UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Course Project

ALGORITHMIC METHODS FOR MATHEMATICAL MODELS

December 10, 2024

Àlex Font Mercadié *
Carles Matoses Gimenez †

*Email: alex.font@estudiantat@upc.edu
†Email: carles.matoses@estudiantat@upc.edu

Contents

1 Statement of the problem **2**

2 ILP Formulation **2**

3 Meta-Heuristics **4**

3.1 Feasibility 4

3.2 Committee members' weight 4

3.3 Greedy Constructive Algorithm 5

3.4 Local Search 6

3.5 GRASP algorithm 7

4 Results **8**

1 Statement of the problem

The *Committee* problem can be formally stated as follows:

Given:

- The set of D departments of the university indexed by $\{1, \ldots, D\}$.
- The set of N faculty members indexed by $\{1, \ldots, N\}$. For each member i its department d_i is specified.
- The compatibility m_{ij} between every pair of members i and j.

Find the commission of members of the faculty subject to the following constraints:

- Each department p has exactly n_p members in the commission.
- There cannot be two members i and j in the commission with compatibility $m_{ij} = 0$.
- For each pair of members i and j in the commission with a compatibility lower than 0.15 there must be a member k in the commission with compatibility higher than 0.85 with both members i and j.

with the *objective* to maximize the average compatibility among all pairs of participants in the commission.

2 ILP Formulation

The *Committee* problem can be modeled as an Integer Linear Program. To this end, the following sets and parameters are defined:

N	Total number of faculty members, index i.
D	Total number of faculty departments, index p.
n_p	Number of members of department p required in the commission.
d_i	Department to which member i belongs.
m_{ij}	Compatibility between members i and j.

The following decision variables are also defined:

c_i	Binary variable representing if faculty member i is selected to be part of the committee.
x_{ij}	Binary variable representing if both faculty member i and j are selected to be part of the committee.

Finally, the LP model for the committee problem is as follows:

maximize

$$\sum_{i=1}^{N} \sum_{j=i+1}^{N} m_{ij} \cdot x_{ij}$$

subject to:

$$x_{ij} \leq c_i \qquad \forall i,j \in \{1,\dots,N\} \tag{1}$$
$$x_{ij} \leq c_j \qquad \forall i,j \in \{1,\dots,N\} \tag{2}$$
$$x_{ij} \geq c_i + c_j - 1 \qquad \forall i,j \in \{1,\dots,N\} \tag{3}$$
$$c_i + c_j \leq 1 \qquad \forall i,j \in \{1,\dots,N\} \text{ s.t. } m_{ij} = 0 \tag{4}$$
$$1 - x_{ij} + \sum_{\substack{1\leq k\leq N,\\ m_{ik},m_{jk}>0.85}} c_k \geq 1 \qquad \forall i,j \in \{1,\dots,N\} \text{ s.t. } m_{ij} < 0.15 \tag{5}$$
$$\sum_{\substack{1\leq i\leq N\\ d_i=p}} c_i = n_p \qquad \forall p \in \{1,\dots,D\} \tag{6}$$

Constraint 4 ensures that no two members with zero compatibility are both in the commission.
Constraint 5 enforces the condition that if members i and j with $m_{ij} < 0.15$ are in the commission then there is another member k in the commission such that $m_{ik} > 0.85$ and $m_{jk} > 0.85$.
Constraint 6 guarantees that every department has the desired number of members in the commission.
Finally, constraints 1, 2 and 3 relate variables x_{ij} with c_i and c_j in order to have $x_{ij} = 1 \iff c_i = 1 \wedge c_j = 1$.

The following graph 1 shows how the solutions require more time if the number of members or the committee size increases. For the same number of members and the same committee size, the execution time decreases the bigger D becomes 1. For a constant D and a constant N, the committee size will increases the complexity of the problem 2.

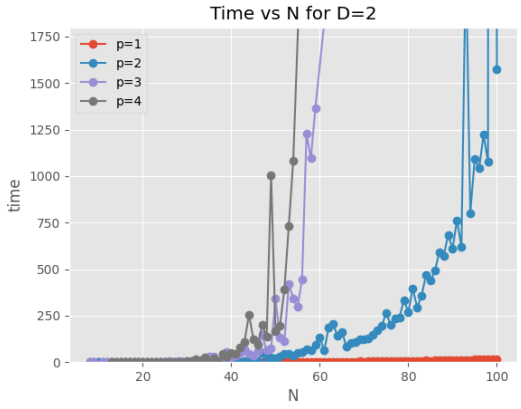

Figure 1: Time required for different values of p

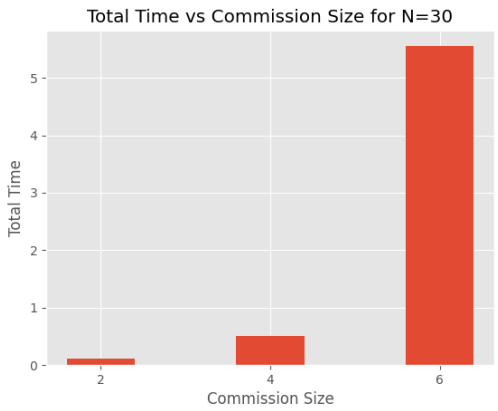

Figure 2: Execution time for instances of D=2 and N=30

N	D	p	size	time
30	2	4	8	3.47
30	4	2	8	0.56
54	2	4	8	1083.16
54	4	2	8	309.06

Table 1: Department Size Increases Complexity

3 Meta-Heuristics

3.1 Feasibility

In the context of the committee problem, a feasible solution is one that satisfies all the constraints. The following function can be used to check the feasibility of a given solution:

Algorithm 1: isFeasible(C)

Input: A set C of faculty members.
foreach *department* $p \in \{1, \ldots, D\}$ **do**
  if *number of members from department* p *in* $C \neq n_p$ **then**
    return false
foreach *pair of members* $i, j \in C$ **do**
  if $m_{ij} = 0$ **then**
    return false
foreach *pair of members* $i, j \in C$ **do**
  if $m_{ij} < 0.15$ **then**
    if *there is no member* $k \in C$ *s.t.* $m_{ik} > 0.85$ *and* $m_{jk} > 0.85$ **then**
      return false

This can be extended to the selection of new members where the candidate is checked against the current C set to determine if it violates any of the constraints. This is not the best approach since we can not verify if the last constraint is satisfied until the entire solution has been completed.

3.2 Committee members' weight

The weight of a faculty member i with respect to a set C of faculty members can be computed as 7:

$$w(i, C) = \sum_{j \in C} m_{ij} \tag{7}$$

For the committee C, the objective function can be computed as the average compatibility among all pairs of members in the committee 8:

$$\text{average\_compatibility}(C) = \frac{1}{|C|} \sum_{i \in C} \sum_{j \in C,\ i<j} m_{ij} \tag{8}$$

These functions, although simple in structure, pose certain challenges for implementing this project. The constraints go beyond merely determining which members to select; they also encompass more complex requirements, such as ensuring compatibility between

members with a value less than 0.15, which can only be verified after the committee is formed.

The new weight function, $w(i, C)$, is designed to evaluate a candidate i by considering its relationship with both the members already in the committee C and those not yet selected ($\neg C$). Unlike strict constraint enforcement, this function introduces a relaxed scoring mechanism. It penalizes candidates who break the constraints by reducing their score.

As a result, candidates that fully respect the constraints will achieve higher scores, while those that introduce conflicts or fail to meet intermediate requirements will incur penalties, reducing their overall weight. This approach allows the algorithm to explore a wider solution space, finding feasible candidates even in highly constrained scenarios, while still prioritizing those that align best with the problem's requirements.

The following formula takes into account the relationship of the current committee members and the candidate i:

$$w_1(i, C) = \sum_{j \in C} \begin{cases} 2m_{ij}, & \text{if } 0.15 \leq m_{ij} \leq 0.85 \text{ or existsIntermediate}(i, j, C), \\ -2, & \text{if } m_{ij} < 0.15 \text{ and no intermediate exists.} \end{cases} \tag{9}$$

For further refinement, the algorithm will also take into account the future candidates members not in the committee:

$$w_2(i, C) = \sum_{j \notin C} \begin{cases} m_{ij}, & \text{if } 0.15 \leq m_{ij} \text{ or existsIntermediate}(i, j, C) \text{ and isValid}(j, C \cup \{i\}), \\ -1, & \text{if } m_{ij} < 0.15 \text{ and no intermediate exists, and isValid}(j, C \cup \{i\}). \end{cases} \tag{10}$$

The weight function will become the addition of both parts:

$$w(i, C) = w_1(i, C) + w_2(i, C) \tag{11}$$

3.3 Greedy Constructive Algorithm

A purely greedy algorithm will produce a deterministic solution, will suffer from myopia, and possibly fail to find a solution for constrained problems. The algorithm will take into account the constraints 4 and 6. In the other hand, it will use the w function to determine if members with low compatibility are relevant for the final solution. To formulate the problem on an organized manner we will divide it in four steps: Input variables, Initialized variables, Iteratively add new members to the committee, Validate solution.

phase 1

These are the variables provided by the *.dat* file.

N: Total number of members
D: Total number of departments
n: Array where n_p is the required members capacity of department p ($1 \leq p \leq D$)
d: Array where d_i is the department of member i ($1 \leq i \leq N$)
m: Compatibility matrix where m_{ij} represents compatibility between members i and j

phase 2

The auxiliary sets of variables required are C as the committee selected members and $members$ as the set of available members.

Initialize $C \leftarrow \emptyset$
Initialize $members \leftarrow \{1, \ldots, N\}$

phase 3

The following algorithm will provide the best candidate in each iteration respecting constrains 4 and 6.

while *C not a solution* **do**
    $feasible\_members \leftarrow \{\}$
    $members\_weights \leftarrow \{\}$
    foreach *$member \notin C$* **do**
        if *isFeasible(member, C)* **then**
            $feasible\_members \leftarrow feasible\_members \cup \{member\}$
            $members\_weights[member] \leftarrow w(member, C)$
    if $|feasible\_members| > 0$ **then**
        $best\_member \leftarrow \arg\max_{m \in feasible\_members} members\_weights[m]$
        $C \leftarrow C \cup \{best\_member\}$

phase 4

To check if the solution is feasible, we can check the not computed constraints.

foreach *pair of members $(i, j) \in C \times C$* **do**
    if $m_{ij} < 0.15$ **then**
        if *there does not exist a $k \in C$ such that $m_{ik} > 0.85$ and $m_{jk} > 0.85$* **then**
            Print "Relaxed solution is not valid for the original problem"
            return *C // Relaxed solution to be improved later*

3.4 Local Search

The function of a local search algorithm aims to improve a given solution, in this case the committee C generated by the greedy algorithm. The neighbor solution compatibility

is compared to C's compatibility to check for improvement. $isFeasible$ checks if the provided solution respects all constraints.

```
C ← current_solution
Comp ← average_compatibility(C)
foreach i ∈ {1, ..., N} do
    if i ∈ C then
        C_neighbor ← C
        C_neighbor ← remove(C_neighbor, i)
        foreach j ∈ {1, ..., N} do
            if j ∉ C_neighbor then
                C_neighbor ← add(C_neighbor, j)
                if isFeasible(C_neighbor) then
                    Comp_neighbor ← average_compatibility(C_neighbor)
                    if Comp_neighbor > Comp then
                        Comp ← Comp_neighbor
                        C ← C_neighbor
```

3.5 GRASP algorithm

The Greedy Randomized Adaptive Search introduces a randomized approach for selecting the new candidates for the committee. First a solution similar to the greedy algorithm is used. Instead of selecting the best suited member as the new committee integrant, a random candidate from the best options is chosen. After that, it improves the solution found with local search as before. Executing this multiple times will find different solutions where the best one is kept.

Taking the greedy algorithm as the starting point, the restricted candidate list 12 will be introduced. C will represent the committee current selected members. The function $w(n, C)$ will return the computed weight of the member against the current committee members. $0 \leq \alpha \leq 1$ determines the the minimum weight of of a member to belong to the RCL set where 1 means a random selection and 0 the most scored member.

$$RCL_{\max} = \{n \in \{1, \ldots, N\} \mid W_n \geq W_{\max} - \alpha \cdot (W_{\max} - W_{\min})\} \tag{12}$$

```
C ← ∅
Candidates ← {1, ..., N}
while C not a solution do
    W ← [0, 0, ..., 0] of size N
    foreach n ∈ Candidates do
        W_n ← w(n, C)
    W_min ← min{W_n | n ∈ Candidates}
    W_max ← max{W_n | n ∈ Candidates}
    RCL_max ← {n ∈ Candidates | W_n ≥ W_max − α(W_max − W_min)}
    Select n ∈ RCL at random
    C ← C ∪ {n}
    Candidates ← Candidates \ {n}
```

To chose the best value for the parameter α we run multiple instances and look at wich

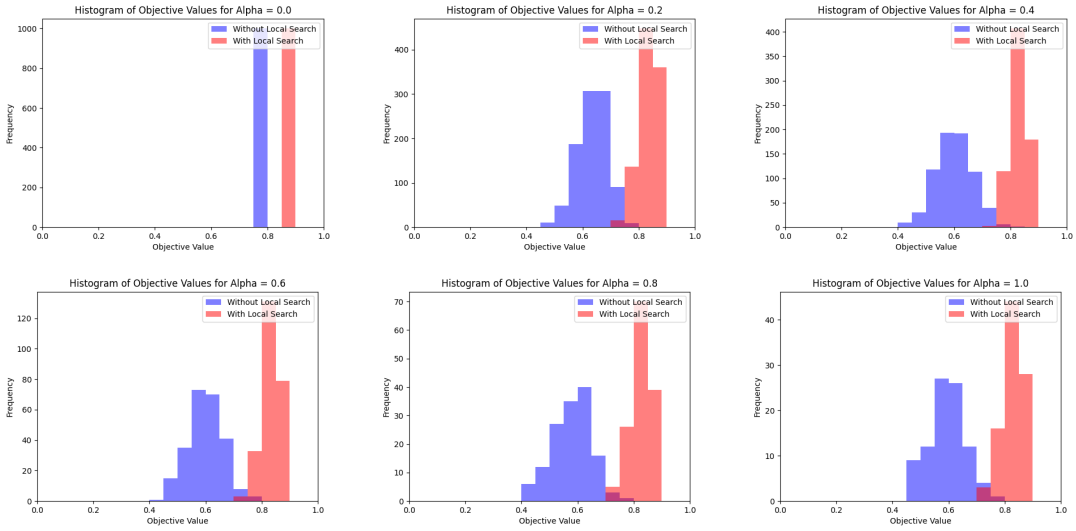

Figure 3: Distribution of construction phase solution values as a function of the RCL parameter α for one instance.

values lead to better results. In 3 we can observe that, except for $\alpha = 0$, we get similar distribution of the objective values, but as we increase the value of α the constructive phase finds less often a solution and we are not improving the objective.

Computing this average for multiple instances with different sizes and we can make a decision. In our case, we conclude that a value of $\alpha = 0.2$ has enough variability and produces the best results.

4 Results

The accuracy of the objectives increases with the use of a local search. The best results are obtained using GRASP together with local search and "$FirstImprovement$" deactivated. The images 4 show that GRASP has the same accuracy as Cplex. The most notable differences occur on really complex situations like the one shown on committee size of 10 with really low number of members (less than 20). This occurs because there is only one solution, which is hard to find with GRASP algorithm or deterministic approaches.

The following images 5 show the accuracy with local search on the GRASP algorithm. The required time also increases, but as shown before, with 60 seconds is enough to obtain the optimal committee.

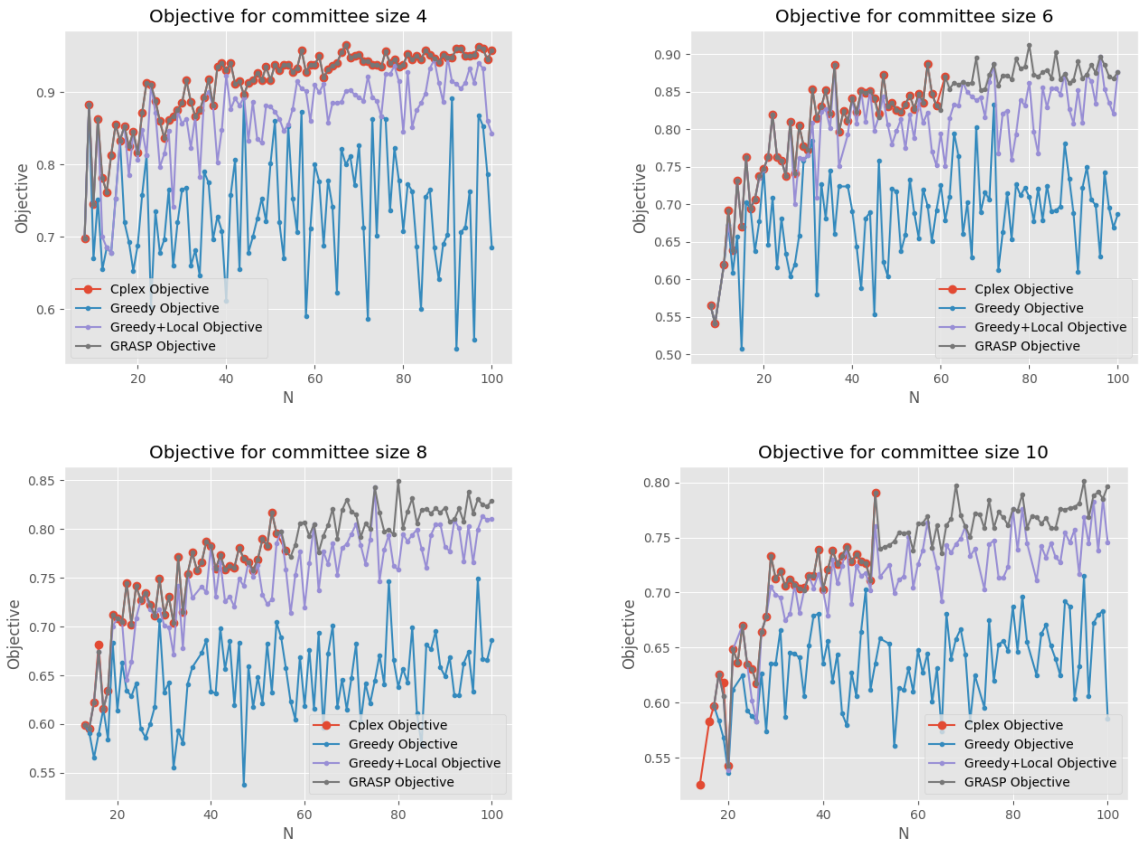

Figure 4: Objective accuracy of GRASP, local search and greedy algorithm

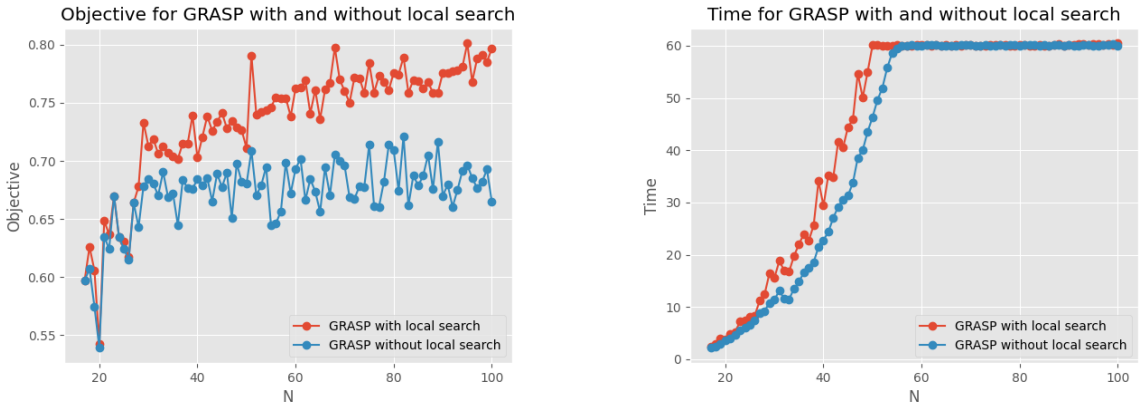

Figure 5: Grasp time and objective comparison with local search activated