Colote Solver

Team 39

Politecnico di Torino

Optimization Methods and Algorithms

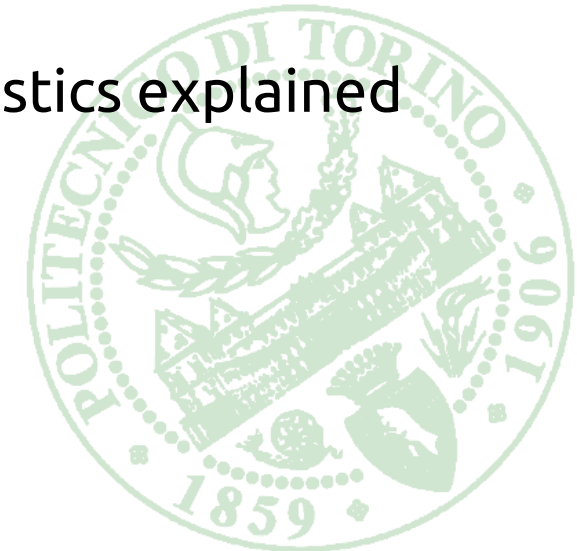
A.Y. 2016/17

The Team

- *Marco Iorio* s239557
- *Paolo Iannino* s239562
- *Aldo Lacuku* s204998
- *Kian Kamyab Masouleh* s226140
- *Mehrnoosh Rigi Kanhari* s236436

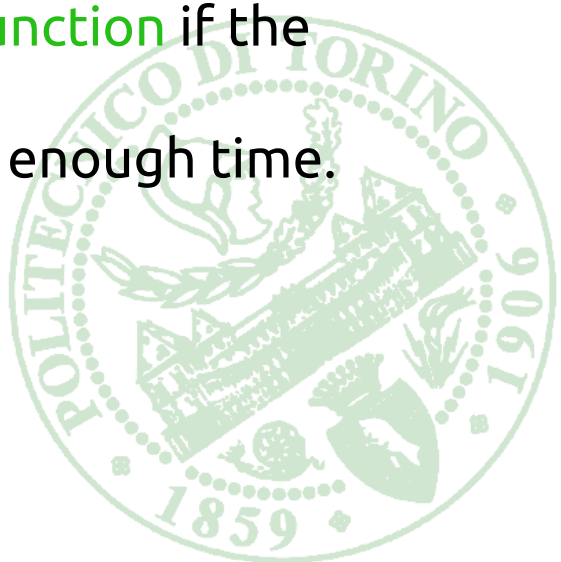
Introduction

- *CoIoTe Solver* is a metaheuristic composed by three main parts:
  - An Inner Heuristic.
  - An Outer Heuristic.
  - An intermediate optimizing step.
- It cannot be associated with any of the heuristics explained during the course.



The *Outer Heuristic*

- It is in charge of doing the following tasks:
  1. Producing a random sequence corresponding to the visiting order of the cells.
  2. Applying the Inner Heuristic on the sequence.
  3. Updating the current value of the Objective Function if the current solution is the best found so far.
  4. Repeating the tasks from point 1 until there is enough time.



The *Greedy*

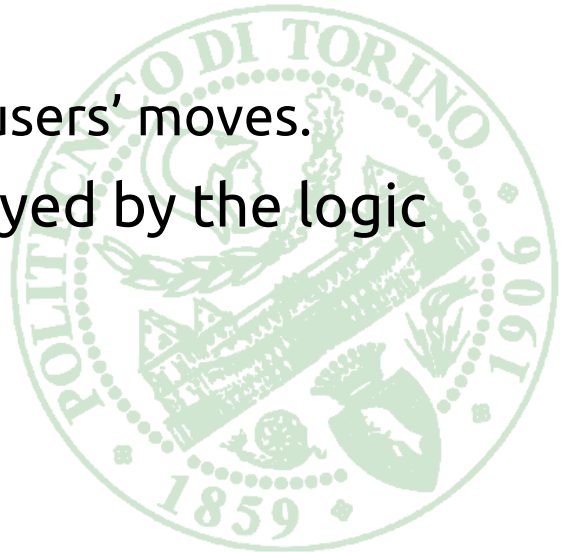
- It is the logical core of our algorithm.
- It takes as input the random sequence generated by the Outer Heuristic
- What it does, on each cell specified by the random sequence, is:
  1. Considering the activities' demand for the given cell j.
  2. Finding the user i, t, m with the minimum cost to move to the cell j.
  3. Assigning an appropiate number of user of type i, t, m to the cell j.
  4. Updating the number of available users and the remaining demand.
  5. Repeating from step 1 until all the requests in cell j are satisfied.

Algorithm's Tuning

- Refinements in the *Greedy*'s logic.
- Parallelization of our program.
- Development of an ad hoc data structure.
- Sorting of the costs.

Further Optimizing Step

- It takes as input the solution computed by the *Greedy*.
- Basic idea: trying to replace some of the previously chosen users with cheaper combinations of them.
- Implementation:
  - Quite sophisticated.
  - Exploits the recursion paradigm.
  - Requires an ad hoc data structure for storing users' moves.
- In its development a remarkable role was played by the logic underlying the *Tabu Search* metaheuristic.



Pseudo-Code #1

```
GREEDY(sequence, available_users, cell_requests, user_activities)

begin body
    for(each_cell_in(sequence)) do
        while(there_are(cell_requests)) do
            cheaper = find_cheaper(available_users)


            n_users = needed(cheaper, available_users, cell_requests)

            available_users -= n_users
            cell_requests -= n_users * user_activities
            current_solution += n_users
        end while
    end for

    return solution;
end body
```

Pseudo-Code #2

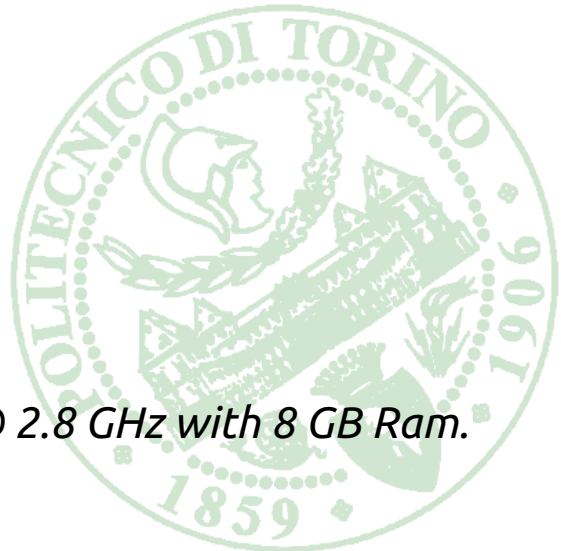
```
for(each_thread) do
    while(still_time) do
        for(ten_times) do
            shuffle(sequence)
            current_solution = GREEDY(sequence, available_users,
                                      cell_requests, user_activities)

            if(current_solution.better_than(solution))
                then solution = current_solution
            end if
        end for

        if(is_feasible(solution))
            then OPTIMIZE(solution)
        end if

        if(is_best(solution))
            then best_solution = solution
        end if
    end while
end for
```

Our Results

- Test set: 128 input files provided by the professor.
- Average execution time: 1.265 seconds
- Percent error:
  - In 124 cases: 0%
  - In 4 cases: ⩽ 0.1%

OPTIMUM IN 97% OF CASES!

- Maximum percent error: 0.064%
- Average percent error: 0.0014%



Tested on Debian Stretch, Intel i7-860 @ 2.8 GHz with 8 GB Ram.

Final Observations

- For the given problem, a *Greedy* approach combined with a source of entropy and an efficient testing environment leads to far better results compared to standard heuristics.

A Free Software

- *CoIoTe Solver* is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
- *CoIoTe Solver* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
- You should have received a copy of the GNU General Public License along with *CoIoTe Solver*. If not, see <http://www.gnu.org/licenses/>.
- You can download *CoIoTe Solver* at the following link:

https://github.com/giorio94/CoIoTeSolver

Thanks for listening!