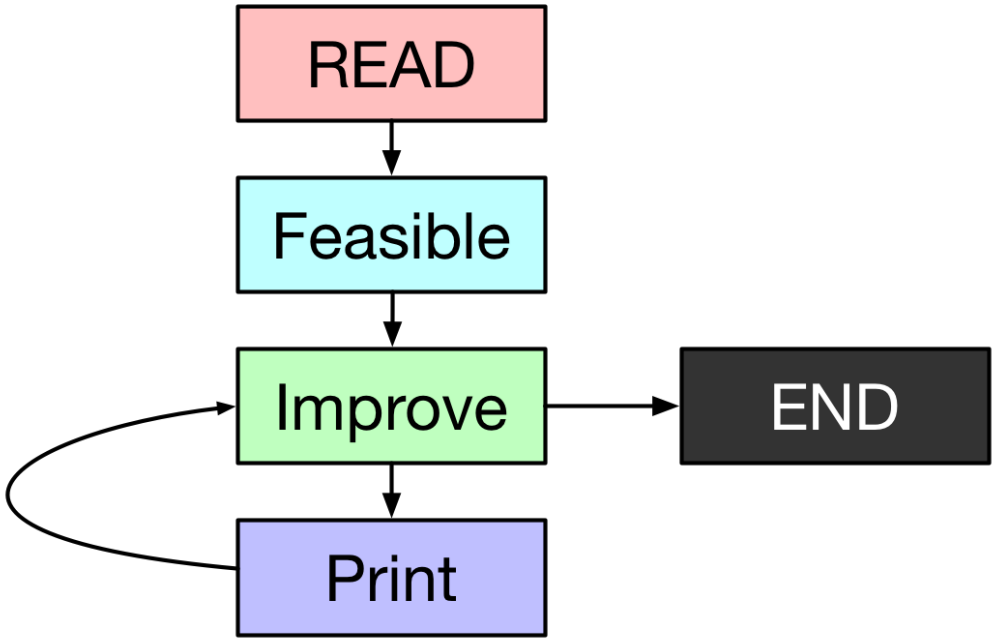
OMA Assignment 2017-2018

Examination Time Tabling : Group 03

Jacopo Maggio – Stefano Munna – Jacopo Nasi – Andrea Santu – Marco Torlaschi

The Problem

- Solving a time scheduling problem trying to minimize the penalties of exam proximities.
- Step:
  1. Read & Store data input
  2. Finding a feasible solution
  3. Improving founded feasible



Data Structure

- The software is written in **C** due to its high performance.
- The main structure is a **GRAPH** built on a adjacency matrix.
- When two exams CAN'T be sustained in the same time slot adjM[e1][e2] = 1, otherwise -1.
- The TABU is of 1000 moves with 7 iterations.

Feasible

Finding Feasible

- The initial idea was to implement a **TABU** SEARCH over the data. Starting with this strategy resulted a little bit slow with those instances that have an high grade of complexity and conflicts.
- The final version use a **GREEDY** algorithm to reduce the complexity and then it pass this partial solution to the **TABU** implementation.
- From ~3 minutes to ~10 seconds.

Greedy

- Trying to **reduce** ”the complexity” of the problem.
- Initial data is sorted from the exam with more collision to the the fewer ones.
- **Workflow**:
  - Use the FIRST available (no collision) time slot for each exam.
  - If there aren’t enough timeslots with this configuration it adds more timeslots.
- The added timeslots will be removed with the next step.

Tabu Search

- It reduces the number of added (by the greedy) time slot till the correct number.
- TO DO…

Improve

Used techniques

Local Search

Local Swap

- Implements a STEEPEST DESCENT STRATEGY.
- Evaluates Neighborhood N(x) of the current optimal solution x.
- **Workflow**:
  - Switches exams scheduled in a timeslot with those contained in any another to find a new feasible solution x'.
  - If x' has a better benchmark than that of x, then x=x'.
  - Loop until there is no improvement anymore
- Slower than FIRST IMPROVEMENT strategy but more efficient.

Simulated Annealing

Greedy Slot Shuffle

Improve

Other techniques tried but not used in the final version…

Benchmark Insertion

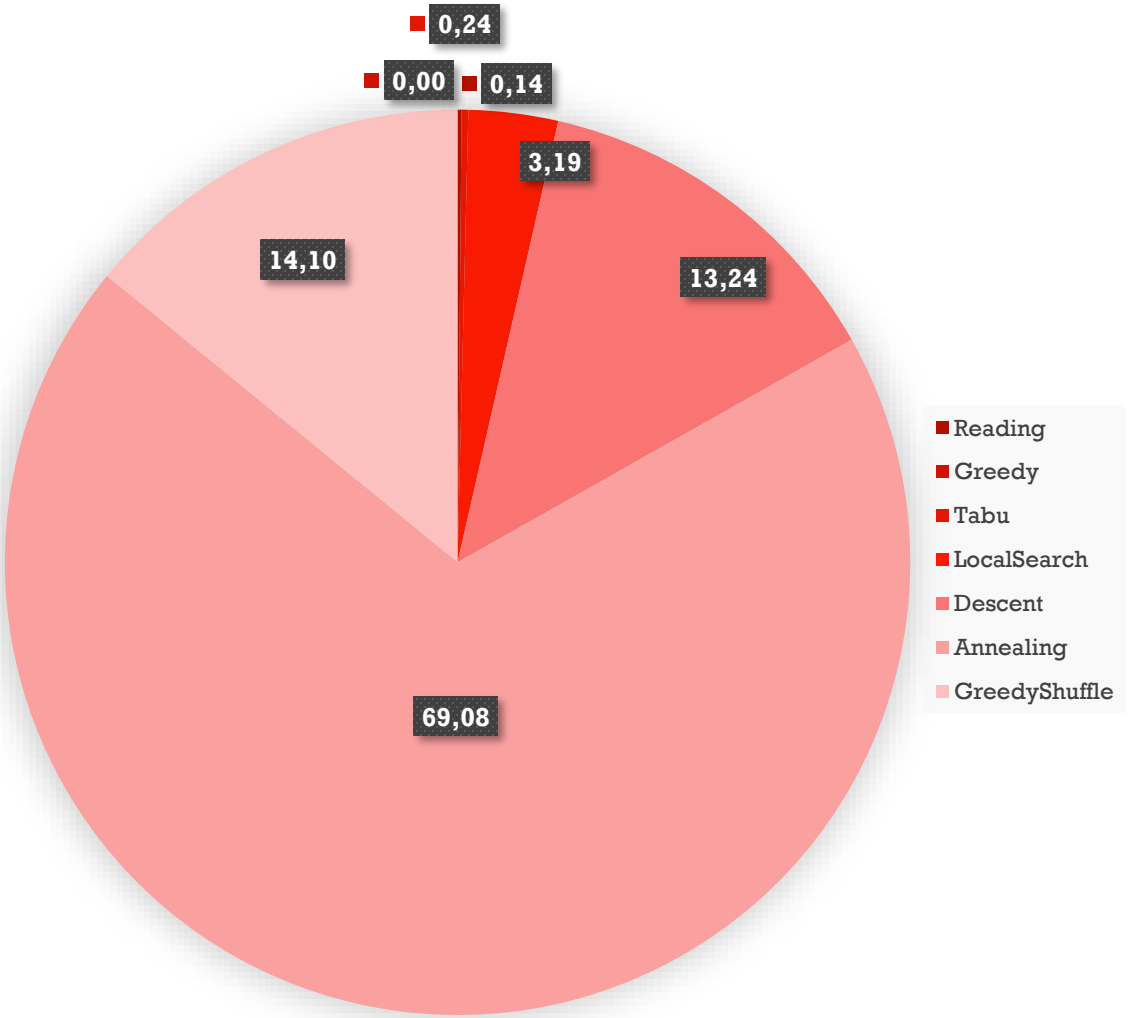
- Starting by selecting a random timeslot and then adding one at the time the best one by checking the benchmark of that temporary solution.
- ADVANTAGES:
  - Avoid ”local minimum” problem
  - Fast and easy to be implemented
- DRAWBACKS:
  - Too random

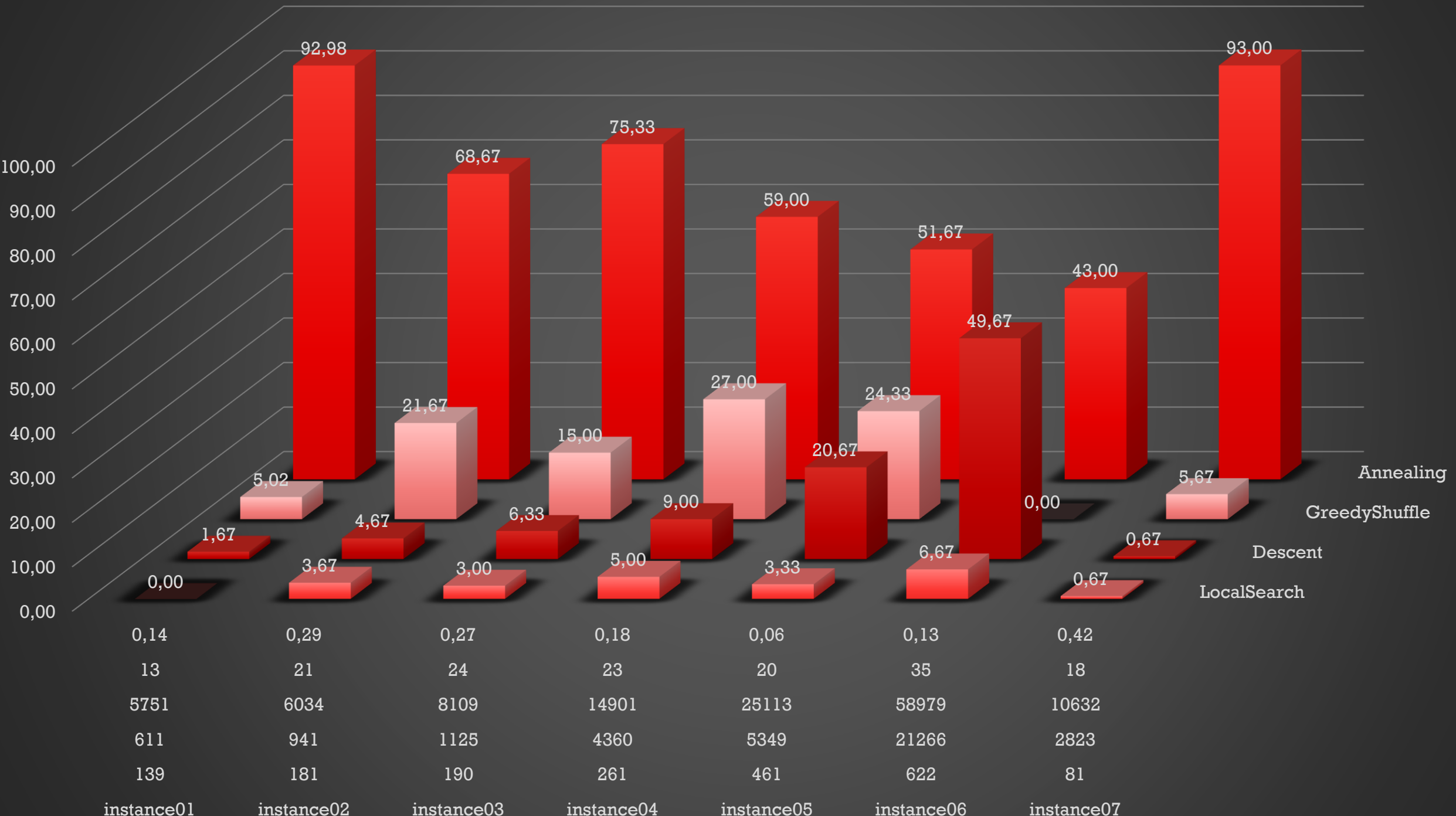
Genetic

- Starting from two or more parents it performs a cross-over between them and occasionally a mutating.
- This algorithm not fit very well this problem because it could generate a lot of unfeasible solution due to its randomness.
- DRAWBACKS:
  - Not too easy
  - Lot of unfeasible solutions

Performance

Average % Algorithm Execution Time




100,00
90,00
80,00
70,00
60,00
50,00
40,00
30,00
20,00
10,00
0,00
92,98
68,67
75,33
59,00
51,67
43,00
93,00
49,67
0,00
5,02
21,67
15,00
27,00
24,33
5,67
1,67
4,67
6,33
9,00
20,67
0,67
0,00
3,67
3,00
5,00
3,33
6,67
0,67
Annealing
GreedyShuffle
Descent
LocalSearch
0,14
13
5751
611
139
0,29
21
6034
941
181
0,27
24
8109
1125
190
0,18
23
14901
4360
261
0,06
20
25113
5349
461
0,13
35
58979
21266
622
0,42
18
10632
2823
81
instance01
instance02
instance03
instance04
instance05
instance06
instance07

Repository

- The whole project with code, math model, instances, presentation and benchmark results are available on a **GitHub** public repository:

https://github.com/Jacopx/OMA_ExamTimeTable

Thanks for the attention!