

# Project AMMM: Selecting the best committee

*Authors:*

**Alex Herrero**  
**Lluna Clavera**

*Professors:*

**Enric Rodríguez-Carbonell**  
**Luís Domingo Velasco**



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Facultat d'Informàtica de Barcelona



# Table of Contents

- 1 The problem
- 2 Problem Formulation
  - Alternative Formulation
  - Fixing to ILP formulation
- 3 Heuristics
  - Greedy
  - Local Search
  - GRASP
- 4 Experimentation and results
- 5 Conclusions

- 1 The problem
- 2 Problem Formulation
  - Alternative Formulation
  - Fixing to ILP formulation
- 3 Heuristics
  - Greedy
  - Local Search
  - GRASP
- 4 Experimentation and results
- 5 Conclusions

Variable	Meaning	Range	Type
$N$	Number of members of faculty	Integer	Integer
$D$	Number of departments in the faculty	Integer	Integer
$d_i$	Department of professor $i$	$1 \leq i \leq N$	Integer
$n_p$	Number of people needed from department $p$	$1 \leq p \leq D$	Integer
$m_{ij}$	Compatibility between professor $i$ and $j$	$0 \leq m_{ij} \leq 1, 1 \leq i, j \leq N$	Real

Table: Input of the problem

- Recall that we are dealing with the problem of assigning professor to committees

Variable	Meaning	Range	Type
$N$	Number of members of faculty	Integer	Integer
$D$	Number of departments in the faculty	Integer	Integer
$d_i$	Department of professor $i$	$1 \leq i \leq N$	Integer
$n_p$	Number of people needed from department $p$	$1 \leq p \leq D$	Integer
$m_{ij}$	Compatibility between professor $i$ and $j$	$0 \leq m_{ij} \leq 1, 1 \leq i, j \leq N$	Real

Table: Input of the problem

- Recall that we are dealing with the problem of assigning professor to committees
- We consider that a subset  $S \subseteq \{1, \dots, N\}$  is a solution if:

Variable	Meaning	Range	Type
$N$	Number of members of faculty	Integer	Integer
$D$	Number of departments in the faculty	Integer	Integer
$d_i$	Department of professor $i$	$1 \leq i \leq N$	Integer
$n_p$	Number of people needed from department $p$	$1 \leq p \leq D$	Integer
$m_{ij}$	Compatibility between professor $i$ and $j$	$0 \leq m_{ij} \leq 1, 1 \leq i, j \leq N$	Real

Table: Input of the problem

- Recall that we are dealing with the problem of assigning professor to committees
- We consider that a subset  $S \subseteq \{1, \dots, N\}$  is a solution if:
  - $|\{i \in S \mid d_i = p\}| = n_p$  for every  $1 \leq p \leq D$

Variable	Meaning	Range	Type
$N$	Number of members of faculty	Integer	Integer
$D$	Number of departments in the faculty	Integer	Integer
$d_i$	Department of professor $i$	$1 \leq i \leq N$	Integer
$n_p$	Number of people needed from department $p$	$1 \leq p \leq D$	Integer
$m_{ij}$	Compatibility between professor $i$ and $j$	$0 \leq m_{ij} \leq 1, 1 \leq i, j \leq N$	Real

Table: Input of the problem

- Recall that we are dealing with the problem of assigning professor to committees
- We consider that a subset  $S \subseteq \{1, \dots, N\}$  is a solution if:
  - $|\{i \in S \mid d_i = p\}| = n_p$  for every  $1 \leq p \leq D$
  - $\forall i, j \in S, m_{ij} \neq 0$

Variable	Meaning	Range	Type
$N$	Number of members of faculty	Integer	Integer
$D$	Number of departments in the faculty	Integer	Integer
$d_i$	Department of professor $i$	$1 \leq i \leq N$	Integer
$n_p$	Number of people needed from department $p$	$1 \leq p \leq D$	Integer
$m_{ij}$	Compatibility between professor $i$ and $j$	$0 \leq m_{ij} \leq 1, 1 \leq i, j \leq N$	Real

Table: Input of the problem

- Recall that we are dealing with the problem of assigning professor to committees
- We consider that a subset  $S \subseteq \{1, \dots, N\}$  is a solution if:
  - $|\{i \in S \mid d_i = p\}| = n_p$  for every  $1 \leq p \leq D$
  - $\forall i, j \in S, m_{ij} \neq 0$
  - $\exists i, j \in S, m_{ij} < 0.15 \implies \exists k \in S, m_{ik} > 0.85 \text{ and } m_{jk} > 0.85$



Variable	Meaning	Range	Type
$N$	Number of members of faculty	Integer	Integer
$D$	Number of departments in the faculty	Integer	Integer
$d_i$	Department of professor $i$	$1 \leq i \leq N$	Integer
$n_p$	Number of people needed from department $p$	$1 \leq p \leq D$	Integer
$m_{ij}$	Compatibility between professor $i$ and $j$	$0 \leq m_{ij} \leq 1, 1 \leq i, j \leq N$	Real

Table: Input of the problem

- Recall that we are dealing with the problem of assigning professor to committees
- We consider that a subset  $S \subseteq \{1, \dots, N\}$  is a solution if:
  - $|\{i \in S \mid d_i = p\}| = n_p$  for every  $1 \leq p \leq D$
  - $\forall i, j \in S, m_{ij} \neq 0$
  - $\exists i, j \in S, m_{ij} < 0.15 \implies \exists k \in S, m_{ik} > 0.85 \text{ and } m_{jk} > 0.85$
- Given  $f(S) = \frac{2}{|S| \cdot (|S|-1)} \sum_{i \in S} \sum_{\substack{j \in S \\ i < j}} m_{ij}$ , find a solution  $S^*$  such that  $f(S^*) \geq f(S)$  for all possible solutions  $S$

- 1 The problem
- 2 Problem Formulation
  - Alternative Formulation
  - Fixing to ILP formulation
- 3 Heuristics
  - Greedy
  - Local Search
  - GRASP
- 4 Experimentation and results
- 5 Conclusions

# Outline

- 1 The problem
- 2 Problem Formulation
  - Alternative Formulation
  - Fixing to ILP formulation
- 3 Heuristics
  - Greedy
  - Local Search
  - GRASP
- 4 Experimentation and results
- 5 Conclusions



Variable	Meaning	Range	Type
$x_i$	Professor $i$ goes to the committee	$1 \leq i \leq N$	Boolean

**Objective Function:**

$$\text{Maximize: } \frac{2}{n \cdot (n - 1)} \sum_{i=1}^N \sum_{j=i+1}^N m_{ij} \cdot x_i \cdot x_j$$

Variable	Meaning	Range	Type
$x_i$	Professor $i$ goes to the committee	$1 \leq i \leq N$	Boolean

**Objective Function:**

$$\text{Maximize: } \frac{2}{n \cdot (n-1)} \sum_{i=1}^N \sum_{j=i+1}^N m_{ij} \cdot x_i \cdot x_j$$

**Subject to:**

$$\text{Number of participants per department: } \sum_{i \in A_p} x_i = n_p \text{ for every } 1 \leq p \leq D$$

$$\text{No two professors with 0 compatibility: } \lceil m_{ij} \rceil \geq x_i \cdot x_j \text{ for every } 1 \leq i < j \leq N$$

$$\text{Not enough trust: } \sum_{k \in W_{ij}} x_k \geq x_i \cdot x_j \text{ for every } (i, j) \in N_{ij}$$

**Where:**

$$A_d = \{i \in \{1, \dots, N\} : d_i = d\}$$

$$W_{ij} = \{k \in \{1, \dots, N\} : m_{ik} > 0.85 \wedge m_{jk} > 0.85\}$$

$$N_{ij} = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 1 \leq i < j \leq N \wedge m_{ij} < 0.15\}$$

Variable	Meaning	Range	Type
$x_i$	Professor $i$ goes to the committee	$1 \leq i \leq N$	Boolean

**Objective Function:**

$$\text{Maximize: } \frac{2}{n \cdot (n-1)} \sum_{i=1}^N \sum_{j=i+1}^N m_{ij} \cdot x_i \cdot x_j$$

**Subject to:**

$$\text{Number of participants per department: } \sum_{i \in A_p} x_i = n_p \text{ for every } 1 \leq p \leq D$$

$$\text{No two professors with 0 compatibility: } \lceil m_{ij} \rceil \geq x_i \cdot x_j \text{ for every } 1 \leq i < j \leq N$$

$$\text{Not enough trust: } \sum_{k \in W_{ij}} x_k \geq x_i \cdot x_j \text{ for every } (i, j) \in N_{ij}$$

**Where:**

$$A_d = \{i \in \{1, \dots, N\} : d_i = d\}$$

$$W_{ij} = \{k \in \{1, \dots, N\} : m_{ik} > 0.85 \wedge m_{jk} > 0.85\}$$

$$N_{ij} = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 1 \leq i < j \leq N \wedge m_{ij} < 0.15\}$$

**This is Non-Linear Programming!** Can we fix it?

# Outline

- 1 The problem
- 2 Problem Formulation
  - Alternative Formulation
  - Fixing to ILP formulation
- 3 Heuristics
  - Greedy
  - Local Search
  - GRASP
- 4 Experimentation and results
- 5 Conclusions



Variable	Meaning	Range	Type
$x_i$	Professor $i$ goes to the committee	$1 \leq i \leq N$	Boolean

### Objective Function:

$$\text{Maximize: } \frac{2}{n \cdot (n-1)} \sum_{i=1}^N \sum_{j=i+1}^N m_{ij} \cdot x_i \cdot x_j$$

### Subject to:

$$\text{Number of participants per department: } \sum_{i \in A_p} x_i = n_p \text{ for every } 1 \leq p \leq D$$

$$\text{No two professors with 0 compatibility: } \lceil m_{ij} \rceil \geq x_i \cdot x_j \text{ for every } 1 \leq i < j \leq N$$

$$\text{Not enough trust: } \sum_{k \in W_{ij}} x_k \geq x_i \cdot x_j \text{ for every } (i, j) \in N_{ij}$$

### Where:

$$A_d = \{i \in \{1, \dots, N\} : d_i = d\}$$

$$W_{ij} = \{k \in \{1, \dots, N\} : m_{ik} > 0.85 \wedge m_{jk} > 0.85\}$$

$$N_{ij} = \{(i, j) \in \mathbb{N} \times \mathbb{N} : 1 \leq i < j \leq N \wedge m_{ij} < 0.15\}$$

Variable	Meaning	Range	Type
$x_i$	Professor $i$ goes to the committee	$1 \leq i \leq N$	Boolean
$y_{ij}$	Professor $i$ and $j$ go to the committee	$1 \leq i, j \leq N$	Boolean

### Objective Function:

$$\text{Maximize: } \frac{2}{n \cdot (n-1)} \sum_{i=1}^N \sum_{j=i+1}^N m_{ij} \cdot y_{ij}$$

### Subject to:

Number of participants per department:  $\sum_{i \in A_p} x_i = n_p$  for every  $1 \leq p \leq D$

No two professors with 0 compatibility:  $\lceil m_{ij} \rceil \geq x_i + x_j - 1$  for every  $1 \leq i < j \leq N$

Not enough trust:  $\sum_{k \in W_{ij}} x_k \geq x_i + x_j - 1$  for every  $(i, j) \in N_{ij}$

Correlation between variables:  $\begin{cases} x_i \geq y_{ij} \\ x_j \geq y_{ij} \end{cases}$

- 1 The problem
- 2 Problem Formulation
  - Alternative Formulation
  - Fixing to ILP formulation
- 3 **Heuristics**
  - Greedy
  - Local Search
  - GRASP
- 4 Experimentation and results
- 5 Conclusions

# Outline

- 1 The problem
- 2 Problem Formulation
  - Alternative Formulation
  - Fixing to ILP formulation
- 3 **Heuristics**
  - **Greedy**
  - Local Search
  - GRASP
- 4 Experimentation and results
- 5 Conclusions

- The greedy algorithm is useful to obtain fast solutions

- The greedy algorithm is useful to obtain fast solutions
- It chooses elements one by one without backtracking

- The greedy algorithm is useful to obtain fast solutions
- It chooses elements one by one without backtracking
- The decisions are based on a *greedy function*

- We use the *greedy function*:



- We use the *greedy function*:  $q(u) = \sum_{s \in S} m_{us}$

- We use the *greedy function*:  $q(u) = \sum_{s \in S} m_{us}$
- We then select the element  $u$  with higher  $q(u)$  at each step

- We use the *greedy function*:  $q(u) = \sum_{s \in S} m_{us}$
- We then select the element  $u$  with higher  $q(u)$  at each step
- It does not make sense to evaluate  $q$  for the first element, we treat it as an special case

- We use the *greedy function*:  $q(u) = \sum_{s \in S} m_{us}$
- We then select the element  $u$  with higher  $q(u)$  at each step
- It does not make sense to evaluate  $q$  for the first element, we treat it as a special case
- The first element will be the professor with the highest mean compatibility

- We use the *greedy function*:  $q(u) = \sum_{s \in S} m_{us}$
- We then select the element  $u$  with higher  $q(u)$  at each step
- It does not make sense to evaluate  $q$  for the first element, we treat it as an special case
- The first element will be the professor with the highest mean compatibility
- The rest of the algorithm is the usual greedy algorithm

```
function GREEDYSOLVE( $N, D, d, n, m$ )  
   $remaining \leftarrow sum(n)$   
   $S \leftarrow \emptyset$   
   $S \leftarrow S \cup \{bestCompat(N, m)\}$   
   $remaining \leftarrow remaining - 1$   
  
  while  $remaining > 0$  do  
     $U \leftarrow feasibleProfs(N, D, d, n, m, S)$   
    if  $U = \emptyset$  then  
      return null  
    else  
       $p \leftarrow \arg \max_{u \in U} \{q(u)\}$   
       $S \leftarrow S \cup \{p\}$   
       $n_{d_p} \leftarrow n_{d_p} - 1$   
       $remaining \leftarrow remaining - 1$   
    end if  
  end while  
  return  $S$   
end function
```

# Outline

- 1 The problem
- 2 Problem Formulation
  - Alternative Formulation
  - Fixing to ILP formulation
- 3 **Heuristics**
  - Greedy
  - **Local Search**
  - GRASP
- 4 Experimentation and results
- 5 Conclusions

- Local Search (LS) is used to improve solutions



- Local Search (LS) is used to improve solutions
- Given an initial solution and an operator, LS will search in  $\mathcal{N}(S)$  for better solutions

- Local Search (LS) is used to improve solutions
- Given an initial solution and an operator, LS will search in  $\mathcal{N}(S)$  for better solutions
- The algorithm stops upon finding  $S^*$  such as
$$f(S^*) \geq f(s) \quad \forall s \in \mathcal{N}(S^*)$$

- Local Search (LS) is used to improve solutions
- Given an initial solution and an operator, LS will search in  $\mathcal{N}(S)$  for better solutions
- The algorithm stops upon finding  $S^*$  such as  $f(S^*) \geq f(s) \quad \forall s \in \mathcal{N}(S^*)$
- We used a Swap Operator with *first improvement strategy*

```
function LOCALSEARCH( $N, D, d, n, m, S$ )  
     $(i, j) \leftarrow \text{findFeasibleSwap}(N, D, d, n, m, S)$   
    while  $(i, j)$  is not null do  
         $S \leftarrow (S \setminus \{i\}) \cup \{j\}$   
         $(i, j) \leftarrow \text{findFeasibleSwap}(N, D, d, n, m, S)$   
    end while  
    return  $S$   
end function
```

```
function FINDFEASIBLESWAP( $N, D, d, n, m, S$ )  
  for  $i = 1, \dots, N$  do  
    for  $j = i + 1, \dots, N$  do  
      if validSwap( $N, D, d, n, m, S, i, j$ ) then  
        outside  $\leftarrow$  from  $i, j$  the one that is not in  $S$   
        inside  $\leftarrow$  from  $i, j$  the one that is in  $S$   
        out  $\leftarrow 0$   
        in  $\leftarrow 0$   
        for  $k = 1, \dots, N$  do  
          if  $k \in S$  and  $k \neq \text{inside}$  then  
            out  $\leftarrow$  out +  $m_{\text{outside},k}$   
            in  $\leftarrow$  in +  $m_{\text{inside},k}$   
          end if  
        end for  
        if out > in then  
          return (inside, outside)  
        end if  
      end if  
    end for  
  end for  
  return null  
end function
```

# Outline

- 1 The problem
- 2 Problem Formulation
  - Alternative Formulation
  - Fixing to ILP formulation
- 3 **Heuristics**
  - Greedy
  - Local Search
  - **GRASP**
- 4 Experimentation and results
- 5 Conclusions

- Randomize the construction of the initial solution for the LS

- Randomize the construction of the initial solution for the LS
- LS can be repeated with different starting points



- Randomize the construction of the initial solution for the LS
- LS can be repeated with different starting points
- Use a *Greedy Function*  $q$  and evaluate all elements accordingly

- Randomize the construction of the initial solution for the LS
- LS can be repeated with different starting points
- Use a *Greedy Function*  $q$  and evaluate all elements accordingly
- Create a Restricted Candidate List (RCL) with all elements  $u$  such as  $q(u) \geq q_{max} - \alpha(q_{max} - q_{min})$

- Randomize the construction of the initial solution for the LS
- LS can be repeated with different starting points
- Use a *Greedy Function*  $q$  and evaluate all elements accordingly
- Create a Restricted Candidate List (RCL) with all elements  $u$  such as  $q(u) \geq q_{max} - \alpha(q_{max} - q_{min})$
- Select an element of the RCL at random

- Randomize the construction of the initial solution for the LS
- LS can be repeated with different starting points
- Use a *Greedy Function*  $q$  and evaluate all elements accordingly
- Create a Restricted Candidate List (RCL) with all elements  $u$  such as  $q(u) \geq q_{max} - \alpha(q_{max} - q_{min})$
- Select an element of the RCL at random
- Execute the whole process (Constructive Greedy + LS) many times to obtain the best solution

- Same greedy function  $q(u) = \sum_{s \in S} m_{us}$

- Same greedy function  $q(u) = \sum_{s \in S} m_{us}$
- No special selection of the first element

- Same greedy function  $q(u) = \sum_{s \in S} m_{us}$
- No special selection of the first element
- The same local search procedure as before

- Same greedy function  $q(u) = \sum_{s \in S} m_{us}$
- No special selection of the first element
- The same local search procedure as before
- We tested the optimal  $\alpha$  value (next section)



```
function GRASP( $N, D, d, n, m, \alpha, maxIter$ )  
   $S \leftarrow \emptyset$   
  for  $i = 1, \dots, maxIter$  do  
     $U \leftarrow constructiveGreedy(N, D, d, n, m, \alpha)$   
    if  $U \neq \emptyset$  then  
       $U \leftarrow localSearch(N, D, d, n, m, U)$   
    end if  
    if  $f(U) > f(S)$  then           ▷ Check average compatibility  
       $S \leftarrow U$   
    end if  
  end for  
  return  $S$   
end function
```

```
function CONSTRUCTIVEGREEDY( $N, D, d, n, m, \alpha$ )  
  remaining  $\leftarrow$  sum( $n$ )  
   $S \leftarrow \emptyset$   
  while remaining  $> 0$  do  
     $U \leftarrow \text{feasibleProfs}(N, D, d, n, m, S)$   
    if  $U = \emptyset$  then  
      return  $\emptyset$   
    end if  
     $q_{\max} \leftarrow \max_{u \in U} \{q(u)\}$   
     $q_{\min} \leftarrow \min_{u \in U} \{q(u)\}$   
     $\text{rcl} \leftarrow \{u \in U \mid q(u) \geq q_{\max} - \alpha \cdot (q_{\max} - q_{\min})\}$   
     $p \leftarrow$  get random element from rcl  
     $S \leftarrow S \cup \{p\}$   
     $n_{d_p} \leftarrow n_{d_p} - 1$   
    remaining  $\leftarrow$  remaining  $- 1$   
  end while  
  return  $S$   
end function
```

- 1 The problem
- 2 Problem Formulation
  - Alternative Formulation
  - Fixing to ILP formulation
- 3 Heuristics
  - Greedy
  - Local Search
  - GRASP
- 4 Experimentation and results
- 5 Conclusions

# Alpha-tuning

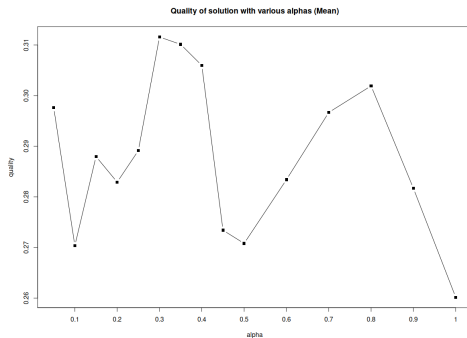


Figure: Mean of objective functions for each alpha

# Alpha-tuning

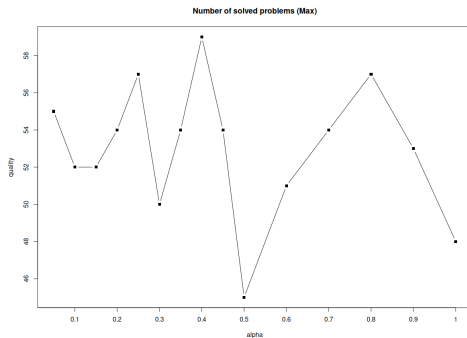


Figure: Number of instances solved by different values of alpha

# Successes and Misses

Algorithm	Success	No solution	Fail
<b>CPLEX</b>	86	14	0
<b>Greedy</b>	36	64	50
<b>Local Search</b>	36	64	50
<b>GRASP First Execution</b>	42	58	44
<b>GRASP Second Execution</b>	43	57	43
<b>GRASP Third Execution</b>	35	65	51
<b>GRASP (Max)</b>	52	48	34
<b>GRASP (Mean)</b>	40	60	46

Table: Success and Failures from different methods

# Greedy Quality

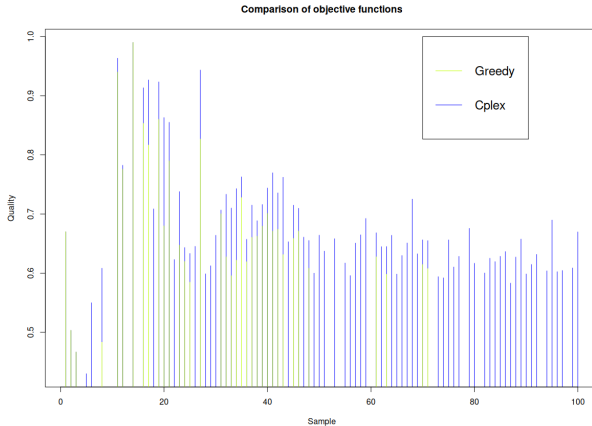


Figure: Greedy Solution

# LS Quality

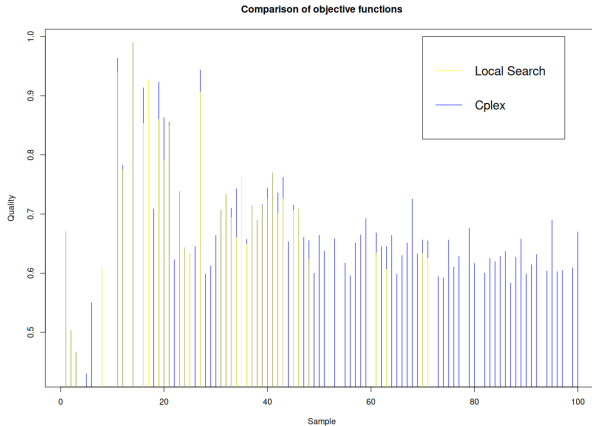


Figure: LS Solution



# GRASP Quality

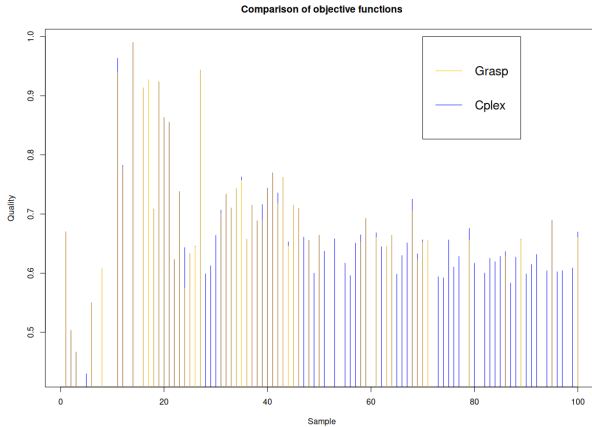


Figure: GRASP Solution

## Quality of heuristics: Global

% Close to optimal solution	<b>Greedy</b>	<b>Local Search</b>	<b>GRASP</b>
100%	3	10	20
> 95%	7	20	31
> 90%	16	5	0
> 85%	5	1	1
< 85%	5	0	0

**Table:** Number of samples that reached certain qualities

# Overall Time

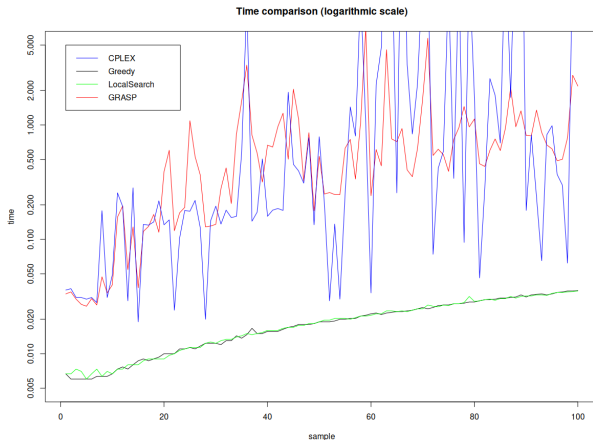


Figure: Overall Time

# CPLEX vs GRASP

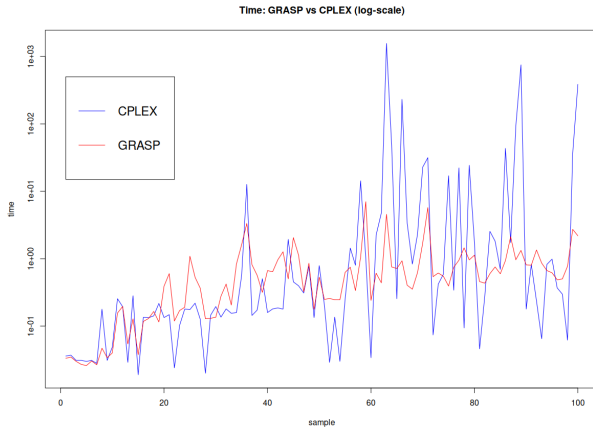


Figure: GRASP vs CPLEX

# Greedy vs LS

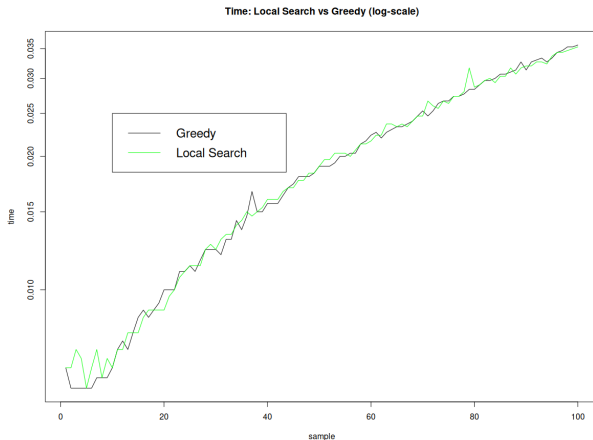


Figure: Greedy vs LS

- 1 The problem
- 2 Problem Formulation
  - Alternative Formulation
  - Fixing to ILP formulation
- 3 Heuristics
  - Greedy
  - Local Search
  - GRASP
- 4 Experimentation and results
- 5 Conclusions

- CPLEX always gives optimal solutions and finds a solution (when it exists), although the execution times could be too high in some instances.

- CPLEX always gives optimal solutions and finds a solution (when it exists), although the execution times could be too high in some instances.
- Heuristics gave us very good solutions (most of the cases being  $> 95\%$  optimal). Unfortunately, it is not guaranteed that they will find a solution when it exists.



- CPLEX always gives optimal solutions and finds a solution (when it exists), although the execution times could be too high in some instances.
- Heuristics gave us very good solutions (most of the cases being  $> 95\%$  optimal). Unfortunately, it is not guaranteed that they will find a solution when it exists.
- Future improvements to find more solutions, maintaining the quality of them, are very promising!.

# Project AMMM: Selecting the best committee

*Authors:*

**Alex Herrero**  
**Lluna Clavera**

*Professors:*

**Enric Rodríguez-Carbonell**  
**Luís Domingo Velasco**



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Facultat d'Informàtica de Barcelona

