated Annealing	1206156	954573.7	74.83
Genetic Algorithm	990077	870088.1	7.44

initial solution = greedy score solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	1115194	1115194.0	1.3
Steepest Hill Climbing	1115194	1115194.0	1.56
Simulated Annealing	1123686	1115601.7	77.09
Genetic Algorithm	990077	870088.1	7.44

initial solution = greedy signup solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	4261958	4261958.0	5.79
Steepest Hill Climbing	4261958	4261958.0	5.78
Simulated Annealing	4261958	4261958.0	79.82
Genetic Algorithm	990077	870088.1	7.44

input = f_libraries_of_the_world

heuristic = Real score (duplicates)

initial solution = random solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	1513810	1079063.8	0.18
Steepest Hill Climbing	1839361	899945.1	0.18
Simulated Annealing	1377378	849665.9	77.0
Genetic Algorithm	365182	215376.4	0.66

initial solution = greedy score solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	1210345	1210345.0	0.13
Steepest Hill Climbing	1210345	1210345.0	0.12
Simulated Annealing	1215512	1211378.4	76.65
Genetic Algorithm	365182	215376.4	0.66

initial solution = greedy signup solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	2855552	2855552.0	0.81
Steepest Hill Climbing	2855552	2855552.0	0.98
Simulated Annealing	2736577	2634479.5	73.33
Genetic Algorithm	365182	215376.4	0.66

heuristic = Simple Heuristic

initial solution = random solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	1339187	720965.9	01.09
Steepest Hill Climbing	1393953	781456.6	1.14
Simulated Annealing	1281839	795145.4	74.18
Genetic Algorithm	1036985	668539.2	6.34

initial solution = greedy score solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	1210345	1210345.0	01.02
Steepest Hill Climbing	1210345	1210345.0	1.18
Simulated Annealing	1210345	1210345.0	75.4
Genetic Algorithm	1036985	668539.2	6.34

initial solution = greedy signup solution

Algorithm	Maximum Score	Average Score	Time (s)
First Accept Hill Climbing	2392740	2392740.0	1.28
Steepest Hill Climbing	2392740	2392740.0	2.25
Simulated Annealing	2544780	2430894.0	73.71
Genetic Algorithm	1036985	668539.2	6.34