

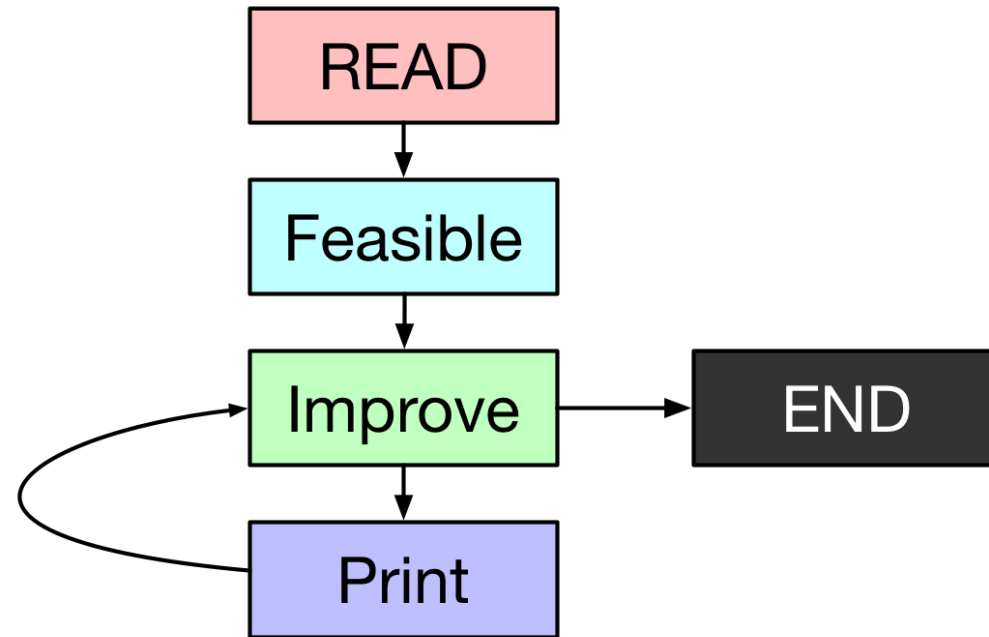
# OMA Assignment 2017-2018

## **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.<sup>2</sup>
- 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.
- We have not implemented multithreading to focusing on algorithms development.
- The main structure is a **GRAPH** built on a adjacency matrix.
- When two exams CAN'T be sustained in the same time slot  $\text{adjM}[e1][e2] = n$  (# of students enrolled in both exams), otherwise -1.



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.
- **Workflow:**
  1. Reduce the timeslots
  2. Try to resolve conflict with the reduced number of timeslots.
    1. If is not able to solve: BACKTRACK
    2. Otherwise: Restart from point 1.
- **ADVANTAGES:**
  - Very reliable
- **DRAWBACKS:**
  - Not too fast



# Improve

Used techniques



## 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.

# Local Search

- Our local search implements 3 different algorithms:
  1. **Move Exam:** Change the timeslot of an exam.
  2. **Swap Exams:** Swap the timeslots of two exams.
  3. **Bounded Shift:** Shift a portion of timeslots.
- These 3 (+ the previous local swap) procedures are used until the solution is improving.

# Simulated Annealing

## ■ Workflow:

1. Start from a feasible solution.
2. Move an exam to another timeslots.
3. If WORST or UNFEASIBLE accept the solution, with probability, depending on temperature.
4. Reduce temperature to converge to a minimum.
5. Reuse the TABU SEARCH to restore the feasibility.
6. Apply local search to find a minimum.

# Greedy Slot Shuffle

- Starting from a feasible, it adds, one at a time a timeslot evaluating the benchmark.
- **ADVANTAGES:**
  - Random
  - Fast

# Improve

Other techniques tried but not used in the final  
version...

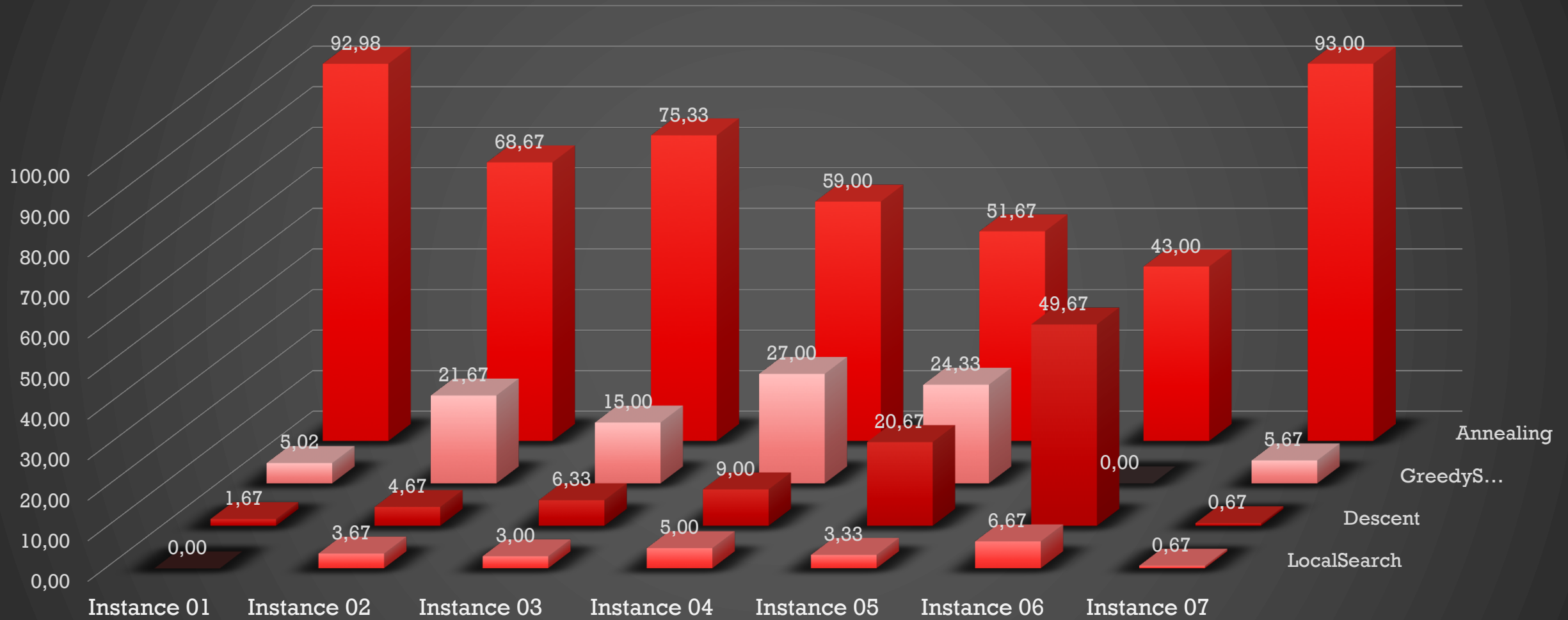
# Genetic

- 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



Execution

## % Algorithms execution time to instance properties

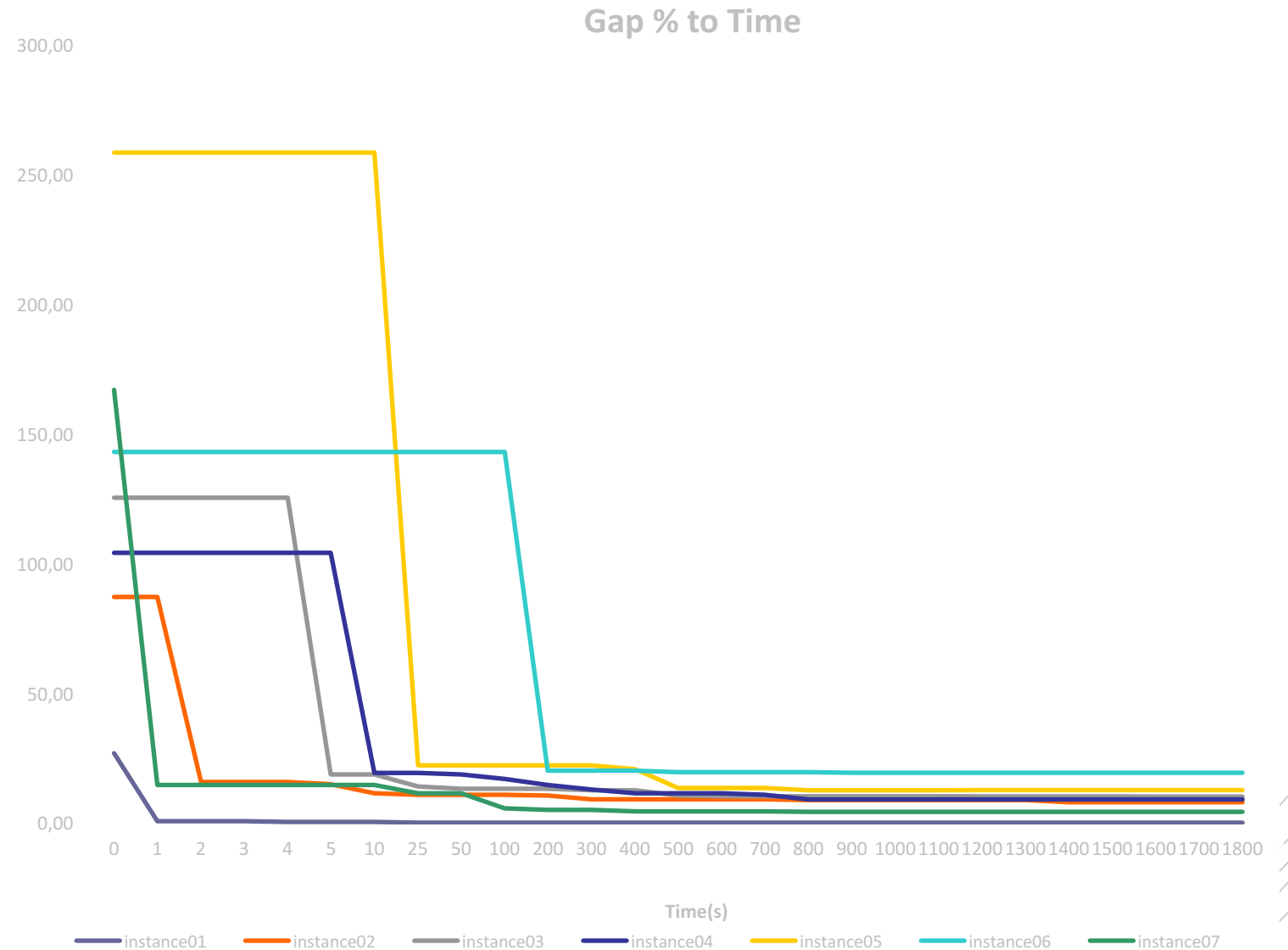




A large red speech bubble graphic with a white outline, pointing downwards. The word "Performance" is written in white inside the bubble.

Performance

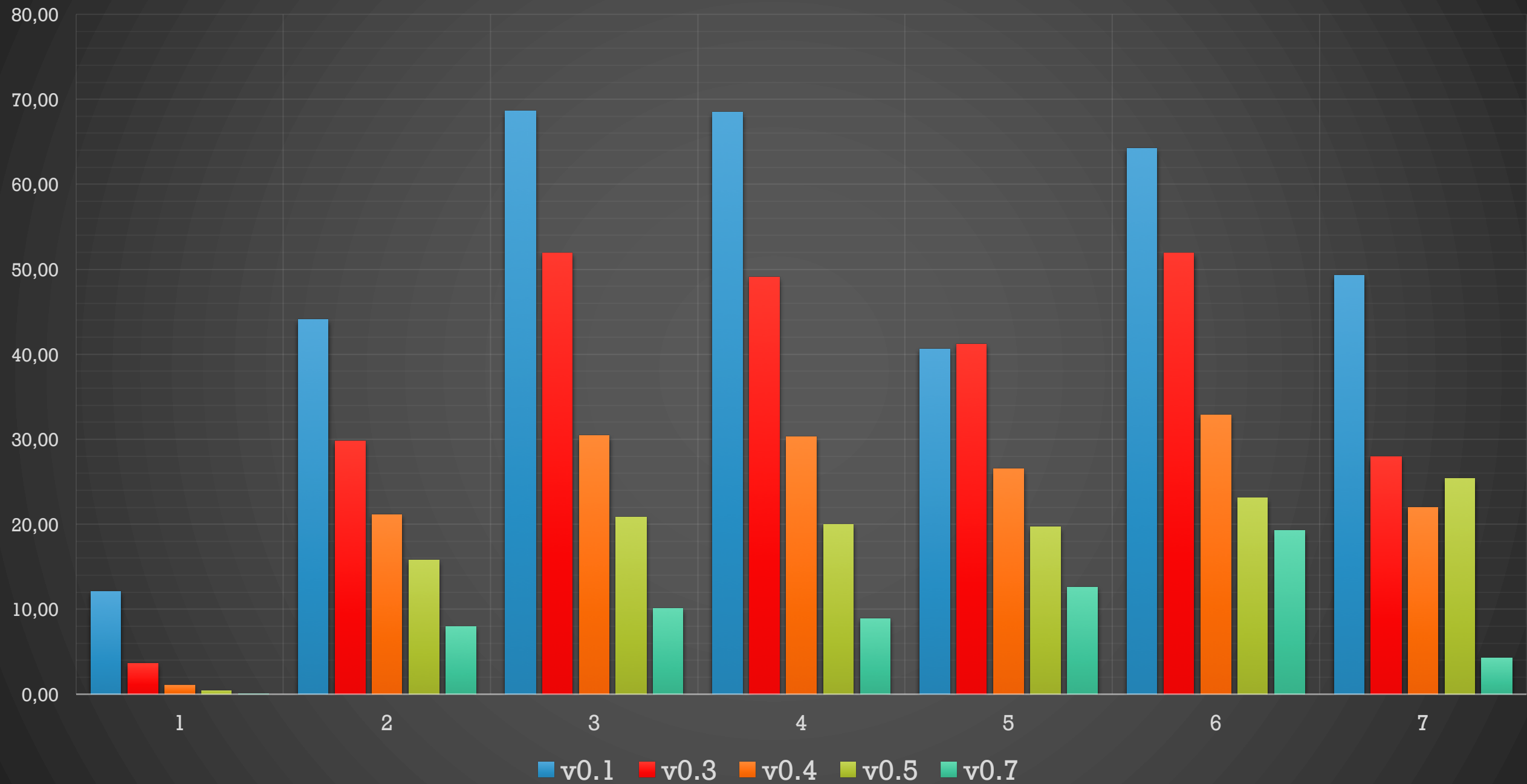
Improving over  
time



# Benchmark

- Our algorithms is flown over 5 different steps:
  1. v0.1: Tabu List search
  2. v0.3: Greedy Preparation at begin
  3. v0.4: Adding Simulated Annealing and local search
  4. v0.5: Implementing Local Swap
  5. v0.6: Various Tuning
  
- In the next graph is possible to view the benchmark results at each step.

**Benchmark % gap during development for each instances**



Latest Result

Benchmark	Best	Gap %
157,033	157,145	0,07
34,709	37,462	7,93
32,627	35,921	10,10
7,717	8,412	9,00
12,901	14,527	12,60
3,045	3,631	19,26
10,050	10,475	4,23
	Best:	0,07
	Avg:	9,03
	Worst:	19,26

Execution Time: 30 minutes

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](https://github.com/Jacopx/OMA_ExamTimeTable)



Thanks for the attention!