POLITECNICO DI TORINO

Master Degree

in MECHATRONIC ENGINEERING

Politecnico
di Torino

Master's Degree Thesis

MATHEURISTICS FOR THE K CLUSTERS WITH FIXED CARDINALITY PROBLEM

Supervisor:
Prof. Marco GHIRARDI

Candidate:
Tommaso BOCCARDO

December 2021

Abstract

The aim of this thesis work is based on the search for a metaheuristic algorithm able to efficiently solve the K clusters with fixed cardinality problems, which are NP-hard combinatorial optimization problems. To do this, the mixed integer linear programming (MILP) formulations that were presented in the study conducted by *Gonclaves & Lourenco, 2019* were used as reference models. Initially, the instances to test were generated randomly using the software MATLAB and the edge graphs present in the CEDRCIC's library, considering the same characteristics as those present in the study. Subsequently the models of the paper were compared. Then an attempt was made to reduce the metaheuristic CPU execution times without straying too far from the optimal solution obtained by the MILP model and promising results have been achieved. These computational experiments were based on LP based local search metaheuristic algorithms. Finally, larger instances were generated, and it was possible to observe that metaheuristics managed not only to improve execution times, but also to obtain better results than the MILP formulation used as a reference. The codes were developed in C++ language and has been written on Visual Studio 2019 using IBM ILOG CPLEX 12.10 as optimization solver.

Summary

As you can see from the title of the thesis, the purpose of this work was to develop a metaheuristics algorithm for the "k cluster with fixed cardinality problem" which is a widespread problem in various applications including Image processing and Pattern Recognition, Market research, Document categorization and Scientific data analysis. It consists in creating groups of items (clusters) with the aim of inserting the most similar elements in the same cluster. The thesis has been divided into four chapters.

The first chapter relates to the introduction where the bases of operational research and its applications are defined and also the description of combinatorial optimization with its practical applications was made. The combinatorial optimization problems are then divided according to their complexity, and it has been observed how the "k cluster with fixed cardinality problem" has been classified among the NP-hard types. The chapter ends with a description of the CPLEX software and the methods used by the solver.

In the second chapter the problem is analysed starting from the results obtained by Gonclaves et Laurenco, 2019 in their study. They faced mixed integer linear programming (MILP) formulations on the problem, with related tests on different types of instances. Subsequently, the two models (F2 and F6), that had obtained the best results in terms of CPU time and number of solved instances, were considered and they were tested on the PC after generating the instances based on the indications in the study. It was observed that the models took longer as instances grew and that the F6 model gave better results. Furthermore, it is possible to observe that the times used for the resolution are lower than those found in the paper and consequently a greater number of instances are resolved.

The third chapter begins with an introduction to metaheuristics which are necessary when the application of the exact methods is not possible for reasons of intrinsic complexity and the time taken to generate the solution is excessive. The study of the algorithm was conducted using Local Search techniques based on the search for a local

optimum starting from an initial solution S0. The first step was to find the starting solution by making a trade-off between the time limit of the solver and the objective function and subsequently an algorithm was generated that at each iteration reduced the number of variables involved. Initially, the algorithm was tested for only two iterations and then for 4 iterations. It was seen that the resolution times were in any case much lower than those found in the MILP of the paper, while reaching values were very close to the objective function of model F6. Given the excellent results obtained on the algorithm, it was decided to test it also on larger instances that were generated in a similar way to the previous ones. Initially, the instances were tested using the model of the paper and it was immediately observed that none reached the optimum in the time limit of 7200 seconds. However, it was observed instead that the algorithm in some instances was able to provide better solutions than those obtained from the F6 model. In the final part of this chapter, it has further modified the algorithm to obtain other improvements.

In the fourth chapter of the thesis the conclusions and the analysis of the results obtained in the tests are reported.

Table of Contents

Chapter 1

Introduction

1.1 Operational Research

The study conducted on this thesis is based on operational research models (OR): it provides a scientific method through mathematical tools to support decision-making activities in which it is necessary to manage and coordinate limited activities and resources to maximize or minimize an objective function [1] [2]. The first studies related to operations research date back to the period of the Second World War to manage complex military operations. In the civil sectors, on the other hand, Operations Research took up techniques known in the industrial sector, improving and enriching them with the use of mathematical tools and organizational knowledge: it dealt with the standardization of production, problems connected with industrial planning and programming. In the United Kingdom, the conversion took place mainly in the public sector, with studies relating to rail, road and urban transport. Some examples of problems that can be addressed by means of Operations Research are the following [3]:

- *Problems in the industrial sectors* such as production planning, optimal stock management, location and sizing of plants.
- *Optimal design problems* which include design of networks and their management in telecommunications, structural design in civil engineering, design of optical systems and design of robots or that responds to pre-established technical requirements by maximizing some parameters related, for example, to accuracy or performance.
- *Problems of economics and finance* related to the choice of investments and portfolio management.
- *Organization problems* such as staff shifts and project planning.
- *Scientific problems* such as image recognition and DNA studies.

- Optimal control problems.

The modelling approach to solve an OR problem or, more generally, the use of methods mathematical solutions for the solution of application problems, is usually carried out through several phases [4]:

1. Analysis of the problem
2. Construction of the model
3. Model analysis
4. Numerical solution
5. Validation of the model

1.2 Combinatorial optimization

Within Operations Research, a role of fundamental importance is played by Mathematical Programming which is the discipline that has as its object the study of the problems in which you want to minimize or maximize a real function defined on $\mathbb{R}^n$ whose variables are bound to belong to a predetermined set. It is therefore a question of Optimization that are problems in which it is desired to minimize or maximize a quantity that is expressed through a function. Given f: $\mathbb{R}^n \rightarrow \mathbb{R}$ and $S \subseteq \mathbb{R}^n$ a general optimization problem is defined as:

(1)
$$\begin{Bmatrix} \min f(x) \\ x \in S \end{Bmatrix}$$

Where f is the *objective function* and S is the *feasible solution.*

A particular case of optimization problems is given by combinatorial optimization (OC), in which the admissible set is defined in terms of combinatorial structures, among which graphs certainly play an important role. The fundamental characteristic of such problems is therefore that of having discrete admissible sets. It can be said that most Combinatorial Optimization problems encountered are difficult. For this reason, the

knowledge of the most relevant OC problems, and of the related solution algorithms, is an important part of the specific skills of operational research. In particular, the various OC problems can be divided into classes according to the time required by the "fastest" algorithm for their resolution. The speed of an algorithm is defined as the number of operations necessary in the worst case in which the size of the problem itself tends to infinity, so only its order of magnitude is considered. The main categories are:

- P problems, for which an algorithm that ends in polynomial time with respect to the size of the data is known. A typical example is the sorting of n values, for which there are algorithms that check this property. The P stands for polynomial.
- NP problems, if its solution can be guessed and verified in polynomial time; the acronym NP stands for nondeterministic polynomial.
- NP-hard problems, which are such that an algorithm for solving one of these problems can be converted into an algorithm for solving any NP problem. In practice, these problems are at least as complicated as any NP problem, although they may be more so. An example of an NP-hard problem is the “K cluster with fixed cardinality problem” [5].
- Full NP problems, which are both NP and NP-hard.

Many OC problems can be formulated as Integer Linear Programming problems. However, the two classes are not completely coincident. ILP provides a powerful language to uniformly express OC problems defined on very different structures. In fact, there may be many different ILP formulations of the same OC problem. The problem analysed in the thesis is instead a *Mixed integer linear programming* (MILP) model with integer/Boolean variables. To make the model linear, the objective function must be a linear function of the variables, just as the constraints must be linear equations or inequalities involving the variables. Integer linear optimization problems are the most difficult to solve, but they are a very effective tool for modelling real problems. Note that solving a problem of this type using continuous linear programming algorithms and rounding in all possible ways to obtain integer solutions, can lead to solutions that are very far from the discrete optimum or even not to find a feasible solution.

A MILP generally consists of three basic elements:

1. Decision variables (x1, x2, ..., xn): These are the unknowns of the problem, entities such that given any (possible) solution to the problem, if we know their value, we can immediately check if the solution is possible and calculate the value of the cost function.
2. Constraints: They are mathematical relations that express the requirements of the considered problem. When a set of variables is properly defined then all requirements can be formulated naturally through equations and inequalities of the variables.
3. Objective function: The function f of the decision variables that must be maximized or minimized.

The solution of an optimization problem, which has been formulated through a mathematical programming model, consists in determining the values of the variables such as to satisfy all the constraints and to maximize or minimize the value of the objective function [6].

1.3 The solver

The codes developed in this thesis has been written in Visual Studio 2019 (IDE developed by Microsoft) and the language used is C++. The optimization solver used is IBM ILOG CPLEX 12.10, an optimizer used to formulate and solve mathematical models that provide a language that is closed to the mathematical formulation. It offers the opportunity to use a C++ library for designing different kind of math problem like those presented in this thesis. To solve MILP problems CPLEX uses the *branch & cut* algorithm which is a hybrid between the *branch & bound* method and the *cutting plane* [7][8].

1.3.1 Branch & Bound

The branch-and-bound (B&B) algorithm is a generic optimization technique that is based on an intelligent enumeration of the set X of the feasible solutions of a combinatorial optimization problem [9]. The B&B algorithm optimizes the search by exploiting a very simple principle: do not explore a region of the solution space if it can be shown that it does not contain better solutions than the known ones. B&B is a divide and conquer approach: divide X into smaller subsets and solve the problem on each subset. This is done recursively, dividing the feasible regions of the sub-problems into subsets. If this recursion were done completely, we would eventually enumerate all the possible feasible solutions of the problem. The recursive division of the set of solutions of the starting problem (set X) gives rise to a tree of feasible solutions, as shown in Figure 2.

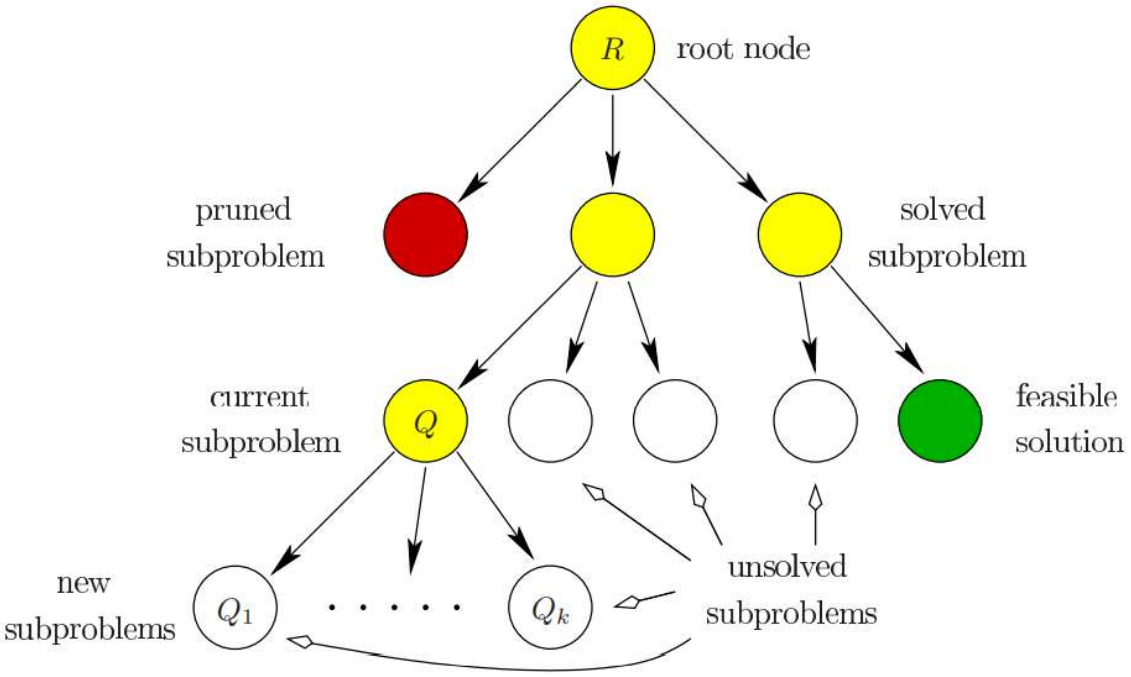

Figure 1, tree of feasible solution.

Considering a maximization problem where $c: X \rightarrow \mathbb{R}$ is a objective function to maximize, $\bar{x} \in X$ is a known feasible solution and the cost $z = c(\bar{x})$ is a lower bound of the optimal solution. The B&B algorithm consists of the following steps:

- In the bounding phase of the algorithm, the problem relaxes and solutions that belong to a superset of X are admitted such that $Y \supseteq X$. The solution of relaxedly provides an upper bound on the value of the optimal solution. If the solution of this relaxation belongs to X or has cost equal to z, the new (if feasible) or known solution is optimal.
- We identify a separation X^* of X, i.e. a finite set X^* of subsets of X, such that:

$$\bigcup_{X_i \in X^*} X_i = X$$

- Each subset X_i is called a child of X. This phase, called "branching", is justified by the fact that if X^* is a separation, then:

$$\max\{c(c) \mid x \in X\} = \max\{\max\{c(x) \mid x \in X_i\} X_i \in X^*\}$$

  X^* is often, though not necessarily, a partition of X. The children of X are added to the queue of the subproblems to be processed.

- A subproblem from the queue is selected and a relaxation of it is solved. There are four possible cases:
  1. If we find a feasible solution better than the incumbent $\bar{x}$, then we replace $\bar{x}$ with the new solution and continue.
  2. If the subproblem admits no solution, then it is discarded (pruning by infeasibility).
  3. Otherwise, compare the relaxation value with the global upper bound z. If it is greater than or equal to z, then it is still possible to discard the subproblem (pruning by optimality), as it cannot lead to a better solution than the one we already know.
  4. Finally, if it was not possible to discard the problem, it is necessary to do further branching and add the children of the subproblem to the list of subproblems to be processed. [10]
- This continues until the list of subproblems becomes empty. When this happens, $\bar{x}$ is the optimal solution.

In a B&B algorithm one of the fundamental choices is that of relaxation because it must be easy to solve and at the same time it must give upper bound values that are reasonable. Considering that the B&B algorithm is mainly used for solving MIP problems, linear relaxation is generally used by removing the constraint of being an integer.

1.3.2 The Cutting-Plane algorithm

The idea of the cutting-plane method is to solve a series of linear relaxations that gradually approximate the feasible region around the optimal solution. Considering as initial linear relaxation:

$$\max\{c^T x \mid Ax \leq b, x \geq 0\}$$

The following steps are followed:

1. Solve the current linear relaxation and let x^* be the basic optimal solution.
2. If $x^* \in X$, then x^* is an optimal solution, STOP.
3. Determine an inequality $\alpha^T x \leq \beta$ valid for X that cut x^*.
4. Add the constraint $\alpha^T x \leq \beta$ to the current linear relaxation and return to 1.

The algorithm *branch and cut* is therefore an improved version of the algorithms described in paragraphs 1.3.1 and 1.3.2 to solve MILP problems.

Chapter 2

K cluster with fixed cardinality problem

2.1 Introduction

Clustering problem is an unsupervised method used in many fields of machine learning and can be summarized as a division of several items in different groups with common features [11]. For instance, when making a product search on the web, groups of other products are proposed to the user, based on the collected data obtain by the search of other users. Relevant examples of applications of clustering method are:

- Image processing and Pattern Recognition
- Market research
- Document categorization
- Scientific data analysis
- Web search

To understand better the concept, it is possible to observe Figure 2, in which a given point $x^{(i)} \in \mathbb{R}^n$, $i = 1, .., N$ is assigned to a cluster of points based on minimizing the distance from point $x^{(i)}$ to other point of the same group [12] [13]. This visualization can be useful when a point is described by a vector of two components that can be represented in a two-dimensional space. For problem with n-dimensional vector, with n > 3, graphical representations become more complex.

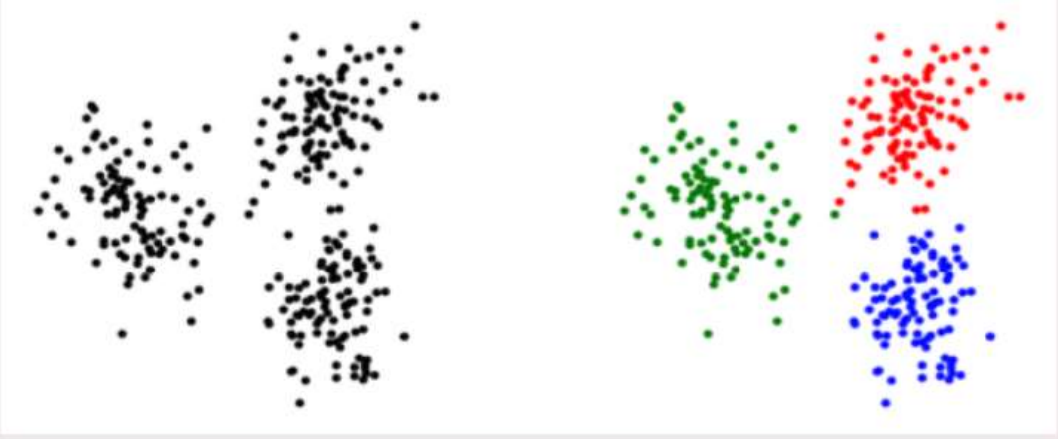

Figure 2, Input data on left (a) and solution on right (b).

This work is based on K clusters with fixed cardinality problem which is a particular problem of clustering in which K disjoint clusters are considered, each one with exactly M_k items, selected from a set of N items such that $\sum_{k=1}^{N} M_k < N$. The best solution to this problem is the one that maximize the similarity between each pair of items (i, j) in the same cluster. The similarity is represented by a similarity matrix s in which the element $s_{ij} \in [0, 1]$.

KCCP is an NP-hard combinatorial optimization problem as demonstrated by *Billionet (2005)* [14] and it is difficult obtain optimal solutions in a small amount of time.

2.2 The paper's models

The study used as a reference in this thesis is that of *Goncalves et Lourenco (2019)* [15]. It is based on the development of MILP formulations capable of solving KCCP problems. Each model has obtained different results and the two that guarantee the best performance in terms of time (F2 and F6) have been chosen for the development of metaheuristics algorithms. To do this the following notations have been considered:

- i, j → items indexes ($i, j \in \{1, \dots, N\}$),
- k → cluster index ($k \in \{1, \dots, K\}$),
- N → number of items ($N \in \mathbb{N}$),

- K → number of clusters ($K \in \mathbb{N}, K < N$),
- Mk → number of items per cluster $k(M_k \in \mathbb{N}, \sum_k M_k < N)$,
- S_{ij} → similarity between items i and j, element of a symmetric matrix with diagonal elements equal to zero ($0 \leq s_{ij} \leq 1$).

2.2.1 The models

In this paragraph the math formulations of model F2 and F6 are presented. The characteristics described by the objective function and the constraints are described in the following list:

I. The objective function that gives the total similarity, which is the sum of the similarity values between pairs of items in the same cluster.
II. It forces y_{ijk} whenever $x_{ik} = 1 = x_{jk}$.
III. It forces each item to belong to one cluster at most
IV. Cardinality constraints do not allow violation of the number of items in each cluster.
V. It defines the total number of variables y_{ijk} which are equal to 1.
VI. It defines x_{ik} as a binary variable.
VII. It forces y_{ijk} to be between 0 and 1

On the F2 model, the following mathematical functions are implemented:

(I) $$\max \sum_{k=1}^{K} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} s_{ij} y_{ijk}$$

(II) $$y_{ijk} \geq x_{ik} + x_{jk} - 1 \quad 1 \leq i < j \leq N;\ k = 1, \ldots, K$$

(III) $$\sum_{k=1}^{K} x_{ik} \leq 1 \qquad i = 1, \ldots, N$$

(IV) $$\sum_{i=1}^{N} x_{ik} = M_k \qquad i = 1,...,K$$

(V) $$\sum_{i=1}^{j-1} y_{ijk} + \sum_{i=j+1}^{N} y_{jik} = (M_k - 1)x_{jk} \quad j = 1,...,N;\ k = 1,...,K$$

(VI) $$x_{ik} \in \{0,1\} \quad i = 1,...N;\ k = 1,...,K$$

(VII) $$0 \le y_{ijk} \le 1 \quad 1 \le i < j \le N;\ k = 1,...,K$$

On the F6 model, the constrains which are describe for F2 model are also so implemented, but two more constrains must be imposed:

VIII. It results in $y_{ijk}=1$ whenever $x_{ik}=1$

IX. It results in $y_{ijk}=1$ whenever $x_{jk}=1$

(I) $$\max \sum_{k=1}^{K} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} s_{ij} y_{ijk}$$

(II) $$y_{ijk} \ge x_{ik} + x_{jk} - 1 \quad 1 \le i < j \le N;\ k = 1,...,K$$

(VIII) $$y_{ijk} \le x_{ik} \quad 1 \le i < j \le N;\ k = 1,...,K$$

(IX) $$y_{ijk} \le x_{jk} \quad 1 \le i < j \le N;\ k = 1,...,K$$

(III) $$\sum_{k=1}^{K} x_{ik} \le 1 \qquad i = 1,...,N$$

(IV) $$\sum_{i=1}^{N} x_{ik} = M_k \qquad i = 1,...,K$$

(V) $$\sum_{i=1}^{j-1} y_{ijk} + \sum_{i=j+1}^{N} y_{jik} = (M_k - 1)x_{jk} \qquad j = 1,...,N;\ k = 1,...,K$$

(VI) $$x_{ik} \in \{0,1\} \qquad i = 1,...N;\ k = 1,...,K$$

(VII) $$0 \le y_{ijk} \le 1 \qquad 1 \le i < j \le N;\ k = 1,...,K$$

2.2.2 The results

The two MILP models and their results are presented in the tables below (Table 1, Table 2 and Table 3 for model F2 and Tables 4, 5, 6 for model F6). In the first column there is the density value, in the second column the total sum of the number of items per cluster, in the third one the time taken by the CPU to resolve an instance and in the fourth the number of instances that have not been resolved within the time limit of 7200 seconds.

The results obtained by the two models are similar, the F2 model (Tables 1, 2, 3) solves a greater number of instances than F6 (Tables 4, 5, 6). The difference between the two models is given by the presence of two additional constraints in the F6 model (7 and 8). It can be noted that the instances that take longer are those with $\sum_k M_k = 30$ and higher density values, in particular in the cases of K = 3, the $\sum_k M_k = 30$ and the $d \geq 50$ no instance has been solved for both models.

Table 1, F2 model for K=1.

d	$\sum_k M_k$	Average CPU time (s)
25	**10**	7,7
25	**20**	16,4
25	**30**	2,7
50	**10**	10,7
50	**20**	95
50	**30**	2,4
75	**10**	10,8
75	**20**	200,4
75	**30**	2,6

Table 2, F2 model for K=2.

d	$\sum_k M_k$	Average CPU time (s)	Unsolved instances
25	**10**	5,3	
25	**20**	208,9	
25	**30**	632,5	
50	**10**	7,1	
50	**20**	1168,4	
50	**30**	3707,7	5
75	**10**	6,3	
75	**20**	1361,6	
75	**30**	3843,2	5

Table 3, F2 model for K=3.

d	$\sum_k M_k$	Average CPU time (s)	Unsolved instances
25	**10**	4	
25	**20**	761,1	
25	**30**	3998,4	3
50	**10**	2,6	
50	**20**	3372,1	1
50	**30**	7202,4	10
75	**10**	2,5	
75	**20**	2411,1	
75	**30**	7202	10

Table 4, F6 model for K=1.

d	$\sum_k M_k$	Average CPU time (s)
25	**10**	6,3
25	**20**	10,5
25	**30**	0,7
50	**10**	9,9
50	**20**	67,9
50	**30**	0,9
75	**10**	11
75	**20**	201,8
75	**30**	1,5

Table 5, F6 model for K=2.

d	$\sum_k M_k$	Average CPU time (s)	Unsolved instances
25	**10**	7	
25	**20**	217,3	
25	**30**	293,9	
50	**10**	10,9	
50	**20**	1291,8	
50	**30**	3715,7	5
75	**10**	7,8	
75	**20**	1397	
75	**30**	4026,6	5

Table 6, F6 model for K=3.

d	$\sum_k M_k$	Average CPU time (s)	Unsolved instances
25	**10**	5,6	
25	**20**	1608,4	
25	**30**	4686,3	3
50	**10**	3,6	
50	**20**	4862,2	5
50	**30**	7202,1	10
75	**10**	3,9	
75	**20**	3347,9	2
75	**30**	7201,7	10

2.2.3 The Instances

Once the results obtained in the paper have been analysed, the first step is to generate the instances to implement the models present in the study. To do this, the edge graphs present in the CEDRIC's library [16] were used (the same ones used in the publication). They are defined by:

- density: 0.25, 0.50 and 0.75.
- number of nodes: N = 40.
- total number of items per cluster $\sum_{k=1}^{K} M_k = 10, 20 \; and \; 30$.

Five different graphs have been associated with each density value, to have 15 different instances. Each graph is composed of a pair of numbers that represent the connections between the items (see an example in appendix A.1.). Through a MATLAB program (see appendix A.2.), each pair has been associated with a positive weight s_{ij}, such that $0 < s_{ij} < 1$. In this way 15 different similarity matrices were obtained. Then the similarity matrixes were saved to a ".txt" file and named in such a way that they could be easily loaded from the code (see an example in appendix A.3.). In fact, they contain the density value, the $\sum_k M_k$ value and a progressive number from 1 to 5 to differentiate the matrices with the same properties but with different s_{ij}. As an example, the name of a similarity matrix with d = 50, $\sum_k M_k = 30$ and 3 as a progressive number is shown: *similaritymatrix_050_30_3.*

The number of total instances is affected by the different values of K:

- If K = 1 the 15 instances have been used for the three different values of M_1 (10, 20, 30), for a total of 45 instances.
- If K = 2 the 15 instances have been used for the two pairs of values M_1 and M_2, for a total of 90 instances.
- If K = 3 the 15 instances have been used for the two triples of values M_1, M_2 and M_3, for a total of 90 instances.

The scheme of the instances is summarized in Table 7.

Table 7, scheme of the instances.

K	$\sum_{k=1}^{K} M_k$	M_k			N. Inst
1	10	$M_1 = 10$			15
	20	$M_1 = 20$			15
	30	$M_1 = 30$			15
2	10	$M_1 = 5\left(\frac{1}{2}\right)$	$M_2 = 5\left(\frac{1}{2}\right)$		15
		$M_1 = 2\left(\frac{1}{5}\right)$	$M_2 = 8\left(\frac{4}{5}\right)$		15
	20	$M_1 = 10\left(\frac{1}{2}\right)$	$M_2 = 10\left(\frac{1}{2}\right)$		15
		$M_1 = 4\left(\frac{1}{5}\right)$	$M_2 = 16\left(\frac{4}{5}\right)$		15
	30	$M_1 = 15\left(\frac{1}{2}\right)$	$M_2 = 15\left(\frac{1}{2}\right)$		15
		$M_1 = 24\left(\frac{4}{5}\right)$	$M_2 = 6\left(\frac{1}{5}\right)$		15
3	10	$M_1 = 3\left(\frac{3}{10}\right)$	$M_2 = 3\left(\frac{3}{10}\right)$	$M_3 = 4\left(\frac{4}{10}\right)$	15
		$M_1 = 2\left(\frac{1}{5}\right)$	$M_2 = 3\left(\frac{3}{10}\right)$	$M_3 = 5\left(\frac{1}{2}\right)$	15
	20	$M_1 = 7\left(\frac{7}{20}\right)$	$M_2 = 7\left(\frac{7}{20}\right)$	$M_3 = 6\left(\frac{3}{10}\right)$	15
		$M_1 = 3\left(\frac{3}{20}\right)$	$M_2 = 7\left(\frac{7}{20}\right)$	$M_3 = 10\left(\frac{1}{2}\right)$	15
	30	$M_1 = 10\left(\frac{1}{3}\right)$	$M_2 = 10\left(\frac{1}{3}\right)$	$M_3 = 10\left(\frac{1}{3}\right)$	15
		$M_1 = 5\left(\frac{1}{6}\right)$	$M_2 = 10\left(\frac{1}{3}\right)$	$M_3 = 15\left(\frac{1}{2}\right)$	15

2.3 Code implementation

After having generated the instances and the related similarity matrices, the code was written to compare the results with the F2 and F6 models of the paper. The first step

was to integrate CPLEX 1210 with Visual Studio 2019 by following the steps described below:

1. Go to Project → Properties → C / C ++ → General → Additional Include Directory and copy the links below:
    - *C:* \ *Program Files* \ *IBM* \ *ILOG* \ *CPLEX_Studio1210* \ *concert* \ *include*
    - *C:* \ *Program Files* \ *IBM* \ *ILOG* \ *CPLEX_Studio1210* \ *cplex* \ *include*
    - *C:* \ *Program Files* \ *IBM* \ *ILOG* \ *CPLEX_Studio1210* \ *cpoptimizer* \ *include*

2. Go to Project → Properties → Linker General → Additional Libraries Directory and copy the links below:
    - *C:* \ *Program Files* \ *IBM* \ *ILOG* \ *CPLEX_Studio1210* \ *cplex* \ *lib* \ *x64_windows_msvc14* \ *stat_mda* \
    - *C:* \ *Program Files* \ *IBM* \ *ILOG* \ *CPLEX_Studio1210* \ *concert* \ *lib* \ *x64_windows_msvc14* \ *stat_mda* \
    - *C:* \ *Program Files* \ *IBM* \ *ILOG* \ *CPLEX_Studio1210* \ *cpoptimizer* \ *lib* \ *x64_windows_msvc14* \ *stat_mda* \

3. Go to Project → Properties → Linker Input → Additional Dependencies and put the links below:

- cplex12100.lib
- ilocplex.lib
- concert.lib
- cp.lib

4. Go to Project → Properties → C / C ++ → Preprocessor → Preprocessor Definitions and put the links below:

- *WIN32*
- *_CONSOLE*

- *IL_STD*
- *_CRT_SECURE_NO_WARNINGS*

After that the code has been developed (see appendix B.1.).

2.4 The results

The first model implemented was F2 and the results from the average of 5 instances (if K=1) and 10 instances (if K=2 or K=3) are shown below (Tables 8, 9 and 10). While the F6 model was subsequently implemented.

For the instances that have not been resolved by the solver, the average gap has been calculated as $\frac{v(\overline{P})-v(P)}{v(P)} * 100\%$ where $v(P)$ is the optimum value of the problem and $v(\overline{P})$ is the linear relaxation of the same problem.

Table 8, model F2 for K=1.

d	$\sum_k M_k$	Average CPU time (s)
25	**10**	3,3654
25	**20**	14,0054
25	**30**	1,367
50	**10**	4,975
50	**20**	17,4622
50	**30**	2,5262
75	**10**	3,6334
75	**20**	16,3456
75	**30**	2,106

Table 9, model F2 for K=2.

d	$\sum_k M_k$	Average CPU time (s)
25	**10**	3,2402
25	**20**	69,7015
25	**30**	174,1914
50	**10**	3,2765
50	**20**	146,0524
50	**30**	1086,0858
75	**10**	2,3186
75	**20**	308,5057
75	**30**	1497,317

Table 10, model F2 for K=3.

d	$\sum_k M_k$	Average CPU time (s)	Unsolved instances	Average gap (%)
25	**10**	1,0118	0	0
25	**20**	294,9746	0	0
25	**30**	1723,9246	0	0
50	**10**	0,8682	0	0
50	**20**	923,2222	0	0
50	**30**	6889,748	9	7,17
75	**10**	0,4477	0	0
75	**20**	490,9513	0	0
75	**30**	6685,546	9	6,61

The tests carried out led to two relevant results:

- CPU times have been significantly reduced except for the only case in which d = 50 and $\sum_k M_k = 30$, these differences are probably caused by the performance of the PC on which the code was run and by a more updated version of CPLEX than the one presented in the paper.
- the instances that have not been solved are those with K = 3, d≥50 and $\sum_k M_k = 30$. In total there are 18 instead of the 34 unresolved totals in the paper.

Table 11, model F6 for K=1.

d	∑Mk	Average CPU time (s)
25	**10**	1,535
25	**20**	3,5134
25	**30**	0,649
50	**10**	3,7572
50	**20**	14,2712
50	**30**	1,42
75	**10**	3,641
75	**20**	19,362
75	**30**	1,267

Table 12,model F6 for K=2.

d	∑Mk	Average CPU time (s)
25	**10**	2,4301
25	**20**	28,3076
25	**30**	58,0065
50	**10**	3,6086
50	**20**	84,9515
50	**30**	214,8924
75	**10**	2,7915
75	**20**	150,0684
75	**30**	515,1628

Table 13, model F6 for K=3.

d	∑Mk	Average CPU time (s)	Unsolved instances	Average gap (%)
25	**10**	1,4041	0	0
25	**20**	121,634	0	0
25	**30**	414,2288	0	0
50	**10**	1,2108	0	0
50	**20**	309,4868889	0	0
50	**30**	2034,947111	0	0
75	**10**	0,9516	0	0
75	**20**	202,0212	0	0
75	**30**	2765,9879	1	0,38

To better visualize the results obtained, the average times used by the CPU to resolve the instances have been graphically represented (Figure 3, 4 and 5).

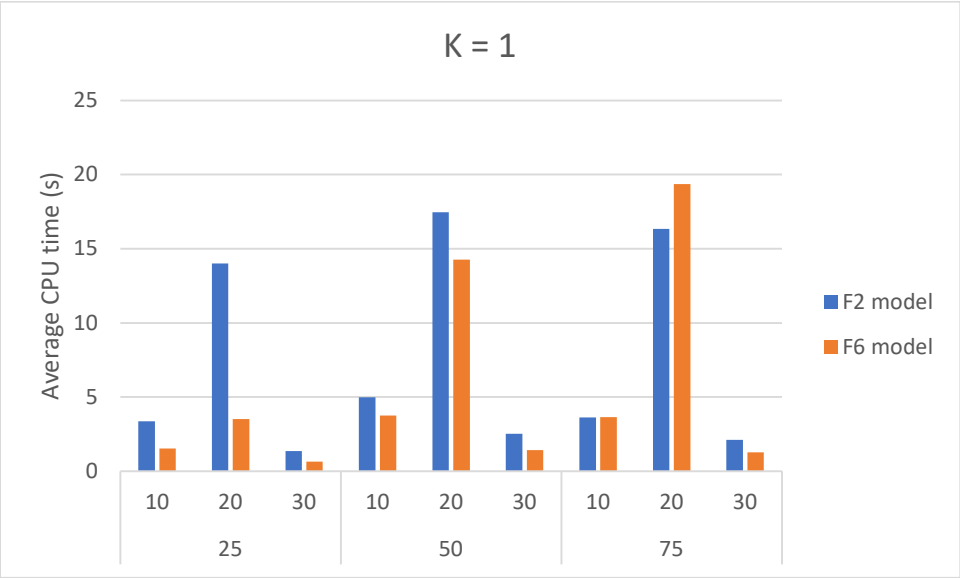

Figure 3, graphic comparison of the F2 and F6 models for K=1.

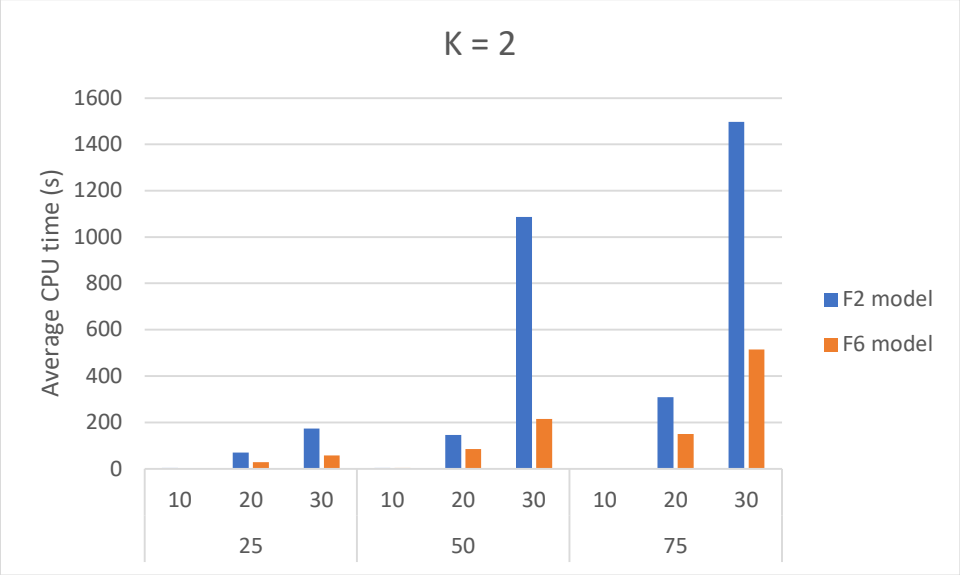

Figure 4, graphic comparison of the F2 and F6 models for K=2.

.

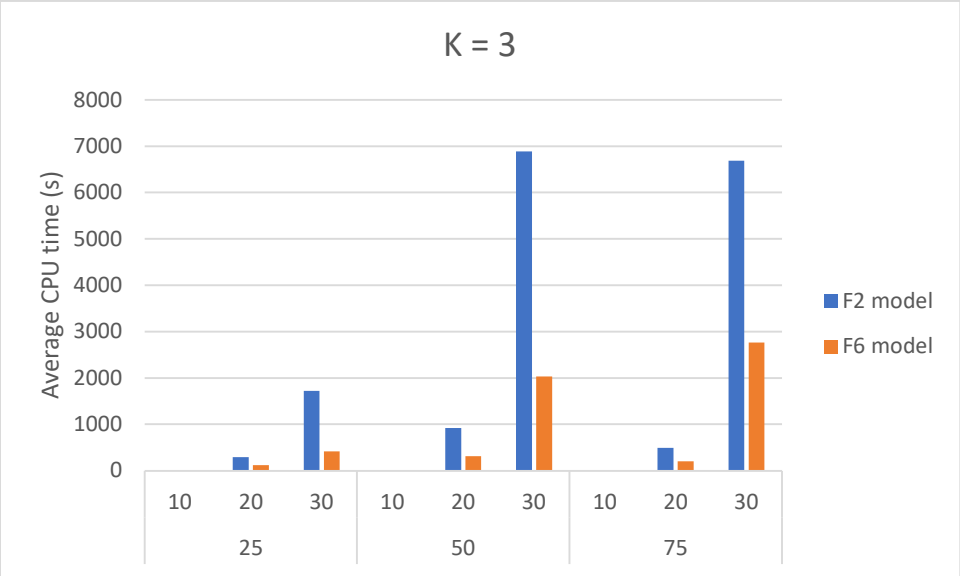

Figure 5, graphic comparison of the F2 and F6 models.

From the results obtained it is possible to observe that in the case of K = 1 the models do not show great differences between them and the average of all the instances never exceeds 20 seconds (Figure 3).

In the case of K = 2 it can be seen how the F6 model has much better performances (Figure 4) and both are able to solve all the models within the time limit (Tables 9 and 12). In cases where $\sum_k M_k = 30$ the results obtained by F6 are 5 times lower than F2 for d = 50 and approximately 3 times lower for d = 75.

If K = 3 the argument is like what was done for K = 2: the F6 model has better performance by guaranteeing the resolution of all instances except for the one with $\sum_k M_k = 30$ and d = 75 having a gap of 3.81 %.

After this analysis it is possible to make the following considerations:

- the instances that take longer are related to a value of K = 3, both in the paper and in the tests carried out.
- the tests carried out are generally better than those present in the paper. This is probably caused by the PC, the CPLEX version used and the CPLEX settings that have been set.

- F6 model provides better results in terms of CPU than the F2 model.

Chapter 3

Metaheuristic algorithm

3.1 Introduction to metaheuristics

The purpose of an optimization problem is closely linked to the definition of a computation procedure, that is an algorithm, capable of solving your instances efficiently. The reasoning behind solving a problem through the application of an algorithm is to obtain the procedure output an optimal solution for a given input instance. For solving polynomial problems, the algorithm that determines an optimal solution is defined exact algorithm. The search for the optimal solution through exact methods explores the entire set of solutions through implicit enumeration techniques such as dynamic programming, branch-and-cut and branch-and-bound. In dealing with NP-arduous problems, on the other hand, the application of an algorithm exact could be computationally intractable, for this reason there is a need to resort to algorithms that return an approximate solution. The algorithms that work in this sense are so-called meta-heuristics such as tabu search and local search [17]. These algorithms can be applied in their general form to a wide range of problems; however, it is often advisable to adapt them to characteristics of the problem to obtain a more efficient algorithm. For this purpose, new forms of hybrid algorithms have been developed in how much they combine exact and meta-heuristic methods and take the name of heuristic metaheuristics. Generally, these criteria differ from each other because they adapt to the intrinsic characteristics of the problem under consideration, but they possess then a fixed base structure.

Metaheuristics are optimization algorithms that exploit the combination of mathematical programming and heuristic methods for solving of combinatorial optimization problems. The use of these methods has allowed to improve the resolution

of various problems of both theoretical and real-world interest. Generally, the "basic" part of the algorithm is solved from a MILP linear programming solver via a heuristic algorithm. The MILP solver is supported by a metaheuristic algorithm whose purpose is to solve a subset of the general problem. Consequently, being the subset a portion of the whole problem, we will have one faster resolution.

The contributions made by research in this field can be classified under two aspects: how metaheuristic methods can improve exact methods and how exact methods can improve metaheuristic algorithms. The second case is the one most studied since potential benefits compared in the state of the art, they are more evident. In this area, they can be identified two directions in which research is moving: the first consists in use of mathematical programming techniques as components to be included in metaheuristic methods, the second in creating metaheuristic algorithms derived from the logic of internal functioning of programming methods mathematics.

3.2 LP based local search

3.2.1 Local Search

The local search technique is a widely used metaheuristics in the field of operational research. The algorithm is based on the search for a local optimum starting from an initial solution S_0 [18]. At each step the solution is iterated to verify the solutions of a neighborhood that can obtain a better objective function than the previous one. This method allows to improve or leave the starting solution unchanged even if it does not guarantee the achievement of the optimum. In fact, it is typical that the algorithm encounters local optimum but not global.

3.2.2 The algorithm: LP BASED LOCAL SEARCH 2-DIVISION

The metaheuristic used in this work is part of the LP based local search, which is part of a subgroup of the Local searches described in paragraph 3.2.1. It consists in the following steps:

1. Select a subset of variables.
2. Fix the variables selected and free the other ones.
3. Run the solver.

The new solution is at least the same or better than the previous.

The KCCP problem provides as a solution a matrix x which has the number of rows equal to the value assigned to K and the number of columns equal to the number of items N (in this case 40). Each line is composed of a string of 1 and 0. The solver gives as solution x [k][i] = 1 if the i-th item belongs to cluster k.

Considering a generic starting solution, the algorithm divides at each iteration all the row vectors x [k] [] into two equal parts (20 elements for each division) and blocks all the 1s present in the first half of the vector leaving all the other variables free then it blocks all the 1s present in the second half of the vector leaving all the other variables free. In this way the solver will have fewer 1's to choose, and the CPU will take less time.

Let's take the *similaritymatrix_075_30_4* instance as an example, with K = 3, M1 = 10, M2 = 10 and M3 = 10. This instance, in model F6, provides the solution taking 3801 seconds and the objective function is 80,617. The goal is to try to reduce the time used by the solver with the passages below. For simplicity only the vector x[k][] with k = 1 is reported but for each step the algorithm also works on the values of k = 2, 3.

1. The first step is to generate an initial solution that respects the constraints of the F6 model. In this specific example it was generated by running the solver for a time limit of 20 seconds (Table 14).

Table 14, inital solution by running the solver for 20 seconds.

i-th	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0/1	0	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	1	0	0

i-th	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
0/1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1

The objective function value that was obtained after this step is 67.7975 with a consequent relative error of 15.90% with respect to the optimal solution.

2. In the second step, starting from the initial solution, the algorithm sets all the variables x [k] [i] = 1 with i ≤20 (highlighted in green in the Table 15) leaving all the others free.

Table 15, second step of the algorithm.

i-th	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0/1	0	1	1	0	0	0	1	0	0	1	1	1	0	1	0	0	0	1	0	0

i-th	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
0/1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1

3. In the third step, starting from the previous solution (point 2), the algorithm sets all the variables x [k] [i] = 1 with i >20 (highlighted in green in the Table 16) leaving all the others free.

Table 16, third step of the algorithm.

i-th	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0/1	0	1	0	1	0	0	1	0	0	1	1	0	0	0	0	0	0	1	0	0

i-th	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
0/1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1

After these steps an OF of 80.5602 was obtained, very close to the optimal one with a relative deviation of 0.07%. The real advantage, however, is visible in the time taken by the CPU to obtain this result: it takes only 21,284 s.

3.2.3 The starting solution

In the example analyzed in the previous paragraph, the starting solution was obtained by running the solver for 20 seconds. However, in the analysis of the results, different initial solution times were tested: 3s, 5s, 10s and 20s were chosen as starting execution times. The goal is to start from a solution that is good enough but that, at the same time, does not excessively affect the time taken by the algorithm. To do this, all instances for k> = 2 have been tested for the different time limit values and the results are reported in the tables below (Tables 17 and 18). In the study of the algorithm the cases of k = 1 have not been analyzed because the results present in tables 8 and 11 guarantee the optimum in good execution times.

Table 17, analisys of the starting solution for K=2.

NAME	M1	M2	OF F6	OF 3 s	OF 5 s	OF 10 s	OF 20 s
similaritymatrix_025_10_1	5	5	11,4966	11,3021	11,4966	11,4966	11,4966
similaritymatrix_025_10_1	2	8	13,9991	13,9991	13,9991	13,9991	13,9991
similaritymatrix_025_10_2	5	5	12,0583	12,0576	12,0583	12,0583	12,0583
similaritymatrix_025_10_2	2	8	13,4004	13,4004	13,4004	13,4004	13,4004
similaritymatrix_025_10_3	5	5	11,9049	11,9049	11,9049	11,9049	11,9049
similaritymatrix_025_10_3	2	8	12,6588	12,6588	12,6588	12,6588	12,6588
similaritymatrix_025_10_4	5	5	12,1112	12,1112	12,1112	12,1112	12,1112
similaritymatrix_025_10_4	2	8	13,653	13,653	13,653	13,653	13,653
similaritymatrix_025_10_5	5	5	11,5467	11,5467	11,5467	11,5467	11,5467
similaritymatrix_025_10_5	2	8	13,2329	13,2329	13,2329	13,2329	13,2329
similaritymatrix_050_10_1	5	5	15,228	15,228	15,228	15,228	15,228
similaritymatrix_050_10_1	2	8	16,3261	16,3261	16,3261	16,3261	16,3261
similaritymatrix_050_10_2	5	5	14,1292	14,082	14,082	14,1292	14,1292
similaritymatrix_050_10_2	2	8	16,7009	16,5117	16,7009	16,7009	16,7009
similaritymatrix_050_10_3	5	5	14,9242	14,6181	14,9242	14,9242	14,9242
similaritymatrix_050_10_3	2	8	17,4663	17,1367	17,4663	17,4663	17,4663

similaritymatrix_050_10_4	5	5	14,4746	14,461	14,4746	14,4746	14,4746
similaritymatrix_050_10_4	2	8	17,4068	17,1917	17,4068	17,4068	17,4068
similaritymatrix_050_10_5	5	5	15,0162	14,9146	15,0162	15,0162	15,0162
similaritymatrix_050_10_5	2	8	17,648	17,648	17,648	17,648	17,648
similaritymatrix_075_10_1	5	5	16,3946	16,3946	16,3946	16,3946	16,3946
similaritymatrix_075_10_1	2	8	20,5697	20,5697	20,5697	20,5697	20,5697
similaritymatrix_075_10_2	5	5	16,2103	16,2103	16,2103	16,2103	16,2103
similaritymatrix_075_10_2	2	8	20,1843	20,1843	20,1843	20,1843	20,1843
similaritymatrix_075_10_3	5	5	16,5491	16,5491	16,5491	16,5491	16,5491
similaritymatrix_075_10_3	2	8	21,914	21,914	21,914	21,914	21,914
similaritymatrix_075_10_4	5	5	16,2832	16,2832	16,2832	16,2832	16,2832
similaritymatrix_075_10_4	2	8	19,9435	19,7932	19,9435	19,9435	19,9435
similaritymatrix_075_10_5	5	5	16,5168	16,5168	16,5168	16,5168	16,5168
similaritymatrix_075_10_5	2	8	20,4485	20,4485	20,4485	20,4485	20,4485
similaritymatrix_025_20_1	10	10	30,3149	26,1712	29,3693	29,3693	29,3693
similaritymatrix_025_20_1	4	16	34,5235	34,4661	34,5235	34,5235	34,5235
similaritymatrix_025_20_2	10	10	32,5831	26,4975	29,7237	29,7237	30,7034
similaritymatrix_025_20_2	4	16	37,9468	33,6142	35,3435	37,9468	37,9468
similaritymatrix_025_20_3	10	10	30,4026	24,9637	26,9538	27,666	30,4026
similaritymatrix_025_20_3	4	16	33,9979	23,9272	32,6317	32,6317	32,8456
similaritymatrix_025_20_4	10	10	30,0218	21,6533	25,4121	28,4167	30,0218
similaritymatrix_025_20_4	4	16	33,6102	22,9393	32,6231	33,6102	33,6102
similaritymatrix_025_20_5	10	10	31,0816	12,6541	26,5453	28,104	31,0816
similaritymatrix_025_20_5	4	16	32,939	13,5581	32,0416	32,939	32,939
similaritymatrix_050_20_1	10	10	46,3435	39,2056	40,786	40,786	43,1889
similaritymatrix_050_20_1	4	16	54,5386	28,6728	53,0482	53,0633	54,0429
similaritymatrix_050_20_2	10	10	48,284	36,3434	42,0535	42,0535	48,284
similaritymatrix_050_20_2	4	16	56,839	40,0073	49,4231	52,2576	56,1295
similaritymatrix_050_20_3	10	10	45,0089	30,2366	42,6208	43,8014	43,9827

similaritymatrix_050_20_3	4	16	54,9534	39,9501	54,16	54,3528	54,6277
similaritymatrix_050_20_4	10	10	43,8548	25,6368	38,9538	38,9538	41,7443
similaritymatrix_050_20_4	4	16	52,5016	45,7184	49,1844	49,1844	51,8445
similaritymatrix_050_20_5	10	10	46,2017	33,154	41,9852	43,0378	45,2717
similaritymatrix_050_20_5	4	16	54,864	48,1948	50,6522	52,433	53,2427
similaritymatrix_075_20_1	10	10	54,4506	37,5038	50,5339	50,5339	52,7475
similaritymatrix_075_20_1	4	16	66,1831	60,63	60,63	61,3877	64,9863
similaritymatrix_075_20_2	10	10	57,8937	46,2159	55,1917	55,1917	55,6069
similaritymatrix_075_20_2	4	16	72,0678	60,6557	68,4048	69,0576	70,6035
similaritymatrix_075_20_3	10	10	57,8223	46,3715	53,8988	53,8988	55,8428
similaritymatrix_075_20_3	4	16	71,1108	51,1978	66,8104	66,8104	71,0732
similaritymatrix_075_20_4	10	10	56,9956	42,7787	50,0169	50,0169	55,2861
similaritymatrix_075_20_4	4	16	71,1807	59,6011	67,996	67,996	71,1807
similaritymatrix_075_20_5	10	10	54,3156	39,625	50,9177	50,9177	52,2318
similaritymatrix_075_20_5	4	16	67,4808	62,222	62,9412	62,9412	65,6206
similaritymatrix_025_30_1	15	15	48,3419	25,3199	46,5825	46,5825	46,7854
similaritymatrix_025_30_1	24	6	56,572	41,9344	56,1846	56,572	56,572
similaritymatrix_025_30_2	15	15	55,9228	32,039	49,1649	50,8098	50,8098
similaritymatrix_025_30_2	24	6	65,4896	46,953	61,3216	63,3461	65,4896
similaritymatrix_025_30_3	15	15	52,7701	30,983	42,8059	42,8059	47,6652
similaritymatrix_025_30_3	24	6	58,7257	41,51	57,4703	57,7441	58,7037
similaritymatrix_025_30_4	15	15	50,5031	32,6199	46,1015	46,4493	47,0349
similaritymatrix_025_30_4	24	6	57,5037	38,3989	56,6294	56,6505	56,9721
similaritymatrix_025_30_5	15	15	46,8051	24,8572	46,4539	46,4539	46,4539
similaritymatrix_025_30_5	24	6	54,3592	45,9317	52,3791	54,199	54,3592
similaritymatrix_050_30_1	15	15	85,0068	66,7404	75,0519	75,0519	75,0519
similaritymatrix_050_30_1	24	6	104,795	91,9379	99,8233	102,12	103,352
similaritymatrix_050_30_2	15	15	86,0076	71,9645	72,8638	74,1295	75,8131
similaritymatrix_050_30_2	24	6	101,291	94,6532	97,1498	100,699	100,699

similaritymatrix_050_30_3	15	15	87,6286	68,5955	69,5229	76,5734	76,5734
similaritymatrix_050_30_3	24	6	106,038	93,419	104,855	104,855	104,855
similaritymatrix_050_30_4	15	15	86,817	67,3357	72,1235	72,609	74,1294
similaritymatrix_050_30_4	24	6	105,351	99,7497	100,319	104,054	104,054
similaritymatrix_050_30_5	15	15	80,8599	61,4516	66,6429	66,6429	67,3854
similaritymatrix_050_30_5	24	6	98,6817	88,1356	91,3437	96,2332	96,5265
similaritymatrix_075_30_1	15	15	109,687	92,9409	93,4282	97,2334	97,2334
similaritymatrix_075_30_1	24	6	135,574	124,618	128,615	130,739	134,722
similaritymatrix_075_30_2	15	15	114,023	92,545	102,58	102,58	102,58
similaritymatrix_075_30_2	24	6	142,987	138,815	139,274	142,255	142,255
similaritymatrix_075_30_3	15	15	111,006	93,1714	95,8913	99,1773	100,226
similaritymatrix_075_30_3	24	6	138,125	125,502	131,37	136,785	138,088
similaritymatrix_075_30_4	15	15	107,143	91,6662	91,6662	91,6662	97,0093
similaritymatrix_075_30_4	24	6	135,314	130,162	130,928	132,087	132,618
similaritymatrix_075_30_5	15	15	108,321	90,6047	90,6047	90,6047	96,3189
similaritymatrix_075_30_5	24	6	139,998	132,466	137,137	137,973	139,561

Observing table 17 it is possible to state that the starting objective function value varies differently according to the time granted to the solver. For instances with $\sum_k M_k = 30$ it is possible to obtain the optimum with very low times, sometimes even less than 3 seconds. As the number of items per cluster increases, it is more evident to give the solver more time to search for the initial solution.

Table 18, analysis of the starting solution for K=3.

NAME	M1	M2	M3	OF F6	OF 3 s	OF 5 s	OF 10 s	OF 20 s
similaritymatrix_025_10_1	3	3	4	8,8993	3,2794	8,8993	8,89929	8,89929
similaritymatrix_025_10_1	2	3	5	9,4637	9,4637	9,4637	9,46372	9,46372
similaritymatrix_025_10_2	3	3	4	9,3906	9,3906	9,3906	9,39057	9,39057

similaritymatrix_025_10_2	2	3	5	10,089	8,8995	10,089	10,0889	10,0889
similaritymatrix_025_10_3	3	3	4	9,7852	9,7852	9,7852	9,78516	9,78516
similaritymatrix_025_10_3	2	3	5	9,9048	9,9048	9,9048	9,90476	9,90476
similaritymatrix_025_10_4	3	3	4	9,8452	9,8452	9,8452	9,84519	9,84519
similaritymatrix_025_10_4	2	3	5	10,099	10,099	10,099	10,0993	10,0993
similaritymatrix_025_10_5	3	3	4	8,9353	8,9353	8,9353	8,9353	8,9353
similaritymatrix_025_10_5	2	3	5	9,5134	9,5134	9,5134	9,51342	9,51342
similaritymatrix_050_10_1	3	3	4	10,641	10,641	10,641	10,6405	10,6405
similaritymatrix_050_10_1	2	3	5	11,734	11,734	11,734	11,7341	11,7341
similaritymatrix_050_10_2	3	3	4	10,608	10,608	10,608	10,6084	10,6084
similaritymatrix_050_10_2	2	3	5	11,191	11,191	11,191	11,1914	11,1914
similaritymatrix_050_10_3	3	3	4	10,527	10,527	10,527	10,5272	10,5272
similaritymatrix_050_10_3	2	3	5	11,381	11,381	11,381	11,3806	11,3806
similaritymatrix_050_10_4	3	3	4	10,486	10,486	10,486	10,4859	10,4859
similaritymatrix_050_10_4	2	3	5	11,184	11,184	11,184	11,184	11,184
similaritymatrix_050_10_5	3	3	4	11,099	11,099	11,099	11,0992	11,0992
similaritymatrix_050_10_5	2	3	5	11,522	11,522	11,522	11,5222	11,5222
similaritymatrix_075_10_1	3	3	4	11,118	11,118	11,118	11,1182	11,1182
similaritymatrix_075_10_1	2	3	5	12,125	12,125	12,125	12,1253	12,1253
similaritymatrix_075_10_2	3	3	4	11,217	11,217	11,217	11,2169	11,2169
similaritymatrix_075_10_2	2	3	5	12,212	12,212	12,212	12,2122	12,2122
similaritymatrix_075_10_3	3	3	4	11,069	11,069	11,069	11,0686	11,0686
similaritymatrix_075_10_3	2	3	5	12,479	12,479	12,479	12,4794	12,4794
similaritymatrix_075_10_4	3	3	4	11,17	11,17	11,17	11,1699	11,1699
similaritymatrix_075_10_4	2	3	5	12,313	12,313	12,313	12,3132	12,3132
similaritymatrix_075_10_5	3	3	4	11,15	11,15	11,15	11,1499	11,1499
similaritymatrix_075_10_5	2	3	5	12,175	12,175	12,175	12,1749	12,1749
similaritymatrix_025_20_1	7	7	6	24,165	18,705	22,28	22,2801	24,0274
similaritymatrix_025_20_1	3	7	10	25,84	22,993	25,219	25,2192	25,2638

similaritymatrix_025_20_2	7	7	6	26,247	23,617	23,617	23,6173	24,7701
similaritymatrix_025_20_2	3	7	10	27,03	22,04	23,87	23,8701	27,7568
similaritymatrix_025_20_3	7	7	6	25,603	20,16	23,892	23,8916	25,7586
similaritymatrix_025_20_3	3	7	10	26,977	22,463	24,308	24,3075	26,4482
similaritymatrix_025_20_4	7	7	6	24,054	22,53	22,53	22,8166	23,8229
similaritymatrix_025_20_4	3	7	10	24,686	22,638	22,638	23,7326	26,0323
similaritymatrix_025_20_5	7	7	6	25,607	23,188	23,188	24,1037	26,2936
similaritymatrix_025_20_5	3	7	10	27,725	26,557	26,557	26,5574	27,9312
similaritymatrix_050_20_1	7	7	6	34,32	28,854	33,525	33,5254	34,3126
similaritymatrix_050_20_1	3	7	10	38,507	36,341	37,532	37,532	40,0155
similaritymatrix_050_20_2	7	7	6	33,965	26,927	29,287	31,3991	33,0592
similaritymatrix_050_20_2	3	7	10	38,58	32,002	35,022	37,0844	39,887
similaritymatrix_050_20_3	7	7	6	34,347	29,238	32,615	32,6149	32,755
similaritymatrix_050_20_3	3	7	10	36,993	35,081	35,081	35,4151	37,2811
similaritymatrix_050_20_4	7	7	6	32,907	30,294	31,137	31,4962	31,6297
similaritymatrix_050_20_4	3	7	10	35,501	34,26	34,26	34,444	35,7499
similaritymatrix_050_20_5	7	7	6	35,255	32,374	33,476	33,4758	35,1948
similaritymatrix_050_20_5	3	7	10	37,347	35,057	35,635	35,6352	35,6352
similaritymatrix_075_20_1	7	7	6	39,4	35,089	39,681	39,6813	39,6813
similaritymatrix_075_20_1	3	7	10	43,275	40,145	40,772	43,441	44,6109
similaritymatrix_075_20_2	7	7	6	42,385	38,937	39,842	42,3756	43,355
similaritymatrix_075_20_2	3	7	10	47,312	44,841	46,509	46,5091	46,7871
similaritymatrix_075_20_3	7	7	6	42,246	33,527	39,162	40,1582	41,1023
similaritymatrix_075_20_3	3	7	10	47,86	42,587	46,354	46,3543	46,7506
similaritymatrix_075_20_4	7	7	6	41,422	33,625	40,962	40,9623	40,9623
similaritymatrix_075_20_4	3	7	10	46,673	39,965	40,365	41,8898	47,0021
similaritymatrix_075_20_5	7	7	6	38,022	33,43	37,717	37,7514	39,4626
similaritymatrix_075_20_5	3	7	10	44,126	42,63	42,63	42,6296	43,9457
similaritymatrix_025_30_1	10	10	10	38,31	25,379	30,828	32,7814	36,2109

similaritymatrix_025_30_1	5	10	15	39,581	30,263	33,185	39,0736	39,0736
similaritymatrix_025_30_2	10	10	10	43,614	32,182	37,212	37,5875	37,8982
similaritymatrix_025_30_2	5	10	15	48,756	27,59	45,28	45,2802	45,2802
similaritymatrix_025_30_3	10	10	10	42,228	27,494	29,679	35,0452	35,0452
similaritymatrix_025_30_3	5	10	15	42,951	24,495	39,679	39,6788	39,6788
similaritymatrix_025_30_4	10	10	10	40,294	20,753	33,698	36,9388	37,3856
similaritymatrix_025_30_4	5	10	15	43,517	21,19	38,499	38,4985	38,4985
similaritymatrix_025_30_5	10	10	10	36,675	20,535	30,61	35,6456	35,6456
similaritymatrix_025_30_5	5	10	15	39,312	29,136	29,136	36,575	39,2753
similaritymatrix_050_30_1	10	10	10	64,562	52,962	52,962	52,962	56,5745
similaritymatrix_050_30_1	5	10	15	70,112	53,575	53,575	56,6324	58,18
similaritymatrix_050_30_2	10	10	10	66,357	49,604	49,604	57,5471	57,5471
similaritymatrix_050_30_2	5	10	15	69,447	52,199	61,641	64,8644	64,8644
similaritymatrix_050_30_3	10	10	10	66,526	47,59	57,822	58,6872	59,2661
similaritymatrix_050_30_3	5	10	15	69,065	52,906	59,64	59,6402	65,3579
similaritymatrix_050_30_4	10	10	10	61,963	49,389	56,169	56,1688	56,1697
similaritymatrix_050_30_4	5	10	15	71,928	53,289	63,128	63,1277	64,7088
similaritymatrix_050_30_5	10	10	10	60,006	46,307	53,644	56,2984	56,2984
similaritymatrix_050_30_5	5	10	15	66,294	55,067	55,322	56,532	59,407
similaritymatrix_075_30_1	10	10	10	78,606	69,179	69,179	72,9083	73,7507
similaritymatrix_075_30_1	5	10	15	89,17	74,594	83,227	79,2164	83,2269
similaritymatrix_075_30_2	10	10	10	82,692	66,294	66,294	74,1313	75,0183
similaritymatrix_075_30_2	5	10	15	89,191	77,304	77,304	84,3798	86,5239
similaritymatrix_075_30_3	10	10	10	77,687	66,661	66,661	72,9682	72,9682
similaritymatrix_075_30_3	5	10	15	88,567	75,551	75,551	82,2657	82,2657
similaritymatrix_075_30_4	10	10	10	77,374	65,392	65,392	67,3835	67,7975
similaritymatrix_075_30_4	5	10	15	86,589	75,025	78,473	78,4725	79,2225
similaritymatrix_075_30_5	10	10	10	79,393	67,037	67,037	69,5716	70,1798
similaritymatrix_075_30_5	5	10	15	91,498	78,646	78,646	78,6459	78,6459

Similarly, to what was said previously, even in the case of K = 3 it shows greater differences in the larger instances and highlights the fact that giving more time to the solver is advantageous. Furthermore, it is possible to observe that it is not easy to make an objective judgment on which is the best choice of initial time limit. This is because the solver behaves differently for each type of instance. However, it can be said that 20 seconds is not much compared to larger instances and for this reason it was considered a good starting choice.

3.2.4 The results: LP BASED LOCAL SEARCH 2-DIVISION

After analysing the starting solutions, the LP BASED LOCAL SEARCH 2-DIVISION algorithm was tested (Table 19 and 20). The two parameters that have been evaluated to observe its quality are the time taken to obtain the solution and the relative error with respect to the objective function that was reached by the F6 model within the time limit of 7200 seconds:

$$relative\,error\,in\,OF\,(\%) = \frac{OF_{F6} - OF_{MET}}{OF_{F6}} \cdot 100$$

OF_{F6} represent the objective function of the model F6, while the OF_{MET} represent the objective function of the metaheuristic.

The results are reported in the table below:

Table 19, LP BASED LOCAL SEARCH 2-DIVISION for K=2.

d	∑Mk	Average relative error in OF (%)			
		S_0 = 3 s	S_0 = 5 s	S_0 = 10 s	S_0 = 20 s
25	**10**	0,00%	0,00%	0,00%	0,00%
25	**20**	2,44%	2,49%	2,07%	1,00%
25	**30**	1,63%	1,59%	1,24%	0,38%
50	**10**	0,67%	0,67%	0,00%	0,00%
50	**20**	2,31%	2,43%	2,28%	0,32%
50	**30**	1,57%	1,57%	1,21%	1,60%
75	**10**	0,15%	0,58%	0,00%	0,00%
75	**20**	2,90%	1,91%	1,98%	0,36%
75	**30**	1,11%	1,28%	1,60%	1,30%

Table 20, LP BASED LOCAL SEARCH 2-DIVISION for K=3.

d	$\sum M_k$	Average relative error in OF (%)			
		S_0 = 3 s	S_0 = 5 s	S_0 = 10 s	S_0 = 20 s
25	**10**	0,00%	0,00%	0,00%	0,00%
25	**20**	4,16%	3,39%	3,02%	1,82%
25	**30**	2,37%	2,04%	1,75%	1,43%
50	**10**	0,00%	0,00%	0,00%	0,00%
50	**20**	2,98%	3,45%	3,01%	2,31%
50	**30**	4,18%	3,68%	4,32%	4,44%
75	**10**	0,00%	0,00%	0,00%	0,00%
75	**20**	3,19%	2,37%	1,84%	1,17%
75	**30**	3,34%	3,07%	2,39%	2,74%

Observing the results, it can be seen that:

- In instances with sum $\sum_k M_k = 10$ the algorithm almost always manages to reach the solution of model F6 except for some cases in which the starting solution has been found for values ≤ 5.
- As expected, we find the best results in general starting from the initial solution found in 20 seconds.
- Starting from worse initial solutions it is possible to obtain better objective functions. This is because metaheuristics is not an exact algorithm, and it can easily come across a local maximum that is lower than what the algorithm finds starting from a better solution. Some examples are show in the tables (K = 3, d = 75, $\sum_k M_k = 30$).

Overall, it is possible to state that the algorithm guarantees good results starting from solutions generated in more than 5 seconds. The objective functions are in fact always below 5% and only in two cases above 3%. As previously stated, to evaluate the goodness of the algorithm it is also necessary to evaluate the time taken by the CPU. The times are shown in the following tables: 21 and 22.

Table 21, average CPU time for K=2.

d	∑Mk	F6 Model	Average CPU time (s)			
			S_0 = 3 s	S_0 = 5 s	S_0 = 10 s	S_0 = 20 s
25	**10**	2,43	3,42	3,26	3,40	3,34
25	**20**	28,31	5,60	8,98	10,43	17,19
25	**30**	47,17	3,95	7,56	7,99	12,83
50	**10**	3,61	3,17	5,82	4,55	4,55
50	**20**	84,95	4,33	10,42	12,21	20,89
50	**30**	214,89	4,93	8,39	12,02	21,74
75	**10**	2,79	2,96	5,34	3,48	3,45
75	**20**	150,07	4,24	10,81	11,17	21,48
75	**30**	515,16	5,54	9,77	12,66	22,11

Table 22, average CPU time for K=3.

d	∑Mk	F6 model	Average CPU (s)			
			S_0 = 3 s	S_0 = 5 s	S_0 = 10 s	S_0 = 20 s
25	**10**	1,40	1,36	1,83	1,63	1,61
25	**20**	121,63	4,62	5,95	11,16	20,82
25	**30**	147,82	3,14	4,39	7,44	12,93
50	**10**	1,21	1,16	1,28	1,50	1,46
50	**20**	309,49	4,23	7,20	12,99	21,11
50	**30**	2034,95	7,12	7,65	13,93	27,23
75	**10**	0,95	0,94	1,07	1,23	1,14
75	**20**	202,02	5,12	6,43	11,57	21,15
75	**30**	2765,99	6,98	8,67	14,26	23,65

Comparing the results obtained by the F6 model and those obtained with the metaheuristic, it can clearly see how there a clear reduction in the times has been. This is very noticeable for more complex and time-consuming instances. If we consider those in which the solver takes more than 2000 seconds using the F6 model, the reduction is 100 times smaller.

3.2.5 LP based local search 4-division

From the results reported in the previous chapter it was observed that the resolution times of the instances are much lower than those of the F6 model. For this reason, an attempt was made to further improve the relative error values obtained using computationally more expensive algorithms. To do this, the starting solutions obtained in 20 seconds are used by dividing the vector x [k] [] into 4 parts in such a way to have more free variables. The results are shown in Table 23 and 24.

Table 23, LP BASED LOCAL SEARCH 4-DIVISION for K=2.

d	∑Mk	Average CPU time (s)	Average relative error (%)
25	**10**	3,40	0,00%
25	**20**	24,31	0,16%
25	**30**	21,66	0,27%
50	**10**	4,58	0,00%
50	**20**	26,87	0,40%
50	**30**	52,68	1,00%
75	**10**	3,85	0,00%
75	**20**	62,78	0,45%
75	**30**	80,81	0,73%

Table 24, LP BASED LOCAL SEARCH 4-DIVISION for K=3.

d	∑Mk	Average CPU time (s)	Average relative error (%)
25	**10**	2,06	0,00%
25	**20**	32,05	1,57%
25	**30**	32,08	0,75%
50	**10**	1,85	0,00%
50	**20**	68,19	0,52%
50	**30**	126,62	2,16%
75	**10**	1,63	0,00%
75	**20**	61,04	0,67%
75	**30**	181,99	1,36%

Using the 4-division method, excellent results have been obtained: the relative error reaches its worst value in the case in which K = 3, $\sum_k M_k = 30$ and d = 50 and it is only 2.16%.

The times are obviously higher than those obtained in the case of the 2-division but in any case, they turn out to be very good in the more computationally longer cases.

3.3 Big instances

Considering the excellent results obtained on similarity instances, it was decided to test the algorithm on larger instances to evaluate its effectiveness. In this case the goal of metaheuristics is not only to get closer to the solution obtained in the time limit, but also to improve the results of the objective function by going to "surpass" the model proposed in the paper.

3.3.1 The Instances

The first step was to generate the instances. For this problem, 5 have been identified in the CEDRIC library that have N = 80. The procedure for the generation was the same used in paragraph 2.2.3. Also, in this case the names of the similarity matrices are characterized by their density value, $\sum_k M_k$ and the progressive number. The instances generated are the following:

- *similaritymatrix_075_20_1.*
- *similaritymatrix_075_40_2.*
- *similaritymatrix_075_60_3.*
- *similaritymatrix_050_40_4.*
- *similaritymatrix_050_60_5.*

After doing this, the values of Mk were matched to each instance, dividing the number of items for each cluster into equal parts. Obviously in the cases of K = 3 it was not always

possible to divide by 3 and for this reason a cluster can have one item more or less than the others.

3.3.2 Model F6: The results

As in the case of the smaller instances, the first step was to test the algorithm of the F6 model. The results are shown in Table 25 and 26.

Table 25,model F6 for K=2.

NAME	M1	M2	OF	AVERAGE CPU TIME (s)	AVERAGE GAP
similaritymatrix_075_20_1	10	10	62,6422	7200	8,70%
similaritymatrix_075_40_2	20	20	203,8060	7200	20,93%
similaritymatrix_075_60_3	30	30	398,3930	7200	19,48%
similaritymatrix_050_40_4	20	20	142,8530	7200	27,96%
similaritymatrix_050_60_5	30	30	304,4560	7200	18,88%

Table 26, model F6 for K=3.

NAME	M1	M2	M3	OF	AVERAGE CPU TIME (s)	AVERAGE GAP
similaritymatrix_075_20_1	6	7	7	44,5461	7200	6,26%
similaritymatrix_075_40_2	13	13	14	149,3230	7200	23,04%
similaritymatrix_075_60_3	20	20	20	271,3940	7200	29,81%
similaritymatrix_050_40_4	13	13	14	114,1330	7200	25,72%
similaritymatrix_050_60_5	20	20	20	214,1600	7200	31,85%

As expected, no instance was resolved within the 7200 second time limit and the average gaps, in some cases, exceed 30%. Only in cases where $\sum_k M_k = 20$ the average gap is lower than 10%.

3.3.3 The metaheuristic results

Considering the results obtained with the smaller instances, the algorithm LP BASED LOCAL SEARCH 4-DIVISION has been implemented also for the larger ones. The following features were used in the implementation:

- the starting solution is obtained after running the solver for 20 seconds.
- the subdivision of the vector x [k] [] is done in 4 equal parts.

Table 27, LP BASED LOCAL SEARCH 4-DIVISION for K=2.

NAME	M1	M2	OF	CPU TIME (s)
similaritymatrix_075_20_1	10	10	58,8765	22,313
similaritymatrix_075_40_2	20	20	198,4920	46,374
similaritymatrix_075_60_3	30	30	406,2620	58,279
similaritymatrix_050_40_4	20	20	150,2210	26,832
similaritymatrix_050_60_5	30	30	301,6170	44,211

Table 28, LP BASED LOCAL SEARCH 4-DIVISION for K=3.

NAME	M1	M2	M3	OF	CPU TIME (s)
similaritymatrix_075_20_1	6	7	7	44,5461	21,802
similaritymatrix_075_40_2	13	13	14	147,0790	59,059
similaritymatrix_075_60_3	20	20	20	298,2540	48,082
similaritymatrix_050_40_4	13	13	14	115,5800	98,586
similaritymatrix_050_60_5	20	20	20	227,9660	26,929

Observing the results in the table 27 and 29 it is possible to notice that for the instances highlighted in yellow, the objective function values are better than those found by the F6 model in the time limit. All the others, on the other hand, are close to or equal them with times even 100 times smaller than 7200 seconds.

3.3.4 Further improvements

The values of tables 27 and 28 are quite good but the question is whether it is possible to improve them further. To do this, a change has been added to the algorithm: in fact, it is run in a loop without stopping after evaluating the 4 subdivisions but until the objective function remains the same for 4 different steps. To clarify these are the steps:

1. Find the initial solution x0 by running the algorithm for 20 seconds and save the OF.

2. Initialize the counter = 0.
3. Fix the variables x[k][i]=1 for i ≤ 20 and run the solver .

   if *OF = OF previous* → counter = counter +1
   else → counter =0

   if *counter = 4* → **go** to point 7

4. Fix the variables x[k][i]=1 for i>20 and i ≤ 40 and run the solver

   if *OF = OF previous* → counter = counter +1
   else → counter =0

   if *counter = 4* → **go** to point 7

5. Fix the variables x[k][i]=1 for i>40 and i ≤ 60 and run the solver

if *OF = OF previous* → counter = counter +1

else → counter =0

if *counter = 4* → **go** to point 7

6. Fix the variables x[k][i]=1 for i>40 and i ≤ 60 and run the solver

if *OF = OF previous* → counter = counter +1

else → counter =0

if *counter = 4* → **go** to point 7

else → **go** to point 3

7. **Stop** the algorithm

The results are shown in the table below:

Table 29, LP BASED LOCAL SEARCH 4-DIVISION in loop for K=2.

NAME	M1	M2	OF	CPU TIME (s)
similaritymatrix_075_20_1	10	10	60,3101	30,461
similaritymatrix_075_40_2	20	20	204,6270	73,238
similaritymatrix_075_60_3	30	30	423,5250	110,039
similaritymatrix_050_40_4	20	20	152,4130	40,821
similaritymatrix_050_60_5	30	30	307,3940	82,968

Table 30, LP BASED LOCAL SEARCH 4-DIVISION in loop for K=3..

NAME	M1	M2	M3	OF	CPU (s)
similaritymatrix_075_20_1	6	7	7	44,5461	22,924
similaritymatrix_075_40_2	13	13	14	148,2880	66,044
similaritymatrix_075_60_3	20	20	20	305,1300	54,07
similaritymatrix_050_40_4	13	13	14	118,0550	103,554
similaritymatrix_050_60_5	20	20	20	234,1740	32,798

As expected, by performing more iterations of the algorithm it was possible to obtain improvements to the solution at the expense of a slight lengthening of the times. In cases where K = 2 *similaritymatrix_050_60_5* and *similaritymatrix_075_40_2* the algorithm managed to overcome the F6 model, which was not achieved in the single iteration.

Chapter 4

Conclusion

In this thesis, a metaheuristics model has been developed capable of reducing the time taken by the CPU to solve the MILP problems studied by Goncalves et Lourenco (2019). The implementation of the models on the new version of CPLEX has already led to an improvement in the results and it has been observed that only the most computationally complex instances need a metaheuristic model (K=2 and K=3). In fact, for K = 1, the worst case takes only 20 seconds to be solved. It was also possible to observe that the F6 model achieves better performance respect to F2 (contrary to what the study stated) and for this reason it was used as a reference model.

To obtain improvements, a metaheuristics LP based Local search algorithm was used which, starting from an initial solution, tries to improve it by selecting some variables and leaving free the other ones. The starting solutions were generated by running the solver for 3s, 5s, 10s and 20s and, in most cases, the objective function was better in cases where the solver was left running for longer. However, a better initial solution does not imply that the metaheuristic algorithm automatically reaches a better solution, even if this happens most of the time. In fact, in the first tested algorithm (2-DIVISION case, which divides the vector x [k] [i] into only two parts), the worst average relative error is 4,44%. It is reached starting from an initial solution of 20 s (case K=3, d=50 and $\sum_k M_k = 30$). In any case, the average time taken by the CPU to resolve the instances never exceeds 30 seconds, highlighting the strength of the algorithm. To try to further improve the difference between the optimal solutions reached in 7200 seconds and those obtained with metaheuristics, the instances were tested with a modification to the algorithm in which the vector x[k][i] was divided into 4 parts (4-DIVISION case) which involved fewer fixed variables but a greater waste of time. The worst-case of average relative error is 2.16% (case K=3, d=50 and $\sum_k M_k = 30$), while all others do not even reach 2%. The computationally more expensive instances (case K=3, d=75 and

$\sum_k M_k = 30$) take an average CPU time of 182 seconds which is still much better than the 2766 seconds taken by the F6 model on the same group of instances.

Considering the excellent results obtained, the metaheuristic algorithm was tested on larger instances which are more complicated for the solver, given the large number of variables. Also in this case, the F6 model was tested first but in no case the optimum was reached in the time limit of 7200 seconds. The best instance (K=3, d=25 and $\sum_k M_k = 20$) has in fact reached a gap of 6.26% with respect to the optimal value of the objective function, while in the worst case (K=3, d=50 and $\sum_k M_k = 60$) the gap is 31.85%. Using the 4-DIVISION technique (the same of the smaller instances) it was seen that the metaheuristics not only always took less than 100 seconds but, in 5 out of 10 cases, was able to obtain results that improve those of the MILP F6 model used in the Paper. Other further improvements were achieved by running the solver in a loop until it managed to grow the objective function and two more instances overcame the result obtained by the F6 model. In particular, it should be noted that metaheuristics manages to improve the results in the instances with $\sum_k M_k \geq 40$ and therefore computationally more complex, in fact 2 of the 3 instances that do not improve the model have $\sum_k M_k = 20$.

Appendix A – The instances

A.1 Instance 25_10_1

The instances were taken from the CEDRIC library. Each pair of numbers represents the link between two items.

1 4	3 28	7 26	10 30	15 23	18 34	22 36	29 34
1 9	3 38	7 27	10 32	15 27	18 37	23 24	29 36
1 12	3 39	7 30	11 18	15 32	18 39	23 28	29 38
1 16	4 8	7 31	11 22	15 33	19 21	23 31	30 33
1 23	4 9	7 33	11 24	15 34	19 23	23 35	30 34
1 25	4 21	7 35	11 27	15 36	19 27	23 37	30 36
1 27	4 24	7 38	11 29	15 37	19 29	24 27	31 33
1 28	4 32	8 11	11 31	15 38	19 34	24 30	31 40
1 31	4 33	8 19	11 32	16 20	19 37	24 31	32 34
1 38	4 34	8 20	12 14	16 23	19 38	24 38	32 37
1 39	5 15	8 29	12 21	16 26	20 21	25 30	32 38
1 40	5 19	8 36	12 28	16 31	20 28	25 32	32 40
2 20	5 26	8 39	12 32	16 37	20 31	25 33	33 39
2 26	5 33	9 14	13 20	17 25	20 34	25 37	33 40
2 33	5 38	9 23	13 25	17 29	20 37	26 27	34 39
2 36	6 18	9 29	13 32	17 32	20 39	26 31	34 40
2 40	6 25	9 31	14 18	17 37	20 40	26 34	35 38
3 4	6 28	9 32	14 19	17 38	21 24	26 37	35 39
3 8	6 33	9 34	14 27	17 39	21 40	27 30	36 37
3 9	7 19	9 38	14 28	17 40	22 23	27 33	38 39
3 10	7 20	9 39	14 32	18 20	22 25	27 34	38 40
3 13	7 22	9 40	14 37	18 28	22 27	27 38	
3 27	7 25	10 29	14 39	18 30	22 35	27 39	

A.2 MATLAB CODE: Edge graph to similarity matrix

The Matlab code below associates a random value that represents the similarity between two items to each pair of numbers in the edge graph.

```
density=[25 50 75];
sumM=[10 20 30];

for z=1:3
for w=1:3
for k=1:5

% Read the file into a matrix

filename                                                    =
['kcluster40_0',num2str(density(z)),'_',num2str(sumM(w
)),'_',num2str(k),'.dat'];
fp_orig=fopen(filename,'rt');


% Inizializzazione matrice "vuota"

M=[];


% Loop per la lettura del file di testo

while(1)


   % Lettura di una riga

     tline=fgetl(fp_orig);


   % Identificazione della fine del file ed uscita dal
loop di lettura

     if ~ischar(tline)
       break
     end
```

```
% Conversione della stringa in numero (scalare o
vettore)

   tmp=str2num(tline);


% Se la riga contiene solo numeri, la variabile "tmp"
conterrà i valori

% letti dal file i quali verranno assegnati alla matrice
"M"

   if(~isempty(tmp))
      M=[M;tmp];
   end
end


% Chiusura dei file

fclose(fp_orig);


% similarity matrix

s=zeros(40);
for i=2:length(M(:,1))

    s(M(i,1),M(i,2))=rand;
    s(M(i,2),M(i,1))=s(M(i,1),M(i,2));

end


% write the similarity matrix

filew=['Smatrix_0',num2str(density(z)),'_',num2str(sum
M(w)),'_',num2str(k),'_0',,'.txt'];
fileID=fopen(filew,'w');

for i=1:40
    for j=1:40
    fprintf(fileID,'%f ',s(i,j));
    if j==40
        fprintf(fileID,'\n');
    end
```

```
    end
end
end
end
end
```

A.3 similaritymatrix_025_10_1

Row 1: 0.000000 0.000000 0.000000 0.310679 0.000000 0.000000 0.000000 0.000000 0.408869 0.000000 0.000000 0.708011 0.000000 0.000000 0.000000 0.143638 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.871322 0.000000 0.083156 0.000000 0.461738 0.030389 0.000000 0.000000 0.753201 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.700043 0.214512 0.679905

Row 2: 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.557293 0.000000 0.000000 0.000000 0.000000 0.000000 0.850679 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.558565 0.000000 0.000000 0.901774 0.000000 0.000000 0.000000 0.419518

Row 3:0.000000 0.000000 0.000000 0.358128 0.000000 0.000000 0.000000 0.488988 0.255962 0.929169 0.000000 0.000000 0.466757 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.254008 0.431218 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.702530 0.402330 0.000000

Row 4: 0.310679 0.000000 0.358128 0.000000 0.000000 0.000000 0.000000 0.181840 0.856251 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.584201 0.000000 0.000000 0.373579 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.221695 0.218994 0.522232 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

Row 5: 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.433423 0.000000 0.000000 0.000000 0.741304 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.070450 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.847333 0.000000 0.000000 0.000000 0.000000 0.679880 0.000000 0.000000

Row: 6 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.136652 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.858402 0.000000 0.000000 0.199834 0.000000 0.000000 0.000000 0.000000 0.607340 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

Row 7: 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.543045 0.162325 0.000000 0.005653 0.000000 0.000000 0.771485 0.764788 0.421069 0.000000 0.000000 0.056813 0.585747 0.000000 0.174155 0.000000 0.728611 0.000000 0.000000 0.534291 0.000000 0.000000

Row 8: 0.000000 0.000000 0.488988 0.181840 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.253064 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.917057 0.758195 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.887031 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.068798 0.000000 0.000000 0.183528 0.000000

Row 9: 0.408869 0.000000 0.255962 0.856251 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.737073 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.696715 0.000000 0.000000 0.000000 0.000000 0.000000 0.776993 0.000000 0.501903 0.425497 0.000000 0.611237 0.000000 0.000000 0.000000 0.855772 0.670797 0.523592

Row 10: 0.000000 0.000000 0.929169 0.000000 0.298815 0.703969 0.000000 0.381611 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

Row 11: 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.253064 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.567685 0.000000 0.000000 0.000000 0.887861 0.000000 0.842949 0.000000 0.000000 0.898799 0.000000 0.939003 0.000000 0.815435 0.001358 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

Row 12: 0.708011 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.003091 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.087469 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.260727 0.000000 0.000000 0.000000 0.022799 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

Row 13: 0.000000 0.000000 0.466757 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.424085 0.000000 0.000000 0.000000 0.000000 0.341065 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.541354 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

Row 14: 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.737073 0.000000 0.000000 0.003091 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.926169 0.298499 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.338085 0.859480 0.000000 0.000000 0.000000 0.340478 0.000000 0.000000 0.000000 0.000000 0.138120 0.000000 0.507799 0.000000

Row 15: 0.000000 0.000000 0.000000 0.000000 0.433423 0.000000 0.856656 0.000000 0.000000 0.000000 0.384314 0.000000 0.000000 0.000000 0.000000 0.695691 0.627904 0.450388 0.000000 0.473618 0.949706 0.083498 0.000000 0.000000

Row 16: 0.143638 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.279829 0.000000 0.000000 0.447007 0.000000 0.000000 0.587571 0.000000 0.000000 0.000000 0.000000 0.000000 0.877634 0.000000 0.000000 0.000000 0.000000 0.000000 0.469101 0.000000 0.000000 0.000000

Row 17: 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.437418 0.000000
0.000000 0.000000 0.746185 0.000000 0.000000 0.467910 0.000000 0.000000 0.000000
0.000000 0.860827 0.466512 0.498104 0.487431

Row 18: 0.000000 0.000000 0.000000 0.000000 0.000000 0.136652 0.000000 0.000000
0.000000 0.000000 0.567685 0.000000 0.000000 0.926169 0.000000 0.000000 0.000000
0.000000 0.000000 0.229469 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.085552 0.000000 0.067383 0.000000 0.000000 0.000000 0.888391 0.000000
0.000000 0.233168 0.000000 0.861596 0.000000

Row 19: 0.000000 0.000000 0.000000 0.000000 0.741304 0.000000 0.543045 0.917057
0.000000 0.000000 0.000000 0.000000 0.000000 0.298499 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.711735 0.000000 0.872813 0.000000 0.000000 0.000000
0.938002 0.000000 0.139689 0.000000 0.000000 0.000000 0.000000 0.393900 0.000000
0.000000 0.980563 0.644794 0.000000 0.000000

Row 20: 0.000000 0.557293 0.000000 0.000000 0.000000 0.000000 0.162325 0.758195
0.000000 0.000000 0.000000 0.000000 0.424085 0.000000 0.000000 0.279829 0.000000
0.229469 0.000000 0.000000 0.896410 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.482230 0.000000 0.000000 0.014093 0.000000 0.000000 0.622880 0.000000
0.000000 0.231095 0.000000 0.527434 0.724992

Row 21: 0.000000 0.000000 0.000000 0.584201 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.087469 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.711735 0.896410 0.000000 0.000000 0.000000 0.607416 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.588366

Row 22: 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.005653 0.000000
0.000000 0.000000 0.887861 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.433435 0.000000 0.244173 0.000000
0.428960 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.010177
0.608821 0.000000 0.000000 0.000000 0.000000

Row 23: 0.871322 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.696715 0.000000 0.000000 0.000000 0.000000 0.000000 0.856656 0.447007 0.000000
0.000000 0.872813 0.000000 0.000000 0.433435 0.000000 0.957975 0.000000 0.000000
0.000000 0.095447 0.000000 0.000000 0.035591 0.000000 0.000000 0.000000 0.886235
0.000000 0.246941 0.000000 0.000000 0.000000

Row 24: 0.000000 0.000000 0.000000 0.373579 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.842949 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.607416 0.000000 0.957975 0.000000 0.000000 0.000000
0.008915 0.000000 0.000000 0.814920 0.140499 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.879866 0.000000 0.000000

Row 25: 0.083156 0.000000 0.000000 0.000000 0.000000 0.858402 0.771485 0.000000
0.000000 0.000000 0.000000 0.000000 0.341065 0.000000 0.000000 0.000000 0.437418
0.000000 0.000000 0.000000 0.000000 0.244173 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.095377 0.000000 0.352560 0.593421 0.000000 0.000000
0.000000 0.585182 0.000000 0.000000 0.000000

Row 26: 0.000000 0.850679 0.000000 0.000000 0.070450 0.000000 0.764788 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.587571 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.667682 0.000000 0.000000 0.000000 0.648027 0.000000 0.000000 0.433369 0.000000 0.000000 0.139759 0.000000 0.000000 0.000000

Row 27: 0.461738 0.000000 0.254008 0.000000 0.000000 0.000000 0.421069 0.000000 0.000000 0.000000 0.898799 0.000000 0.000000 0.338085 0.384314 0.000000 0.000000 0.000000 0.938002 0.000000 0.000000 0.428960 0.000000 0.008915 0.000000 0.667682 0.000000 0.000000 0.000000 0.751930 0.000000 0.000000 0.241787 0.650459 0.000000 0.000000 0.000000 0.857374 0.084370 0.000000

Row 28: 0.030389 0.000000 0.431218 0.000000 0.000000 0.199834 0.000000 0.000000 0.000000 0.000000 0.000000 0.260727 0.000000 0.859480 0.000000 0.000000 0.000000 0.085552 0.000000 0.482230 0.000000 0.000000 0.095447 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

Row 29: 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.887031 0.776993 0.298815 0.939003 0.000000 0.000000 0.000000 0.000000 0.000000 0.746185 0.000000 0.139689 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.972089 0.000000 0.031460 0.000000 0.835405 0.000000 0.000000

Row 30: 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.056813 0.000000 0.000000 0.703969 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.067383 0.000000 0.000000 0.000000 0.000000 0.000000 0.814920 0.095377 0.000000 0.751930 0.000000 0.000000 0.000000 0.000000 0.000000 0.835713 0.049858 0.000000 0.545886 0.000000 0.000000 0.000000 0.000000

Row 31: 0.753201 0.000000 0.000000 0.000000 0.000000 0.000000 0.585747 0.000000 0.501903 0.000000 0.815435 0.000000 0.000000 0.000000 0.000000 0.877634 0.000000 0.000000 0.000000 0.014093 0.000000 0.000000 0.035591 0.140499 0.000000 0.648027 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.943170 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.321473

Row 32: 0.000000 0.000000 0.000000 0.221695 0.000000 0.000000 0.000000 0.000000 0.425497 0.381611 0.001358 0.022799 0.541354 0.340478 0.695691 0.000000 0.467910 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.352560 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.806467 0.000000 0.000000 0.601399 0.789620 0.000000 0.799185

Row 33: 0.000000 0.558565 0.000000 0.218994 0.847333 0.607340 0.174155 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.627904 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.593421 0.000000 0.241787 0.000000 0.000000 0.835713 0.943170 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.049565 0.283199

Row 34: 0.000000 0.000000 0.000000 0.522232 0.000000 0.000000 0.000000 0.000000 0.611237 0.000000 0.000000 0.000000 0.000000 0.000000 0.450388 0.000000 0.000000 0.888391 0.393900 0.622880 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.433369 0.650459 0.000000 0.972089 0.049858 0.000000 0.806467 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.653457 0.489655

Row 35: 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.728611 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.010177 0.886235 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.972852 0.748490 0.567841

Row 36: 0.000000 0.901774 0.000000 0.000000 0.000000 0.000000 0.000000 0.068798 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.473618 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.608821 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.031460 0.545886 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.298964 0.000000 0.000000 0.000000

Row 37: 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.138120 0.949706 0.469101 0.860827 0.233168 0.980563 0.231095 0.000000 0.000000 0.246941 0.000000 0.585182 0.139759 0.000000 0.000000 0.000000 0.000000 0.000000 0.601399 0.000000 0.000000 0.000000 0.298964 0.000000 0.000000 0.000000 0.000000

Row 38: 0.700043 0.000000 0.702530 0.000000 0.679880 0.000000 0.534291 0.000000 0.855772 0.000000 0.000000 0.000000 0.000000 0.000000 0.083498 0.000000 0.466512 0.000000 0.644794 0.000000 0.000000 0.000000 0.000000 0.879866 0.000000 0.000000 0.857374 0.000000 0.835405 0.000000 0.000000 0.789620 0.000000 0.000000 0.972852 0.000000 0.000000 0.000000 0.256110 0.886564

Row 39: 0.214512 0.000000 0.402330 0.000000 0.000000 0.000000 0.000000 0.183528 0.670797 0.000000 0.000000 0.000000 0.000000 0.507799 0.000000 0.000000 0.498104 0.861596 0.000000 0.527434 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.084370 0.000000 0.000000 0.000000 0.000000 0.000000 0.049565 0.653457 0.748490 0.000000 0.000000 0.256110 0.000000 0.000000

Row 40: 0.679905 0.419518 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.523592 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.487431 0.000000 0.000000 0.724992 0.588366 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.321473 0.799185 0.283199 0.489655 0.567841 0.000000 0.000000 0.886564 0.000000 0.000000

Appendix B – The codes

B.1. F6 model

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include "ilcplex\ilocplex.h"
#include "ilcp\cp.h"

using namespace std;
const int N = 40;
vector<vector<double>> mat;

/*reading a matrix from a file*/
float** readMatrix(string name) {
    float mat[N][N];
    float** matrice = new float*[N];
    string filename;
    filename = name;

    ifstream dati(filename);

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (dati.eof())
                break;
            dati >> mat[i][j];
        }
    }
    dati.close();
    for (int i = 0; i < N; i++) {
        matrice[i] = new float[N];
        for (int j = 0; j < N; j++) {
            matrice[i][j] = mat[i][j];
        }
    }
    return matrice;
}
/*take the time*/
double getTime()
{
    time_t time1;
    double time2;
    time1 = clock();
    time2 = double(time1) / CLOCKS_PER_SEC;
    return time2;
};
```

```cpp
int main() {
    int M[3];
    string line;
    string num;
    string sim_name;
    double firstSolTime = 0.0;

    for (int K = 1; K < 4; K++) {
        ifstream myfile("C:\\Users\\Tommaso\\Desktop\\Istances\\istances.txt");
        ofstream result;

        if (K == 1)
            result.open("C:\\Users\\Tommaso\\Desktop\\Results\\10F6_model_results_K=1.txt");
        if (K == 2)
            result.open("C:\\Users\\Tommaso\\Desktop\\Results\\10F6_model_results_K=2.txt");
        if (K == 3)
            result.open("C:\\Users\\Tommaso\\Desktop\\Results\\10F6_model_results_K=3.txt");


        if (myfile.is_open()) {

            while (getline(myfile, line)) {

                for (int par = 0; par < 2; par++) {
                    if (K == 1)
                        par++;

                    double startTime = getTime();

                    // To take the value of sum(M(k)) from the name of the file
                    sim_name = line.substr(39, 25);
                    num = sim_name.substr(21, 2);

                    if (K == 1)
                        M[0] = stoi(num);

                    if (K == 2) {
                        if (par == 0) {
                            if (stoi(num) == 10) {
                                M[0] = 5;
                                M[1] = 5;
                            }
                            if (stoi(num) == 20) {
                                M[0] = 10;
                                M[1] = 10;
                            }
                            if (stoi(num) == 30) {
                                M[0] = 15;
                                M[1] = 15;
                            }
                        }
```

```
                if (par == 1) {
                    if (stoi(num) == 10) {
                        M[0] = 2;
                        M[1] = 8;
                    }
                    if (stoi(num) == 20) {
                        M[0] = 4;
                        M[1] = 16;
                    }
                    if (stoi(num) == 30) {
                        M[0] = 24;
                        M[1] = 6;
                    }
                }
            }

            if (K == 3) {
                if (par == 0) {
                    if (stoi(num) == 10) {
                        M[0] = 3;
                        M[1] = 3;
                        M[2] = 4;
                    }
                    if (stoi(num) == 20) {
                        M[0] = 7;
                        M[1] = 7;
                        M[2] = 6;
                    }
                    if (stoi(num) == 30) {
                        M[0] = 10;
                        M[1] = 10;
                        M[2] = 10;
                    }
                }

                if (par == 1) {
                    if (stoi(num) == 10) {
                        M[0] = 2;
                        M[1] = 3;
                        M[2] = 5;
                    }
                    if (stoi(num) == 20) {
                        M[0] = 3;
                        M[1] = 7;
                        M[2] = 10;
                    }
                    if (stoi(num) == 30) {
                        M[0] = 5;
                        M[1] = 10;
                        M[2] = 15;
                    }
                }
            }
```

```
cout << "\n\n" << sim_name;
cout << "  " << M[0] << " " << M[1] << " " << M[2] << "\n\n";

float** mat = readMatrix(line);

// To write the similarity matrix
/*for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        cout << mat[i][j] << " ";
    }
    cout << "\n";
}*/


// IMPLEMENTATION OF THE MODEL IN CPLEX

IloEnv env;             // Cplex environment and model init
IloModel model(env);

typedef IloArray <IloBoolVarArray> IloBoolVarMatrix;

// 3-dimensional matrix of boolean variables
typedef IloArray <IloBoolVarMatrix> IloBoolVar3Matrix;

IloBoolVar3Matrix y(env, K); // definition of 3Matrix y
for (int k = 0; k < K; k++) {
    y[k] = IloBoolVarMatrix(env, N);
    for (int i = 0; i < N; i++) {
        y[k][i] = IloBoolVarArray(env, N);
    }
}

IloBoolVarMatrix x(env, K); // and matrix x.
for (int k = 0; k < K; k++) {
    x[k] = IloBoolVarArray(env, N);
}


//CONSTRAINTS

//#1
for (int k = 0; k < K; k++) {
    for (int j = 0; j < N; j++) {
        for (int i = 0; i < j; i++) {
            model.add(y[k][i][j] >= x[k][i] + x[k][j] - 1);
        }
    }
}
```

```
//#2
for (int i = 0; i < N; i++) {
    IloExpr expr1(env);
    for (int k = 0; k < K; k++) {
        expr1 += x[k][i];
    }
    IloConstraint c1(expr1 <= 1);
    model.add(c1);
}

//#3
for (int k = 0; k < K; k++) {
    IloExpr expr2(env);
    for (int i = 0; i < N; i++) {
        expr2 += x[k][i];
    }
    IloConstraint c2(expr2 == M[k]);
    model.add(c2);
}

//#4
for (int j = 0; j < N; j++) {
    for (int k = 0; k < K; k++) {
        IloExpr expr3_1(env);
        IloExpr expr3_2(env);
        IloExpr expr3(env);
        for (int i = 0; i < j; i++) {
            expr3_1 += y[k][i][j];
        }
        for (int i = j + 1; i < N; i++) {
            expr3_2 += y[k][j][i];
        }
        expr3 = expr3_1 + expr3_2;
        IloConstraint c3(expr3 == (M[k] - 1) * x[k][j]);
        model.add(c3);
    }
}

//#5 model F6
for (int k = 0; k < K; k++) {
    for (int j = 0; j < N; j++) {
        for (int i = 0; i < j; i++) {
            model.add(y[k][i][j] <= x[k][i]);
        }
    }
}
```

```
//#6 model F6
for (int k = 0; k < K; k++) {
    for (int j = 0; j < N; j++) {
        for (int i = 0; i < j; i++) {
            model.add(y[k][i][j] <= x[k][j]);
        }
    }
}

//#OF
IloExpr OF(env);
for (int k = 0; k < K; k++) {
    for (int i = 0; i < (N - 1); i++) {
        for (int j = i + 1; j < N; j++) {
            OF += mat[i][j] * y[k][i][j];
        }
    }
}
model.add(IloMaximize(env, OF));

// For exporting LP files - y[k][i][j] and x[k][j]

/*for (int k = 0; k < y.getSize(); k++)
    for (int i = 0; i < y[k].getSize(); i++)
        for (int j = 0; j < y[k][i].getSize(); j++) {

            std::string tmpName = "y(" + std::to_string(k) + ")
            (" + std::to_string(i) + ")(" + std::to_string(j) + ")";
            y[k][i][j].setName(tmpName.c_str());
        }

for (int k = 0; k < x.getSize(); k++)
    for (int j = 0; j < x[k].getSize(); j++) {
        std::string tmpName = "x(" + std::to_string(k) + ")(" + std::to_string(j) + ")";
        x[k][j].setName(tmpName.c_str());
    }*/

    // cplex optimizer init
IloCplex  cplex(model);

// cplex parameters (time limit)
cplex.setParam(IloCplex::Param::TimeLimit, 7200);

// cplex parameters (avoid cplex textual output)
//cplex.setOut(env.getNullStream());

// how to excport the model in a lp file
cplex.exportModel("out.lp");

// RUN CPLEX
cplex.solve();
```

```
// Output
cout << "*** Cplex status = " << cplex.getCplexStatus() << " ";
float cplexSolVal = float(cplex.getObjValue());
cout << "OF = " << cplexSolVal;
cout << endl;

//Get the values of x and y
for (int k = 0; k < K; k++) {
    for (int i = 0; i < N; i++) {
        cout << cplex.getValue(x[k][i]);
        cout << " ";
        if (i == (N - 1))
            cout << "\n";
    }
}
cout << "\n";

/* for (int k = 0; k < K; k++) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (cplex.isExtracted(y[k][i][j])) {
                cout << cplex.getValue(y[k][i][j]);
                cout << " ";
            }
            else
                cout << "NA ";
            if (j == (N - 1))
                cout << "\n";
        }
    }
}*/
cout << "\n";

// To compute average gap %
IloNum objval = cplex.getObjValue();
IloNum bound = cplex.getBestObjValue();
IloNum gap = fabs(objval - bound) / (1.0 + fabs(bound)) * 100.0;

firstSolTime = getTime() - startTime;

if (K ==1)
result << sim_name << "  " << M[0] << " " << cplexSolVal << "  " <<
    cplex.getCplexStatus() << " " << firstSolTime << " " << gap << "\n";
if (K==2)
result << sim_name << "  " << M[0] << " " << M[1] << " " << cplexSolVal <<
    "  " << cplex.getCplexStatus() << " " << firstSolTime << " " << gap << "\n";
if (K==3)
result << sim_name << "  " << M[0] << " " << M[1] << " " << M[2] << " " <<
    cplexSolVal << "  " << cplex.getCplexStatus() << " " << firstSolTime << " " << gap << "\n";
```

```
                    // Everything must be closed (cplex, model, environment)
                    cplex.end();
                    model.end();
                    env.end();

                }
            }
        }

        else cout << "Unable to open file";

        myfile.close();
        result.close();
    }
}
```

B.2. Metaheuristic 4-DIVISION loop

```
while (a <= end) {
    cout << "\n\n" << sim_name;
    cout << "  " << M[0] << " " << M[1] << " " << M[2] << "   a = " << a << "\n\n";

    float** mat = readMatrix(line);

    // To write the similarity matrix
    /*for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            cout << mat[i][j] << " ";
        }
        cout << "\n";
    }*/

    // IMPLEMENTATION OF THE MODEL IN CPLEX

    IloEnv env;             // Cplex environment and model init
    IloModel model(env);

    typedef IloArray <IloBoolVarArray> IloBoolVarMatrix;
    // 3-dimensional matrix of boolean variables
    typedef IloArray <IloBoolVarMatrix> IloBoolVar3Matrix;

    IloBoolVar3Matrix y(env, K); // definition of 3Matrix y
    for (int k = 0; k < K; k++) {
        y[k] = IloBoolVarMatrix(env, N);
        for (int i = 0; i < N; i++) {
            y[k][i] = IloBoolVarArray(env, N);
        }
    }

    IloBoolVarMatrix x(env, K); // and matrix x.
    for (int k = 0; k < K; k++) {
        x[k] = IloBoolVarArray(env, N);
    }


    //CONSTRAINTS
    //#1
    for (int k = 0; k < K; k++) {
        for (int j = 0; j < N; j++) {
            for (int i = 0; i < j; i++) {
                model.add(y[k][i][j] >= x[k][i] + x[k][j] - 1);
            }
        }
    }
```

```
//#2
for (int i = 0; i < N; i++) {
    IloExpr expr1(env);
    for (int k = 0; k < K; k++) {
        expr1 += x[k][i];
    }
    IloConstraint c1(expr1 <= 1);
    model.add(c1);
}

//#3
for (int k = 0; k < K; k++) {
    IloExpr expr2(env);
    for (int i = 0; i < N; i++) {
        expr2 += x[k][i];
    }
    IloConstraint c2(expr2 == M[k]);
    model.add(c2);
}

//#4
for (int j = 0; j < N; j++) {
    for (int k = 0; k < K; k++) {
        IloExpr expr3_1(env);
        IloExpr expr3_2(env);
        IloExpr expr3(env);
        for (int i = 0; i < j; i++) {
            expr3_1 += y[k][i][j];
        }
        for (int i = j + 1; i < N; i++) {
            expr3_2 += y[k][j][i];
        }
        expr3 = expr3_1 + expr3_2;

        IloConstraint c3(expr3 == (M[k] - 1) * x[k][j]);
        model.add(c3);
    }
}

//#5 model F6
for (int k = 0; k < K; k++) {
    for (int j = 0; j < N; j++) {
        for (int i = 0; i < j; i++) {
            model.add(y[k][i][j] <= x[k][i]);
        }
    }
}
```

```
//#6 model F6
for (int k = 0; k < K; k++) {
    for (int j = 0; j < N; j++) {
        for (int i = 0; i < j; i++) {
            model.add(y[k][i][j] <= x[k][j]);
        }
    }
}

//metaeuristica
if (start != 0) {
    for (int k = 0; k < K; k++) {
        for (int i = 0; i < N; i++) {



            if (b == 0) {
                if (i > N * 1 / 4 && a == 1 && xcurrent[k][i] == 1)
                    model.add(x[k][i] == xcurrent[k][i]);

                if ((i <= N * 1 / 4 || i > N * 2 / 4) && a == 2 && xcurrent[k][i] == 1)
                    model.add(x[k][i] == xcurrent[k][i]);

                if ((i <= N * 2 / 4 || i > N * 3 / 4) && a == 3 && xcurrent[k][i] == 1)
                    model.add(x[k][i] == xcurrent[k][i]);

                if (i < N * 3 / 4 && a == 4 && xcurrent[k][i] == 1) {
                    model.add(x[k][i] == xcurrent[k][i]);

                }
            }

            if (b == 1) {
                if ((i <= N * 1 / 8 || i > N * 3 / 8) && a == 1 && xcurrent[k][i] == 1)
                    model.add(x[k][i] == xcurrent[k][i]);

                if ((i <= N * 3 / 8 || i > N * 5 / 8) && a == 2 && xcurrent[k][i] == 1)
                    model.add(x[k][i] == xcurrent[k][i]);

                if ((i <= N * 5 / 8 || i > N * 7 / 8) && a == 3 && xcurrent[k][i] == 1)
                    model.add(x[k][i] == xcurrent[k][i]);

                if ((i <= N * 7 / 8 || i > N * 1 / 8) && a == 4 && xcurrent[k][i] == 1) {
                    model.add(x[k][i] == xcurrent[k][i]);

                }
            }
        }
    }
}
```

```
//#OF
IloExpr OF(env);
for (int k = 0; k < K; k++) {
    for (int i = 0; i < (N - 1); i++) {
        for (int j = i + 1; j < N; j++) {
            OF += mat[i][j] * y[k][i][j];
        }
    }
}
model.add(IloMaximize(env, OF));

// For exporting LP files - y[k][i][j] and x[k][j]
/*for (int k = 0; k < y.getSize(); k++)
    for (int i = 0; i < y[k].getSize(); i++)
        for (int j = 0; j < y[k][i].getSize(); j++) {

            std::string tmpName = "y(" + std::to_string(k) + ")
            (" + std::to_string(i) + ")(" + std::to_string(j) + ")";
            y[k][i][j].setName(tmpName.c_str());
        }

for (int k = 0; k < x.getSize(); k++)
    for (int j = 0; j < x[k].getSize(); j++) {
        std::string tmpName = "x(" + std::to_string(k) + ")(" + std::to_string(j) + ")";
        x[k][j].setName(tmpName.c_str());
    }*/

    // cplex optimizer init
IloCplex  cplex(model);

// cplex parameters (time limit)
if (start == 0) {
    cplex.setParam(IloCplex::Param::TimeLimit, 20);
}
else {
    cplex.setParam(IloCplex::Param::TimeLimit, 7200);
}

// cplex parameters (avoid cplex textual output)
//cplex.setOut(env.getNullStream());

// how to excport the model in a lp file
cplex.exportModel("out.lp");

// RUN CPLEX
cplex.solve();

// Output
float cplexCurrentVal = float(cplex.getObjValue());
```

```
cout << "*** Cplex status = " << cplex.getCplexStatus() << " ";
cout << "OF = " << cplexCurrentVal;
cout << endl;


if (cplexBestVal == cplexCurrentVal) {
    loop++;
    /*if (loop == end_loop) {
        if (b == 0) {
            b = 1;
            loop = 0;
        }
        else {
            b = 0;
            loop = 0;
        }
    }*/
    if (loop == end_loop)
        a = end;
}
else {
    loop = 0;
}

if (start == 0) {
    start = 1;
    a = 0;
    cplexBestVal = cplexCurrentVal;
    for (int k = 0; k < K; k++) {
        for (int i = 0; i < N; i++) {
            xcurrent[k][i] = cplex.getValue(x[k][i]);
        }
    }
}

if (start == 1 && cplexBestVal <= cplexCurrentVal) {
    cplexBestVal = cplexCurrentVal;
    if (a == 4 && loop != end_loop)
        a = 0;
    for (int k = 0; k < K; k++) {
        for (int i = 0; i < N; i++) {
            xcurrent[k][i] = cplex.getValue(x[k][i]);
        }
    }
}
```

```
    //Get the values of x and y and print
    for (int k = 0; k < K; k++) {
        for (int i = 0; i < N; i++) {
            cout << cplex.getValue(x[k][i]);
            cout << " ";
            xcurrent[k][i] = cplex.getValue(x[k][i]);
            if (i == (N - 1))
                cout << "\n";
        }
    }

    cout << "\n";

    //To estract y[k][i][j]
    /* for (int k = 0; k < K; k++) {
         for (int i = 0; i < N; i++) {
             for (int j = 0; j < N; j++) {

                 if (cplex.isExtracted(y[k][i][j])) {
                     cout << cplex.getValue(y[k][i][j]);
                     cout << " ";
                 }
                 else
                     cout << "NA ";

                 if (j == (N - 1))
                     cout << "\n";
             }
         }
     }*/

    cout << "\n";

    if (a == end) {
        //To compute average gap %
        IloNum objval = cplex.getObjValue();
        IloNum bound = cplex.getBestObjValue();
        IloNum gap = fabs(objval - bound) / (1.0 + fabs(bound)) * 100.0;

        firstSolTime = getTime() - startTime;

        if (K == 1)
            result << sim_name << "  " << M[0] << " " << cplexCurrentVal <<
            "  " << cplex.getCplexStatus() << " " << firstSolTime << " " << gap << "\n";
        if (K == 2)
            result << sim_name << "  " << M[0] << " " << M[1] << " " << cplexCurrentVal << "  " <<
            cplex.getCplexStatus() << " " << firstSolTime << " " << gap << "\n";
        if (K == 3)
            result << sim_name << "  " << M[0] << " " << M[1] << " " << M[2] << " " <<
            cplexCurrentVal << "  " << cplex.getCplexStatus() << " " << firstSolTime << " " << gap << "\n";
    }


    // Everything must be closed (cplex, model, environment)
    cplex.end();
    model.end();
    env.end();
    a++;
}
```

Appendix C – The results

C.1. CPU time for K=2

				2-DIVISION				4-DIVISION
NAME	M_1	M_2	CPU time F6 (s)	CPU time (s), S_0 = 3 s	CPU time (s), S_0 = 5 s	CPU time (s), S_0 = 10 s	CPU time (s), S_0 = 20 s	CPU time (s), S_0 = 20 s
similaritymatrix_025_10_1	5	5	3,157	5,042	4,792	4,168	4,169	4,763
similaritymatrix_025_10_1	2	8	1,001	5,179	1,646	1,794	1,8	2,031
similaritymatrix_025_10_2	5	5	3,262	3,688	4,632	5,115	4,974	4,781
similaritymatrix_025_10_2	2	8	1,858	3,304	2,681	2,878	2,856	2,811
similaritymatrix_025_10_3	5	5	3,538	3,355	4,349	4,625	4,535	4,477
similaritymatrix_025_10_3	2	8	2,228	2,911	2,737	2,954	2,816	3,116
similaritymatrix_025_10_4	5	5	2,805	3,299	3,41	3,672	3,617	3,468
similaritymatrix_025_10_4	2	8	0,974	1,831	1,553	1,556	1,478	1,611
similaritymatrix_025_10_5	5	5	3,879	3,344	4,578	4,991	4,931	4,396
similaritymatrix_025_10_5	2	8	1,599	2,269	2,186	2,22	2,233	2,584
similaritymatrix_050_10_1	5	5	1,216	1,803	1,737	1,801	1,761	2,198
similaritymatrix_050_10_1	2	8	2,724	3,268	3,348	3,514	3,543	3,377
similaritymatrix_050_10_2	5	5	6,108	3,336	5,362	7,814	7,804	7,777
similaritymatrix_050_10_2	2	8	4,133	3,406	5,67	5,309	5,221	6,3
similaritymatrix_050_10_3	5	5	4,453	3,344	7,124	5,416	5,367	4,635
similaritymatrix_050_10_3	2	8	4,16	3,254	7,544	4,816	5,009	4,47
similaritymatrix_050_10_4	5	5	4,166	3,285	7,586	5,09	5,124	4,432
similaritymatrix_050_10_4	2	8	2,918	3,325	7,295	3,719	3,774	3,648
similaritymatrix_050_10_5	5	5	3,721	3,278	8,359	4,758	4,714	5,334
similaritymatrix_050_10_5	2	8	2,487	3,384	4,206	3,225	3,232	3,668
similaritymatrix_075_10_1	5	5	3,003	3,557	5,618	3,929	3,946	4,022
similaritymatrix_075_10_1	2	8	1,706	2,135	2,18	2,174	2,178	2,402
similaritymatrix_075_10_2	5	5	3,751	3,263	5,462	4,603	4,675	4,551
similaritymatrix_075_10_2	2	8	2,389	3,044	4,596	2,938	2,912	4,914
similaritymatrix_075_10_3	5	5	4,081	3,342	5,646	4,949	4,957	4,71
similaritymatrix_075_10_3	2	8	1,438	1,836	5,651	1,794	1,764	2,076
similaritymatrix_075_10_4	5	5	3,352	3,343	7,443	4,146	4,063	4,934
similaritymatrix_075_10_4	2	8	4,08	3,381	8,315	4,919	4,787	4,785
similaritymatrix_075_10_5	5	5	2,163	3,106	2,769	2,889	2,701	2,524
similaritymatrix_075_10_5	2	8	1,952	2,617	5,695	2,496	2,516	3,607
similaritymatrix_025_20_1	10	10	23,122	4,427	8,527	10,766	20,843	24,84
similaritymatrix_025_20_1	4	16	5,104	3,409	7,94	6,208	5,949	5,75
similaritymatrix_025_20_2	10	10	63,882	11,858	9,616	12,062	20,957	43,186
similaritymatrix_025_20_2	4	16	16,498	4,11	8,397	10,674	19,013	21,267
similaritymatrix_025_20_3	10	10	75,767	9,871	8,613	11,2	20,899	33,186
similaritymatrix_025_20_3	4	16	35,23	6,639	12,232	10,714	20,734	43,322
similaritymatrix_025_20_4	10	10	22,364	3,95	10,433	10,799	20,359	23,892
similaritymatrix_025_20_4	4	16	10,06	3,562	7,883	9,832	9,351	11,942
similaritymatrix_025_20_5	10	10	18,345	4,496	8,372	10,876	19,68	21,872
similaritymatrix_025_20_5	4	16	12,704	3,669	7,776	11,209	14,112	13,88
similaritymatrix_050_20_1	10	10	74,923	5,005	22,845	15,721	21,496	32,566
similaritymatrix_050_20_1	4	16	22,357	3,93	8,283	11,699	21,053	24,514
similaritymatrix_050_20_2	10	10	43,867	5,114	8,575	12,552	20,655	27,197
similaritymatrix_050_20_2	4	16	28,275	5,37	15,767	12,262	20,725	28,662

similaritymatrix_050_20_3	10	10	294,19	4,237	6,61	12,265	20,953	24,198
similaritymatrix_050_20_3	4	16	20,633	3,406	7,891	10,758	20,729	21,214
similaritymatrix_050_20_4	10	10	233,355	4,413	8,845	11,705	20,96	29,817
similaritymatrix_050_20_4	4	16	23,956	3,973	8,76	11,509	20,888	32,517
similaritymatrix_050_20_5	10	10	86,303	3,962	8,565	11,856	20,722	26,983
similaritymatrix_050_20_5	4	16	21,656	3,876	8,041	11,724	20,698	21,076
similaritymatrix_075_20_1	10	10	293,146	3,602	8,128	11,423	20,965	273,212
similaritymatrix_075_20_1	4	16	34,58	3,825	9,014	11,867	23,575	52,386
similaritymatrix_075_20_2	10	10	282,788	4,079	8,691	11,639	21,246	33,914
similaritymatrix_075_20_2	4	16	28,18	3,879	8,179	11,689	21,039	32,401
similaritymatrix_075_20_3	10	10	187,318	5,418	9,532	10,581	21,01	81,45
similaritymatrix_075_20_3	4	16	24,806	5,337	29,76	10,919	20,938	34,284
similaritymatrix_075_20_4	10	10	305,647	4,111	9,107	10,956	20,872	26,265
similaritymatrix_075_20_4	4	16	22,669	4,18	8,547	10,919	21,054	43,254
similaritymatrix_075_20_5	10	10	293,32	4,102	8,173	10,932	22,887	26,679
similaritymatrix_075_20_5	4	16	28,23	3,891	9,005	10,728	21,192	23,978
similaritymatrix_025_30_1	15	15	54,496	3,932	8,316	10,55	20,943	30,011
similaritymatrix_025_30_1	24	6	9,799	3,395	8,025	10,092	10,995	15,673
similaritymatrix_025_30_2	15	15	149,89	5,619	9,859	11,155	21,147	36,258
similaritymatrix_025_30_2	24	6	27,895	3,418	6,921	10,481	20,662	27,661
similaritymatrix_025_30_3	15	15	89,787	4,68	9,43	12,003	21,126	31,22
similaritymatrix_025_30_3	24	6	35,971	3,574	7,829	10,61	20,508	25,04
similaritymatrix_025_30_4	15	15	91,784	3,884	8,337	11,021	20,608	26,711
similaritymatrix_025_30_4	24	6	24,462	3,599	7,927	10,704	20,938	23,379
similaritymatrix_025_30_5	15	15	85,85	3,545	8,044	10,348	20,475	26,405
similaritymatrix_025_30_5	24	6	10,131	3,658	7,888	10,336	11,405	15,502
similaritymatrix_050_30_1	15	15	295,352	5,243	9,632	14,84	25,379	64,617
similaritymatrix_050_30_1	24	6	32,359	3,82	6,654	10,576	20,689	25,371
similaritymatrix_050_30_2	15	15	503,896	6,037	10,037	12,185	22,525	88,359
similaritymatrix_050_30_2	24	6	110,98	6,007	7,284	10,801	20,85	26,05
similaritymatrix_050_30_3	15	15	275,598	4,14	7,357	10,973	21,597	68,041
similaritymatrix_050_30_3	24	6	47,007	3,872	8,392	10,863	20,914	30,604
similaritymatrix_050_30_4	15	15	365,912	5,004	8,999	11,451	21,486	100,553
similaritymatrix_050_30_4	24	6	49,583	4,542	8,371	10,859	20,716	33,079
similaritymatrix_050_30_5	15	15	426,482	5,959	10,32	16,654	22,163	63,245
similaritymatrix_050_30_5	24	6	41,755	4,643	6,805	11,02	21,084	26,89
similaritymatrix_075_30_1	15	15	1285,74	5,582	10,057	12,762	22,873	107,279
similaritymatrix_075_30_1	24	6	68,678	4,293	6,555	11,526	20,997	35,485
similaritymatrix_075_30_2	15	15	418,272	5,751	10,224	11,885	22,296	60,98
similaritymatrix_075_30_2	24	6	41,023	3,63	6,473	11,364	21,066	24,719
similaritymatrix_075_30_3	15	15	1021,76	7,646	15,931	15,768	21,913	209,039
similaritymatrix_075_30_3	24	6	58,95	5,752	9,892	10,831	20,718	44,847
similaritymatrix_075_30_4	15	15	1268,12	5,28	7,688	12,674	23,343	93,853
similaritymatrix_075_30_4	24	6	45,911	3,742	6,401	10,987	21,297	37,163
similaritymatrix_075_30_5	15	15	916,145	9,399	17,548	16,956	25,672	165,767
similaritymatrix_075_30_5	24	6	27,029	4,322	6,908	11,8	20,928	28,933

C.2. OF for K=2

				OF S_0				2-DIVISION				4-DIVISION
NAME	M_1	M_2	OF F6	OF 3 s	OF 5 s	OF 10 s	OF 20 s	OF, S_0 = 3 s	OF, S_0 = 5 s	OF, S_0 = 10 s	OF, S_0 = 20 s	OF, S_0 = 20 s
similaritymatrix_025_10_1	5	5	11,4966	11,3021	11,4966	11,4966	11,4966	11,4966	11,4966	11,4966	11,4966	11,4966
similaritymatrix_025_10_1	2	8	13,9991	13,9991	13,9991	13,9991	13,9991	13,9991	13,9991	13,9991	13,9991	13,9991
similaritymatrix_025_10_2	5	5	12,0583	12,0576	12,0583	12,0583	12,0583	12,0583	12,0583	12,0583	12,0583	12,0583
similaritymatrix_025_10_2	2	8	13,4004	13,4004	13,4004	13,4004	13,4004	13,4004	13,4004	13,4004	13,4004	13,4004
similaritymatrix_025_10_3	5	5	11,9049	11,9049	11,9049	11,9049	11,9049	11,9049	11,9049	11,9049	11,9049	11,9049
similaritymatrix_025_10_3	2	8	12,6588	12,6588	12,6588	12,6588	12,6588	12,6588	12,6588	12,6588	12,6588	12,6588
similaritymatrix_025_10_4	5	5	12,1112	12,1112	12,1112	12,1112	12,1112	12,1112	12,1112	12,1112	12,1112	12,1112
similaritymatrix_025_10_4	2	8	13,653	13,653	13,653	13,653	13,653	13,653	13,653	13,653	13,653	13,653
similaritymatrix_025_10_5	5	5	11,5467	11,5467	11,5467	11,5467	11,5467	11,5467	11,5467	11,5467	11,5467	11,5467
similaritymatrix_025_10_5	2	8	13,2329	13,2329	13,2329	13,2329	13,2329	13,2329	13,2329	13,2329	13,2329	13,2329
similaritymatrix_050_10_1	5	5	15,228	15,228	15,228	15,228	15,228	15,228	15,228	15,228	15,228	15,228
similaritymatrix_050_10_1	2	8	16,3261	16,3261	16,3261	16,3261	16,3261	16,3261	16,3261	16,3261	16,3261	16,3261
similaritymatrix_050_10_2	5	5	14,1292	14,082	14,082	14,1292	14,1292	14,082	14,082	14,1292	14,1292	14,1292
similaritymatrix_050_10_2	2	8	16,7009	16,5117	16,7009	16,7009	16,7009	16,5117	16,5117	16,7009	16,7009	16,7009
similaritymatrix_050_10_3	5	5	14,9242	14,6181	14,9242	14,9242	14,9242	14,703	14,703	14,9242	14,9242	14,9242
similaritymatrix_050_10_3	2	8	17,4663	17,1367	17,4663	17,4663	17,4663	17,1367	17,1367	17,4663	17,4663	17,4663
similaritymatrix_050_10_4	5	5	14,4746	14,461	14,4746	14,4746	14,4746	14,4746	14,4746	14,4746	14,4746	14,4746
similaritymatrix_050_10_4	2	8	17,4068	17,1917	17,4068	17,4068	17,4068	17,1917	17,1917	17,4068	17,4068	17,4068
similaritymatrix_050_10_5	5	5	15,0162	14,9146	15,0162	15,0162	15,0162	14,9146	14,9146	15,0162	15,0162	15,0162
similaritymatrix_050_10_5	2	8	17,648	17,648	17,648	17,648	17,648	17,648	17,648	17,648	17,648	17,648
similaritymatrix_075_10_1	5	5	16,3946	16,3946	16,3946	16,3946	16,3946	16,3946	16,2761	16,3946	16,3946	16,3946
similaritymatrix_075_10_1	2	8	20,5697	20,5697	20,5697	20,5697	20,5697	20,5697	20,5697	20,5697	20,5697	20,5697
similaritymatrix_075_10_2	5	5	16,2103	16,2103	16,2103	16,2103	16,2103	16,2103	15,9207	16,2103	16,2103	16,2103
similaritymatrix_075_10_2	2	8	20,1843	20,1843	20,1843	20,1843	20,1843	20,1843	20,1843	20,1843	20,1843	20,1843
similaritymatrix_075_10_3	5	5	16,5491	16,5491	16,5491	16,5491	16,5491	16,5491	16,5491	16,5491	16,5491	16,5491
similaritymatrix_075_10_3	2	8	21,914	21,914	21,914	21,914	21,914	21,914	21,914	21,914	21,914	21,914
similaritymatrix_075_10_4	5	5	16,2832	16,2832	16,2832	16,2832	16,2832	16,1163	16,2832	16,2832	16,2832	16,2832
similaritymatrix_075_10_4	2	8	19,9435	19,7932	19,9435	19,9435	19,9435	19,8531	19,8531	19,9435	19,9435	19,9435
similaritymatrix_075_10_5	5	5	16,5168	16,5168	16,5168	16,5168	16,5168	16,5168	16,5168	16,5168	16,5168	16,5168
similaritymatrix_075_10_5	2	8	20,4485	20,4485	20,4485	20,4485	20,4485	20,4485	19,8617	20,4485	20,4485	20,4485
similaritymatrix_025_20_1	10	10	30,3149	26,1712	29,3693	29,3693	29,3693	29,304	29,304	29,3693	29,3693	30,3149
similaritymatrix_025_20_1	4	16	34,5235	34,4661	34,5235	34,5235	34,5235	34,5235	34,5235	34,5235	34,5235	34,5235
similaritymatrix_025_20_2	10	10	32,5831	26,4975	29,7237	29,7237	30,7034	30,2097	31,2765	31,2765	30,7034	32,0712
similaritymatrix_025_20_2	4	16	37,9468	33,6142	35,3435	37,9468	37,9468	37,59	37,59	37,9468	37,9468	37,9468
similaritymatrix_025_20_3	10	10	30,4026	24,9637	26,9538	27,666	30,4026	29,8375	29,582	29,0053	30,4026	30,4026
similaritymatrix_025_20_3	4	16	33,9979	23,9272	32,6317	32,6317	32,8456	33,311	33,311	32,6317	33,6133	33,9979
similaritymatrix_025_20_4	10	10	30,0218	21,6533	25,4121	28,4167	30,0218	27,8548	27,8548	29,5418	30,0218	30,0218
similaritymatrix_025_20_4	4	16	33,6102	22,9393	32,6231	33,6102	33,6102	33,1606	33,1606	33,6102	33,6102	33,6102
similaritymatrix_025_20_5	10	10	31,0816	12,6541	26,5453	28,104	31,0816	31,0816	30,1478	30,1478	31,0816	31,0816
similaritymatrix_025_20_5	4	16	32,939	13,5581	32,0416	32,939	32,939	32,8159	32,8159	32,8159	32,939	32,939
similaritymatrix_050_20_1	10	10	46,3435	39,2056	40,786	40,786	43,1889	45,4879	45,6442	44,8917	45,5861	45,4879
similaritymatrix_050_20_1	4	16	54,5386	28,6728	53,0482	53,0633	54,0429	53,1405	50,2674	54,2216	54,5386	54,5386
similaritymatrix_050_20_2	10	10	48,284	36,3434	42,0535	42,0535	48,284	45,2421	45,2421	45,8687	48,284	48,284
similaritymatrix_050_20_2	4	16	56,839	40,0073	49,4231	52,2576	56,1295	56,22	56,839	56,5904	56,839	56,839

similaritymatrix_050_20_3	10	10	45,0089	30,2366	42,6208	43,8014	43,9827	43,4646	44,6667	43,8014	45,0089	44,745
similaritymatrix_050_20_3	4	16	54,9534	39,9501	54,16	54,3528	54,6277	54,7671	54,7671	54,7671	54,7671	54,9534
similaritymatrix_050_20_4	10	10	43,8548	25,6368	38,9538	38,9538	41,7443	42,1592	42,1592	41,5482	43,4894	43,8548
similaritymatrix_050_20_4	4	16	52,5016	45,7184	49,1844	49,1844	51,8445	51,9359	51,9359	51,9873	52,5016	52,5016
similaritymatrix_050_20_5	10	10	46,2017	33,154	41,9852	43,0378	45,2717	45,1185	45,1185	46,2017	46,007	45,4728
similaritymatrix_050_20_5	4	16	54,864	48,1948	50,6522	52,433	53,2427	54,7182	54,7182	52,433	54,864	54,864
similaritymatrix_075_20_1	10	10	54,4506	37,5038	50,5339	50,5339	52,7475	53,0877	53,0877	53,0877	54,4132	54,4506
similaritymatrix_075_20_1	4	16	66,1831	60,63	60,63	61,3877	64,9863	64,587	64,587	64,587	66,1831	65,1535
similaritymatrix_075_20_2	10	10	57,8937	46,2159	55,1917	55,1917	55,6069	52,8765	57,6894	57,6894	57,2565	57,3284
similaritymatrix_075_20_2	4	16	72,0678	60,6557	68,4048	69,0576	70,6035	70,2542	70,2542	70,2542	72,0678	72,0418
similaritymatrix_075_20_3	10	10	57,8223	46,3715	53,8988	53,8988	55,8428	56,9692	56,9692	55,9316	57,8223	57,8223
similaritymatrix_075_20_3	4	16	71,1108	51,1978	66,8104	66,8104	71,0732	67,1821	69,5221	68,9112	71,1108	71,1108
similaritymatrix_075_20_4	10	10	56,9956	42,7787	50,0169	50,0169	55,2861	56,0428	56,0428	55,5272	56,8374	56,5932
similaritymatrix_075_20_4	4	16	71,1807	59,6011	67,996	67,996	71,1807	71,1807	71,1807	71,1807	71,1807	71,1807
similaritymatrix_075_20_5	10	10	54,3156	39,625	50,9177	50,9177	52,2318	52,0071	52,0071	53,0128	54,0492	54,0661
similaritymatrix_075_20_5	4	16	67,4808	62,222	62,9412	62,9412	65,6206	67,4808	66,3642	67,0119	66,3285	66,9765
similaritymatrix_025_30_1	15	15	48,3419	25,3199	46,5825	46,5825	46,7854	46,8868	46,6393	47,467	48,3419	48,3419
similaritymatrix_025_30_1	24	6	56,572	41,9344	56,1846	56,572	56,572	55,9133	55,9133	56,572	56,572	56,572
similaritymatrix_025_30_2	15	15	55,9228	32,039	49,1649	50,8098	50,8098	51,9999	51,9999	54,2478	54,2478	55,5662
similaritymatrix_025_30_2	24	6	65,4896	46,953	61,3216	63,3461	65,4896	64,1749	62,5499	64,861	65,4896	65,4896
similaritymatrix_025_30_3	15	15	52,7701	30,983	42,8059	42,8059	47,6652	51,85	51,85	47,7574	49,8581	51,4026
similaritymatrix_025_30_3	24	6	58,7257	41,51	57,4703	57,7441	58,7037	57,1111	57,1111	57,7441	58,7257	58,7257
similaritymatrix_025_30_4	15	15	50,5031	32,6199	46,1015	46,4493	47,0349	48,2079	48,2079	48,0482	50,5031	48,4049
similaritymatrix_025_30_4	24	6	57,5037	38,3989	56,6294	56,6505	56,9721	56,8788	56,8788	56,8788	57,1746	57,0378
similaritymatrix_025_30_5	15	15	46,8051	24,8572	46,4539	46,4539	46,4539	44,8645	44,8645	46,4539	46,4539	46,5957
similaritymatrix_025_30_5	24	6	54,3592	45,9317	52,3791	54,199	54,3592	53,2383	54,3592	54,3592	54,3592	54,3592
similaritymatrix_050_30_1	15	15	85,0068	66,7404	75,0519	75,0519	75,0519	84,5781	84,5781	82,2477	82,2477	83,9247
similaritymatrix_050_30_1	24	6	104,795	91,9379	99,8233	102,12	103,352	102,52	102,52	104,795	103,352	103,352
similaritymatrix_050_30_2	15	15	86,0076	71,9645	72,8638	74,1295	75,8131	84,2887	84,2887	85,6245	85,6245	85,1101
similaritymatrix_050_30_2	24	6	101,291	94,6532	97,1498	100,699	100,699	100,713	100,699	100,699	100,699	100,777
similaritymatrix_050_30_3	15	15	87,6286	68,5955	69,5229	76,5734	76,5734	87,5057	87,5057	86,8973	86,8973	85,1545
similaritymatrix_050_30_3	24	6	106,038	93,419	104,855	104,855	104,855	105,705	105,705	106,038	106,038	106,038
similaritymatrix_050_30_4	15	15	86,817	67,3357	72,1235	72,609	74,1294	83,2809	83,2809	84,8565	84,3037	86,1702
similaritymatrix_050_30_4	24	6	105,351	99,7497	100,319	104,054	104,054	103,884	103,884	104,454	104,454	104,454
similaritymatrix_050_30_5	15	15	80,8599	61,4516	66,6429	66,6429	67,3854	79,6025	79,6025	79,0951	76,226	80,8599
similaritymatrix_050_30_5	24	6	98,6817	88,1356	91,3437	96,2332	96,5265	95,7567	95,7567	97,0363	97,0363	97,3539
similaritymatrix_075_30_1	15	15	109,687	92,9409	93,4282	97,2334	97,2334	106,551	106,551	106,112	106,112	107,85
similaritymatrix_075_30_1	24	6	135,574	124,618	128,615	130,739	134,722	135,267	133,411	134,965	134,958	134,901
similaritymatrix_075_30_2	15	15	114,023	92,545	102,58	102,58	102,58	113,223	113,223	108,104	108,104	114,023
similaritymatrix_075_30_2	24	6	142,987	138,815	139,274	142,255	142,255	142,607	142,227	142,255	142,255	142,411
similaritymatrix_075_30_3	15	15	111,006	93,1714	95,8913	99,1773	100,226	109,081	109,081	107,958	110,119	108,278
similaritymatrix_075_30_3	24	6	138,125	125,502	131,37	136,785	138,088	136,674	136,674	138,088	138,088	138,125
similaritymatrix_075_30_4	15	15	107,143	91,6662	91,6662	91,6662	97,0093	104,935	104,935	104,935	105,999	105,961
similaritymatrix_075_30_4	24	6	135,314	130,162	130,928	132,087	132,618	135,004	135,004	135,004	134,163	134,624
similaritymatrix_075_30_5	15	15	108,321	90,6047	90,6047	90,6047	96,3189	106,665	106,665	106,665	106,462	107,663
similaritymatrix_075_30_5	24	6	139,998	132,466	137,137	137,973	139,561	139,315	139,315	139,998	139,561	139,998

C.3. Relative error for K=2

			2-DIVISION				4-DIVISION
NAME	M_1	M_2	RELATIVE ERROR, S_0 = 3 s	RELATIVE ERROR, S_0 = 5 s	RELATIVE ERROR, S_0 = 10 s	RELATIVE ERROR, S_0 = 20 s	RELATIVE ERROR, S_0 = 20 s
similaritymatrix_025_10_1	5	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_1	2	8	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_2	5	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_2	2	8	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_3	5	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_3	2	8	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_4	5	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_4	2	8	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_5	5	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_5	2	8	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_1	5	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_1	2	8	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_2	5	5	0,33%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_2	2	8	1,13%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_3	5	5	1,48%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_3	2	8	1,89%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_4	5	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_4	2	8	1,24%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_5	5	5	0,68%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_5	2	8	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_1	5	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_1	2	8	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_2	5	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_2	2	8	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_3	5	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_3	2	8	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_4	5	5	1,02%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_4	2	8	0,45%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_5	5	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_5	2	8	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_20_1	10	10	3,33%	2,43%	3,12%	3,22%	0,00%
similaritymatrix_025_20_1	4	16	0,00%	6,48%	0,00%	0,00%	0,00%
similaritymatrix_025_20_2	10	10	7,28%	1,37%	4,01%	6,12%	1,57%
similaritymatrix_025_20_2	4	16	0,94%	6,78%	0,00%	0,00%	0,00%
similaritymatrix_025_20_3	10	10	1,86%	2,12%	4,60%	0,00%	0,00%
similaritymatrix_025_20_3	4	16	2,02%	0,52%	4,02%	1,14%	0,00%
similaritymatrix_025_20_4	10	10	7,22%	4,20%	1,60%	0,00%	0,00%
similaritymatrix_025_20_4	4	16	1,34%	8,45%	0,00%	0,00%	0,00%
similaritymatrix_025_20_5	10	10	0,00%	2,68%	3,00%	0,00%	0,00%
similaritymatrix_025_20_5	4	16	0,37%	0,74%	0,37%	0,00%	0,00%
similaritymatrix_050_20_1	10	10	1,85%	5,81%	3,13%	1,66%	1,85%
similaritymatrix_050_20_1	4	16	2,56%	4,39%	0,58%	0,00%	0,00%
similaritymatrix_050_20_2	10	10	6,30%	2,99%	5,00%	0,00%	0,00%
similaritymatrix_050_20_2	4	16	1,09%	3,39%	0,44%	0,00%	0,00%

similaritymatrix_050_20_3	10	10	3,43%	0,00%	2,68%	0,00%	0,59%
similaritymatrix_050_20_3	4	16	0,34%	2,62%	0,34%	0,34%	0,00%
similaritymatrix_050_20_4	10	10	3,87%	4,20%	5,26%	0,84%	0,00%
similaritymatrix_050_20_4	4	16	1,08%	6,18%	0,98%	0,00%	0,00%
similaritymatrix_050_20_5	10	10	2,34%	2,15%	0,00%	0,42%	1,58%
similaritymatrix_050_20_5	4	16	0,27%	4,28%	4,43%	0,00%	0,00%
similaritymatrix_075_20_1	10	10	2,50%	1,81%	2,50%	0,07%	0,00%
similaritymatrix_075_20_1	4	16	2,41%	4,58%	2,41%	0,00%	1,56%
similaritymatrix_075_20_2	10	10	8,67%	2,29%	0,35%	1,11%	0,98%
similaritymatrix_075_20_2	4	16	2,52%	1,41%	2,52%	0,00%	0,04%
similaritymatrix_075_20_3	10	10	1,48%	0,00%	3,27%	0,00%	0,00%
similaritymatrix_075_20_3	4	16	5,52%	0,00%	3,09%	0,00%	0,00%
similaritymatrix_075_20_4	10	10	1,67%	1,80%	2,58%	0,28%	0,71%
similaritymatrix_075_20_4	4	16	0,00%	2,48%	0,00%	0,00%	0,00%
similaritymatrix_075_20_5	10	10	4,25%	7,61%	2,40%	0,49%	0,46%
similaritymatrix_075_20_5	4	16	0,00%	2,76%	0,69%	1,74%	0,75%
similaritymatrix_025_30_1	15	15	3,01%	3,33%	1,81%	0,00%	0,00%
similaritymatrix_025_30_1	24	6	1,16%	10,01%	0,00%	0,00%	0,00%
similaritymatrix_025_30_2	15	15	7,01%	5,80%	3,00%	3,09%	0,64%
similaritymatrix_025_30_2	24	6	2,01%	3,07%	0,96%	0,00%	0,00%
similaritymatrix_025_30_3	15	15	1,74%	1,44%	9,50%	5,84%	2,59%
similaritymatrix_025_30_3	24	6	2,75%	7,60%	1,67%	0,00%	0,00%
similaritymatrix_025_30_4	15	15	4,54%	6,33%	4,86%	0,00%	4,15%
similaritymatrix_025_30_4	24	6	1,09%	3,85%	1,09%	0,58%	0,81%
similaritymatrix_025_30_5	15	15	4,15%	6,08%	0,75%	0,76%	0,45%
similaritymatrix_025_30_5	24	6	2,06%	6,34%	0,00%	0,00%	0,00%
similaritymatrix_050_30_1	15	15	0,50%	0,16%	3,25%	3,35%	1,27%
similaritymatrix_050_30_1	24	6	2,17%	4,98%	0,00%	1,40%	1,38%
similaritymatrix_050_30_2	15	15	2,00%	0,00%	0,45%	0,45%	1,04%
similaritymatrix_050_30_2	24	6	0,57%	3,85%	0,58%	0,59%	0,51%
similaritymatrix_050_30_3	15	15	0,14%	0,75%	0,83%	0,84%	2,82%
similaritymatrix_050_30_3	24	6	0,31%	9,15%	0,00%	0,00%	0,00%
similaritymatrix_050_30_4	15	15	4,07%	9,38%	2,26%	2,98%	0,75%
similaritymatrix_050_30_4	24	6	1,39%	2,66%	0,85%	0,86%	0,85%
similaritymatrix_050_30_5	15	15	1,56%	4,25%	2,18%	6,08%	0,00%
similaritymatrix_050_30_5	24	6	2,96%	3,93%	1,67%	1,70%	1,35%
similaritymatrix_075_30_1	15	15	2,86%	2,11%	3,26%	3,37%	1,67%
similaritymatrix_075_30_1	24	6	0,23%	2,26%	0,45%	0,46%	0,50%
similaritymatrix_075_30_2	15	15	0,70%	2,16%	5,19%	5,48%	0,00%
similaritymatrix_075_30_2	24	6	0,27%	7,23%	0,51%	0,51%	0,40%
similaritymatrix_075_30_3	15	15	1,73%	4,07%	2,75%	0,81%	2,46%
similaritymatrix_075_30_3	24	6	1,05%	3,40%	0,03%	0,03%	0,00%
similaritymatrix_075_30_4	15	15	2,06%	4,19%	2,06%	1,08%	1,10%
similaritymatrix_075_30_4	24	6	0,23%	4,27%	0,23%	0,86%	0,51%
similaritymatrix_075_30_5	15	15	1,53%	1,65%	1,53%	1,75%	0,61%
similaritymatrix_075_30_5	24	6	0,49%	0,66%	0,00%	0,31%	0,00%

C.4. CPU time for K=3

					2-DIVISION				4 DIVISION
NAME	M_1	M_2	M_3	CPU time F6 (s)	CPU time (s), S_0 = 3 s	CPU time (s), S_0 = 5 s	CPU time (s), S_0 = 10 s	CPU time (s), S_0 = 20 s	CPU time (s), S_0 = 20 s
similaritymatrix_025_10_1	3	3	4	2,83	2,311	2,35	2,324	2,287	3,282
similaritymatrix_025_10_1	2	3	5	1,103	1,09	4,715	1,338	1,269	1,754
similaritymatrix_025_10_2	3	3	4	1,799	1,824	1,973	2,025	2,197	2,967
similaritymatrix_025_10_2	2	3	5	1,14	1,32	1,441	1,49	1,825	1,842
similaritymatrix_025_10_3	3	3	4	1,521	1,513	1,725	2,175	2,068	1,997
similaritymatrix_025_10_3	2	3	5	0,739	0,809	0,919	1,387	0,93	1,377
similaritymatrix_025_10_4	3	3	4	0,968	0,951	1,082	1,262	1,178	1,725
similaritymatrix_025_10_4	2	3	5	0,833	0,811	0,926	0,992	0,918	1,411
similaritymatrix_025_10_5	3	3	4	1,817	1,733	1,814	1,901	1,899	2,425
similaritymatrix_025_10_5	2	3	5	1,291	1,233	1,347	1,394	1,557	1,811
similaritymatrix_050_10_1	3	3	4	1,319	1,246	1,451	1,438	1,667	1,875
similaritymatrix_050_10_1	2	3	5	1,116	1,166	1,288	1,378	1,337	1,995
similaritymatrix_050_10_2	3	3	4	1,459	1,469	1,548	1,696	1,831	1,984
similaritymatrix_050_10_2	2	3	5	0,756	0,862	0,947	0,966	0,985	1,657
similaritymatrix_050_10_3	3	3	4	1,449	1,377	1,557	1,892	1,554	2,453
similaritymatrix_050_10_3	2	3	5	1,086	1,125	1,22	1,442	1,298	2,048
similaritymatrix_050_10_4	3	3	4	1,737	1,466	1,586	2,04	2,352	2,072
similaritymatrix_050_10_4	2	3	5	1,366	1,101	1,21	1,329	1,222	1,633
similaritymatrix_050_10_5	3	3	4	0,797	0,823	0,917	1,171	0,967	1,271
similaritymatrix_050_10_5	2	3	5	1,023	0,996	1,064	1,692	1,408	1,527
similaritymatrix_075_10_1	3	3	4	0,775	0,819	0,976	1,44	0,936	1,242
similaritymatrix_075_10_1	2	3	5	0,808	0,856	0,979	1,055	1,24	1,256
similaritymatrix_075_10_2	3	3	4	0,853	0,807	0,898	0,977	0,992	1,831
similaritymatrix_075_10_2	2	3	5	0,741	0,811	0,925	0,997	0,881	1,628
similaritymatrix_075_10_3	3	3	4	1,776	1,775	1,936	2,406	2,017	3,349
similaritymatrix_075_10_3	2	3	5	0,779	0,887	1,011	1,041	1,071	1,39
similaritymatrix_075_10_4	3	3	4	0,801	0,835	0,939	1,288	0,963	1,427
similaritymatrix_075_10_4	2	3	5	0,935	0,958	1,078	1,15	1,12	1,643
similaritymatrix_075_10_5	3	3	4	1,198	0,822	1	0,962	1,06	1,394
similaritymatrix_075_10_5	2	3	5	0,85	0,802	0,921	0,984	1,156	1,113
similaritymatrix_025_20_1	7	7	6	156,416	6,381	5,654	10,659	20,671	37,582
similaritymatrix_025_20_1	3	7	10	72,576	4,791	5,803	11,216	20,727	25,469
similaritymatrix_025_20_2	7	7	6	365,097	4,996	7,125	12,265	21,082	37,785
similaritymatrix_025_20_2	3	7	10	88,565	4,215	6,181	11,894	21,801	30,632
similaritymatrix_025_20_3	7	7	6	230,279	5,154	5,68	11,096	20,542	52,937
similaritymatrix_025_20_3	3	7	10	115,35	5,713	5,587	10,618	20,978	36,162
similaritymatrix_025_20_4	7	7	6	80,458	3,794	5,875	11,136	20,645	24,408
similaritymatrix_025_20_4	3	7	10	44,883	4,008	6,13	10,944	20,603	28,852
similaritymatrix_025_20_5	7	7	6	33,908	3,53	5,725	10,684	20,661	23,28
similaritymatrix_025_20_5	3	7	10	28,808	3,659	5,747	11,133	20,531	23,416
similaritymatrix_050_20_1	7	7	6	352,119	4,097	11,849	17,257	21,103	64,68
similaritymatrix_050_20_1	3	7	10	40,609	4,274	5,865	11,071	21,297	23,643
similaritymatrix_050_20_2	7	7	6	1113	4,686	8,639	24,715	20,842	381,258
similaritymatrix_050_20_2	3	7	10	91,876	5,053	10,737	11,233	20,84	27,371

similaritymatrix_050_20_3	7	7	6	1258,52	5,075	5,923	10,744	21,294	25,393
similaritymatrix_050_20_3	3	7	10	271,076	3,758	5,754	11,16	21,689	47,135
similaritymatrix_050_20_4	7	7	6	363,206	3,597	6,175	11,026	21,359	32,713
similaritymatrix_050_20_4	3	7	10	91,648	3,798	5,841	10,855	21,159	25,503
similaritymatrix_050_20_5	7	7	6	243,669	3,686	5,631	10,74	20,847	29,453
similaritymatrix_050_20_5	3	7	10	72,659	4,317	5,625	11,105	20,699	24,765
similaritymatrix_075_20_1	7	7	6	522,168	8,395	5,891	10,837	21,011	322,564
similaritymatrix_075_20_1	3	7	10	116,726	3,873	6,195	11,357	21,03	31,206
similaritymatrix_075_20_2	7	7	6	102,192	3,52	7,102	10,693	20,651	32,695
similaritymatrix_075_20_2	3	7	10	210,091	3,923	5,664	10,96	21,186	26,902
similaritymatrix_075_20_3	7	7	6	369,553	4,98	6,267	12,4	23,354	41,384
similaritymatrix_075_20_3	3	7	10	62,767	4,258	10,127	15,519	21,044	61,953
similaritymatrix_075_20_4	7	7	6	300,423	5,247	5,655	10,609	20,64	25,241
similaritymatrix_075_20_4	3	7	10	41,939	3,952	5,622	11,387	20,606	23,093
similaritymatrix_075_20_5	7	7	6	222,101	9,002	5,618	10,731	20,911	22,46
similaritymatrix_075_20_5	3	7	10	72,252	4,088	6,18	11,174	21,069	22,918
similaritymatrix_025_30_1	10	10	10	274,833	3,903	6,33	11,325	23,133	60,203
similaritymatrix_025_30_1	5	10	15	202,323	4,648	6,816	11,564	22,633	52,323
similaritymatrix_025_30_2	10	10	10	768,868	3,955	6,506	11,489	22,507	76,016
similaritymatrix_025_30_2	5	10	15	510,705	4,509	6,669	11,937	23,07	64,405
similaritymatrix_025_30_3	10	10	10	791,253	5,411	7,543	11,738	22,539	63,699
similaritymatrix_025_30_3	5	10	15	414,113	4,261	6,389	13,632	24,662	47,181
similaritymatrix_025_30_4	10	10	10	235,896	4,442	7,005	13,505	25,999	68,61
similaritymatrix_025_30_4	5	10	15	300,362	4,215	6,859	12,246	23,659	58,502
similaritymatrix_025_30_5	10	10	10	281,449	6,321	6,314	11,516	22,167	36,931
similaritymatrix_025_30_5	5	10	15	362,486	3,893	6,034	10,973	21,454	49,247
similaritymatrix_050_30_1	10	10	10	4613,54	6,364	8,412	13,75	26,34	90,469
similaritymatrix_050_30_1	5	10	15	907,799	5,521	7,598	12,582	24,003	71,515
similaritymatrix_050_30_2	10	10	10	1924,95	4,322	6,354	18,062	33,303	125,84
similaritymatrix_050_30_2	5	10	15	1662,58	12,369	6,393	14,418	27,919	54,477
similaritymatrix_050_30_3	10	10	10	2195,66	10,617	9,893	18,244	33,675	58,996
similaritymatrix_050_30_3	5	10	15	904,955	4,906	7,478	12,902	24,49	88,981
similaritymatrix_050_30_4	10	10	10	2047	9,858	6,871	12,32	25,036	519,504
similaritymatrix_050_30_4	5	10	15	1411,28	4,672	6,394	11,472	24,472	73,554
similaritymatrix_050_30_5	10	10	10	2944,07	6,909	9,204	12,118	30,939	91,006
similaritymatrix_050_30_5	5	10	15	1749,69	5,706	7,863	13,48	22,124	91,825
similaritymatrix_075_30_1	10	10	10	5798,44	5,94	7,982	12,193	23,336	341,388
similaritymatrix_075_30_1	5	10	15	1388,85	5,293	6,211	14,695	23,836	154,357
similaritymatrix_075_30_2	10	10	10	1999,55	12,271	14,339	21,979	23,352	64,699
similaritymatrix_075_30_2	5	10	15	760,99	6,56	8,546	11,305	24,687	66,723
similaritymatrix_075_30_3	10	10	10	7200,54	6,427	8,547	12,897	23,113	132,611
similaritymatrix_075_30_3	5	10	15	2318,38	7,324	9,272	15,145	21,506	77,189
similaritymatrix_075_30_4	10	10	10	3801,08	4,762	6,839	11,684	21,284	95,241
similaritymatrix_075_30_4	5	10	15	1276,02	7,78	7,142	13,058	24,915	47,253
similaritymatrix_075_30_5	10	10	10	2256,86	9,152	11,27	17,471	28,882	746,755
similaritymatrix_075_30_5	5	10	15	859,169	4,263	6,531	12,197	21,616	93,722

C.5. OF for K=3

					OF S_0				2-DIVISION				4-DIVISION
NAME	M_1	M_2	M_3	OF F6	OF 3 s	OF 5 s	OF 10 s	OF 20 s	OF, S_0 = 3 s	OF, S_0 = 5 s	OF, S_0 = 10 s	OF, S_0 = 20 s	OF, S_0 = 20 s
similaritymatrix_025_10_1	3	3	4	8,89929	3,27942	8,89929	8,8993	8,89929	8,89929	8,89929	8,89929	8,89929	8,89929
similaritymatrix_025_10_1	2	3	5	9,46372	9,46372	9,46372	9,4637	9,46372	9,46372	9,46372	9,46372	9,46372	9,46372
similaritymatrix_025_10_2	3	3	4	9,39057	9,39057	9,39057	9,3906	9,39057	9,39057	9,39057	9,39057	9,39057	9,39057
similaritymatrix_025_10_2	2	3	5	10,0889	8,89952	10,0889	10,089	10,0889	10,0889	10,0889	10,0889	10,0889	10,0889
similaritymatrix_025_10_3	3	3	4	9,78516	9,78516	9,78516	9,7852	9,78516	9,78516	9,78516	9,78516	9,78516	9,78516
similaritymatrix_025_10_3	2	3	5	9,90476	9,90476	9,90476	9,9048	9,90476	9,90476	9,90476	9,90476	9,90476	9,90476
similaritymatrix_025_10_4	3	3	4	9,84519	9,84519	9,84519	9,8452	9,84519	9,84519	9,84519	9,84519	9,84519	9,84519
similaritymatrix_025_10_4	2	3	5	10,0993	10,0993	10,0993	10,099	10,0993	10,0993	10,0993	10,0993	10,0993	10,0993
similaritymatrix_025_10_5	3	3	4	8,9353	8,9353	8,9353	8,9353	8,9353	8,9353	8,9353	8,9353	8,9353	8,9353
similaritymatrix_025_10_5	2	3	5	9,51342	9,51342	9,51342	9,5134	9,51342	9,51342	9,51342	9,51342	9,51342	9,51342
similaritymatrix_050_10_1	3	3	4	10,6405	10,6405	10,6405	10,641	10,6405	10,6405	10,6405	10,6405	10,6405	10,6405
similaritymatrix_050_10_1	2	3	5	11,7341	11,7341	11,7341	11,734	11,7341	11,7341	11,7341	11,7341	11,7341	11,7341
similaritymatrix_050_10_2	3	3	4	10,6084	10,6084	10,6084	10,608	10,6084	10,6084	10,6084	10,6084	10,6084	10,6084
similaritymatrix_050_10_2	2	3	5	11,1914	11,1914	11,1914	11,191	11,1914	11,1914	11,1914	11,1914	11,1914	11,1914
similaritymatrix_050_10_3	3	3	4	10,5272	10,5272	10,5272	10,527	10,5272	10,5272	10,5272	10,5272	10,5272	10,5272
similaritymatrix_050_10_3	2	3	5	11,3806	11,3806	11,3806	11,381	11,3806	11,3806	11,3806	11,3806	11,3806	11,3806
similaritymatrix_050_10_4	3	3	4	10,4859	10,4859	10,4859	10,486	10,4859	10,4859	10,4859	10,4859	10,4859	10,4859
similaritymatrix_050_10_4	2	3	5	11,184	11,184	11,184	11,184	11,184	11,184	11,184	11,184	11,184	11,184
similaritymatrix_050_10_5	3	3	4	11,0992	11,0992	11,0992	11,099	11,0992	11,0992	11,0992	11,0992	11,0992	11,0992
similaritymatrix_050_10_5	2	3	5	11,5222	11,5222	11,5222	11,522	11,5222	11,5222	11,5222	11,5222	11,5222	11,5222
similaritymatrix_075_10_1	3	3	4	11,1182	11,1182	11,1182	11,118	11,1182	11,1182	11,1182	11,1182	11,1182	11,1182
similaritymatrix_075_10_1	2	3	5	12,1253	12,1253	12,1253	12,125	12,1253	12,1253	12,1253	12,1253	12,1253	12,1253
similaritymatrix_075_10_2	3	3	4	11,2169	11,2169	11,2169	11,217	11,2169	11,2169	11,2169	11,2169	11,2169	11,2169
similaritymatrix_075_10_2	2	3	5	12,2122	12,2122	12,2122	12,212	12,2122	12,2122	12,2122	12,2122	12,2122	12,2122
similaritymatrix_075_10_3	3	3	4	11,0686	11,0686	11,0686	11,069	11,0686	11,0686	11,0686	11,0686	11,0686	11,0686
similaritymatrix_075_10_3	2	3	5	12,4794	12,4794	12,4794	12,479	12,4794	12,4794	12,4794	12,4794	12,4794	12,4794
similaritymatrix_075_10_4	3	3	4	11,1699	11,1699	11,1699	11,17	11,1699	11,1699	11,1699	11,1699	11,1699	11,1699
similaritymatrix_075_10_4	2	3	5	12,3132	12,3132	12,3132	12,313	12,3132	12,3132	12,3132	12,3132	12,3132	12,3132
similaritymatrix_075_10_5	3	3	4	11,1499	11,1499	11,1499	11,15	11,1499	11,1499	11,1499	11,1499	11,1499	11,1499
similaritymatrix_075_10_5	2	3	5	12,1749	12,1749	12,1749	12,175	12,1749	12,1749	12,1749	12,1749	12,1749	12,1749
similaritymatrix_025_20_1	7	7	6	24,165	18,7047	22,2801	22,28	24,0274	23,6813	24,165	24,165	24,2693	24,7515
similaritymatrix_025_20_1	3	7	10	25,8403	22,9934	25,2192	25,219	25,2638	25,8575	25,8403	25,8403	25,8403	25,9972
similaritymatrix_025_20_2	7	7	6	26,2465	23,6173	23,6173	23,617	24,7701	26,2465	26,2465	26,2465	26,5776	26,2733
similaritymatrix_025_20_2	3	7	10	27,0299	22,0396	23,8701	23,87	27,7568	26,1916	27,0299	27,0299	28,1965	27,9515
similaritymatrix_025_20_3	7	7	6	25,6025	20,1597	23,8916	23,892	25,7586	25,5389	25,6025	25,6025	25,7586	25,7586
similaritymatrix_025_20_3	3	7	10	26,9766	22,4633	24,3075	24,308	26,4482	26,2569	26,9766	26,9766	26,9402	27,0184
similaritymatrix_025_20_4	7	7	6	24,0535	22,5303	22,5303	22,817	23,8229	24,0535	24,0535	24,8028	24,3432	24,7599
similaritymatrix_025_20_4	3	7	10	24,686	22,6376	22,6376	23,733	26,0323	24,686	24,686	25,1051	26,0323	26,0323
similaritymatrix_025_20_5	7	7	6	25,6069	23,1883	23,1883	24,104	26,2936	25,6069	25,6069	25,4001	26,2936	26,2936
similaritymatrix_025_20_5	3	7	10	27,7251	26,5574	26,5574	26,557	27,9312	27,7251	27,7251	27,7251	27,9312	27,9312
similaritymatrix_050_20_1	7	7	6	34,3198	28,8544	33,5254	33,525	34,3126	33,7914	34,3198	34,3198	34,5567	36,3151
similaritymatrix_050_20_1	3	7	10	38,5069	36,3406	37,532	37,532	40,0155	39,4478	38,5069	38,5069	40,1966	40,1966
similaritymatrix_050_20_2	7	7	6	33,9653	26,9267	29,2868	31,399	33,0592	34,3986	33,9653	34,5296	33,6748	34,677
similaritymatrix_050_20_2	3	7	10	38,5804	32,0015	35,0221	37,084	39,887	38,8628	38,5804	39,6276	39,887	39,887

similaritymatrix_050_20_3	7	7	6	34,3472	29,2376	32,6149	32,615	32,755	34,301	34,3472	34,3472	34,1486	34,3472
similaritymatrix_050_20_3	3	7	10	36,9927	35,0813	35,0813	35,415	37,2811	36,9927	36,9927	37,3747	37,4346	37,8653
similaritymatrix_050_20_4	7	7	6	32,907	30,294	31,1368	31,496	31,6297	33,2444	32,907	32,346	32,8918	33,9485
similaritymatrix_050_20_4	3	7	10	35,5008	34,26	34,26	34,444	35,7499	35,5008	35,5008	35,7879	36,3012	36,7872
similaritymatrix_050_20_5	7	7	6	35,2546	32,3739	33,4758	33,476	35,1948	35,211	35,2546	35,2546	35,7609	35,7609
similaritymatrix_050_20_5	3	7	10	37,3473	35,0569	35,6352	35,635	35,6352	37,7814	37,3473	37,3473	37,3473	38,9444
similaritymatrix_075_20_1	7	7	6	39,3996	35,0888	39,6813	39,681	39,6813	39,9266	39,3996	40,1144	40,1144	40,1144
similaritymatrix_075_20_1	3	7	10	43,2748	40,1445	40,7724	43,441	44,6109	45,2558	43,2748	43,441	45,2523	45,2558
similaritymatrix_075_20_2	7	7	6	42,3854	38,9368	39,8421	42,376	43,355	41,7205	42,3854	43,355	43,355	43,355
similaritymatrix_075_20_2	3	7	10	47,3123	44,8413	46,5091	46,509	46,7871	46,8899	47,3123	47,3123	47,3123	47,5465
similaritymatrix_075_20_3	7	7	6	42,246	33,5271	39,1623	40,158	41,1023	41,3737	42,246	41,1841	41,1841	41,8166
similaritymatrix_075_20_3	3	7	10	47,8601	42,5872	46,3543	46,354	46,7506	46,0929	47,8601	47,8601	47,8601	47,8601
similaritymatrix_075_20_4	7	7	6	41,4215	33,6252	40,9623	40,962	40,9623	40,9345	41,4215	41,3486	41,6735	41,7693
similaritymatrix_075_20_4	3	7	10	46,6728	39,9654	40,3648	41,89	47,0021	44,161	46,6728	46,8295	47,832	47,832
similaritymatrix_075_20_5	7	7	6	38,0217	33,4297	37,7168	37,751	39,4626	38,3229	38,0217	39,3739	39,4626	40,4879
similaritymatrix_075_20_5	3	7	10	44,1256	42,6296	42,6296	42,63	43,9457	44,1256	44,1256	44,1256	43,9457	44,0706
similaritymatrix_025_30_1	10	10	10	38,3095	25,3793	30,8275	32,781	36,2109	37,3075	38,3095	38,3095	37,8325	38,7742
similaritymatrix_025_30_1	5	10	15	39,5814	30,2628	33,1847	39,074	39,0736	40,3365	39,5814	40,7372	40,7372	41,8854
similaritymatrix_025_30_2	10	10	10	43,614	32,1816	37,2122	37,588	37,8982	44,0645	43,614	42,3198	42,3198	45,0588
similaritymatrix_025_30_2	5	10	15	48,7557	27,5903	45,2802	45,28	45,2802	44,9767	48,7557	48,7557	48,7557	49,7224
similaritymatrix_025_30_3	10	10	10	42,2283	27,4937	29,6794	35,045	35,0452	41,5593	42,2283	40,8043	40,8043	41,3502
similaritymatrix_025_30_3	5	10	15	42,951	24,4952	39,6788	39,679	39,6788	42,951	42,951	45,0498	45,0498	45,8491
similaritymatrix_025_30_4	10	10	10	40,2944	20,7525	33,6976	36,939	37,3856	39,8489	40,2944	40,9308	40,9308	42,2745
similaritymatrix_025_30_4	5	10	15	43,5165	21,1901	38,4985	38,499	38,4985	43,5366	43,5165	43,5165	43,5165	43,1297
similaritymatrix_025_30_5	10	10	10	36,6754	20,5348	30,6095	35,646	35,6456	35,75	36,6754	37,0767	37,0767	37,7624
similaritymatrix_025_30_5	5	10	15	39,3115	29,136	29,136	36,575	39,2753	39,3115	39,3115	40,6024	39,7345	40,9105
similaritymatrix_050_30_1	10	10	10	64,5623	52,962	52,962	52,962	56,5745	64,5623	64,5623	64,5623	64,5623	63,9735
similaritymatrix_050_30_1	5	10	15	70,1122	53,5753	53,5753	56,632	58,18	70,1122	70,1122	71,3751	71,3751	69,952
similaritymatrix_050_30_2	10	10	10	66,357	49,6035	49,6035	57,547	57,5471	66,357	66,357	62,7458	62,7458	65,9619
similaritymatrix_050_30_2	5	10	15	69,4465	52,1986	61,6406	64,864	64,8644	70,2836	69,4465	70,7588	70,7588	71,2278
similaritymatrix_050_30_3	10	10	10	66,5263	47,59	57,8224	58,687	59,2661	63,5379	66,5263	62,8842	62,8842	66,7117
similaritymatrix_050_30_3	5	10	15	69,0653	52,9062	59,6402	59,64	65,3579	69,713	69,0653	69,0653	69,0653	71,4714
similaritymatrix_050_30_4	10	10	10	61,9632	49,3887	56,1688	56,169	56,1697	61,9368	61,9632	61,9632	61,9632	64,9732
similaritymatrix_050_30_4	5	10	15	71,9277	53,2887	63,1277	63,128	64,7088	71,5924	71,9277	71,9277	71,9277	73,5727
similaritymatrix_050_30_5	10	10	10	60,0056	46,3072	53,6437	56,298	56,2984	60,0056	60,0056	60,5579	60,0056	61,4662
similaritymatrix_050_30_5	5	10	15	66,2944	55,0671	55,3218	56,532	59,407	64,8571	66,2944	66,2944	66,0931	67,634
similaritymatrix_075_30_1	10	10	10	78,6057	69,1793	69,1793	72,908	73,7507	78,6057	78,6057	78,9859	79,2328	79,9483
similaritymatrix_075_30_1	5	10	15	89,1697	74,5943	83,2269	79,216	83,2269	87,1078	89,1697	89,2297	89,2297	88,2783
similaritymatrix_075_30_2	10	10	10	82,6918	66,2943	66,2943	74,131	75,0183	82,6918	82,6918	83,7852	83,6455	84,4781
similaritymatrix_075_30_2	5	10	15	89,1913	77,3038	77,3038	84,38	86,5239	89,1913	89,1913	94,046	90,5242	92,3274
similaritymatrix_075_30_3	10	10	10	77,6871	66,661	66,661	72,968	72,9682	77,6871	77,6871	78,3554	78,3554	80,8648
similaritymatrix_075_30_3	5	10	15	88,5667	75,5506	75,5506	82,266	82,2657	88,5667	88,5667	88,5667	90,3354	90,3342
similaritymatrix_075_30_4	10	10	10	77,3737	65,3924	65,3924	67,384	67,7975	77,3737	77,3737	76,6704	80,5602	79,0843
similaritymatrix_075_30_4	5	10	15	86,5889	75,025	78,4725	78,473	79,2225	86,1947	86,5889	86,5889	84,1076	90,2881
similaritymatrix_075_30_5	10	10	10	79,3932	67,0372	67,0372	69,572	70,1798	79,3932	79,3932	79,3932	76,185	78,8103
similaritymatrix_075_30_5	5	10	15	91,4982	78,6459	78,6459	78,646	78,6459	91,4982	91,4982	91,4982	91,4982	91,1876

C.6. Relative error for K=3

				2-DIVISION				4-DIVISION
NAME	M_1	M_2	M_3	RELATIVE ERROR, S_0 = 3 s	RELATIVE ERROR, S_0 = 5 s	RELATIVE ERROR, S_0 = 10 s	RELATIVE ERROR, S_0 = 20 s	RELATIVE ERROR, S_0 = 20 s
similaritymatrix_025_10_1	3	3	4	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_1	2	3	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_2	3	3	4	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_2	2	3	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_3	3	3	4	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_3	2	3	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_4	3	3	4	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_4	2	3	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_5	3	3	4	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_025_10_5	2	3	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_1	3	3	4	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_1	2	3	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_2	3	3	4	0,00%	0,34%	0,00%	0,00%	0,00%
similaritymatrix_050_10_2	2	3	5	0,00%	1,15%	0,00%	0,00%	0,00%
similaritymatrix_050_10_3	3	3	4	0,00%	1,50%	0,00%	0,00%	0,00%
similaritymatrix_050_10_3	2	3	5	0,00%	1,92%	0,00%	0,00%	0,00%
similaritymatrix_050_10_4	3	3	4	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_050_10_4	2	3	5	0,00%	1,25%	0,00%	0,00%	0,00%
similaritymatrix_050_10_5	3	3	4	0,00%	0,68%	0,00%	0,00%	0,00%
similaritymatrix_050_10_5	2	3	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_1	3	3	4	0,00%	0,73%	0,00%	0,00%	0,00%
similaritymatrix_075_10_1	2	3	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_2	3	3	4	0,00%	1,82%	0,00%	0,00%	0,00%
similaritymatrix_075_10_2	2	3	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_3	3	3	4	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_3	2	3	5	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_4	3	3	4	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_4	2	3	5	0,00%	0,46%	0,00%	0,00%	0,00%
similaritymatrix_075_10_5	3	3	4	0,00%	0,00%	0,00%	0,00%	0,00%
similaritymatrix_075_10_5	2	3	5	0,00%	2,95%	0,00%	0,00%	0,00%
similaritymatrix_025_20_1	7	7	6	4,32%	3,45%	2,37%	1,99%	0,00%
similaritymatrix_025_20_1	3	7	10	6,03%	0,00%	6,09%	6,48%	5,52%
similaritymatrix_025_20_2	7	7	6	1,35%	4,18%	1,35%	0,11%	1,25%
similaritymatrix_025_20_2	3	7	10	9,25%	0,95%	6,35%	2,36%	3,15%
similaritymatrix_025_20_3	7	7	6	2,32%	2,77%	2,08%	1,50%	1,48%
similaritymatrix_025_20_3	3	7	10	3,17%	2,06%	0,52%	0,65%	0,36%
similaritymatrix_025_20_4	7	7	6	4,03%	7,78%	1,04%	2,96%	1,21%
similaritymatrix_025_20_4	3	7	10	7,79%	1,36%	6,23%	2,84%	2,76%
similaritymatrix_025_20_5	7	7	6	2,61%	3,10%	3,40%	0,00%	0,00%
similaritymatrix_025_20_5	3	7	10	0,74%	0,38%	0,74%	0,00%	0,00%
similaritymatrix_050_20_1	7	7	6	6,95%	1,53%	5,49%	5,09%	0,00%
similaritymatrix_050_20_1	3	7	10	1,86%	8,50%	4,20%	0,00%	0,00%
similaritymatrix_050_20_2	7	7	6	1,66%	6,72%	1,29%	3,87%	0,86%
similaritymatrix_050_20_2	3	7	10	2,57%	0,00%	0,65%	0,00%	0,00%

similaritymatrix_050_20_3	7	7	6	0,13%	0,77%	0,00%	0,58%	0,00%
similaritymatrix_050_20_3	3	7	10	2,55%	0,34%	1,55%	1,41%	0,25%
similaritymatrix_050_20_4	7	7	6	3,05%	4,02%	5,67%	4,25%	1,00%
similaritymatrix_050_20_4	3	7	10	5,82%	1,09%	5,06%	3,84%	2,41%
similaritymatrix_050_20_5	7	7	6	2,23%	2,40%	2,11%	0,71%	0,70%
similaritymatrix_050_20_5	3	7	10	2,99%	0,27%	4,10%	4,28%	0,00%
similaritymatrix_075_20_1	7	7	6	0,47%	2,57%	0,00%	0,00%	0,00%
similaritymatrix_075_20_1	3	7	10	0,00%	2,47%	4,01%	0,01%	0,00%
similaritymatrix_075_20_2	7	7	6	3,77%	0,35%	0,00%	0,00%	0,00%
similaritymatrix_075_20_2	3	7	10	2,27%	2,58%	1,39%	1,41%	0,90%
similaritymatrix_075_20_3	7	7	6	2,06%	1,50%	2,51%	2,58%	1,02%
similaritymatrix_075_20_3	3	7	10	3,69%	2,29%	0,00%	0,00%	0,00%
similaritymatrix_075_20_4	7	7	6	2,93%	1,70%	1,95%	1,19%	0,95%
similaritymatrix_075_20_4	3	7	10	7,67%	0,00%	2,10%	0,00%	0,00%
similaritymatrix_075_20_5	7	7	6	6,34%	4,44%	3,77%	3,68%	1,04%
similaritymatrix_075_20_5	3	7	10	2,69%	1,68%	2,69%	3,18%	2,81%
similaritymatrix_025_30_1	10	10	10	5,76%	3,65%	3,23%	4,64%	2,05%
similaritymatrix_025_30_1	5	10	15	7,37%	1,18%	6,45%	6,89%	3,81%
similaritymatrix_025_30_2	10	10	10	4,50%	7,54%	8,29%	9,03%	2,35%
similaritymatrix_025_30_2	5	10	15	10,50%	4,70%	2,98%	3,07%	1,05%
similaritymatrix_025_30_3	10	10	10	2,98%	1,77%	4,74%	4,98%	3,47%
similaritymatrix_025_30_3	5	10	15	7,06%	2,83%	2,52%	2,58%	0,79%
similaritymatrix_025_30_4	10	10	10	7,00%	4,76%	4,47%	4,68%	1,33%
similaritymatrix_025_30_4	5	10	15	3,66%	1,10%	3,71%	3,85%	4,56%
similaritymatrix_025_30_5	10	10	10	8,11%	4,33%	4,70%	4,93%	2,94%
similaritymatrix_025_30_5	5	10	15	5,96%	0,00%	2,87%	5,21%	2,13%
similaritymatrix_050_30_1	10	10	10	0,16%	0,51%	0,16%	0,16%	1,07%
similaritymatrix_050_30_1	5	10	15	4,75%	2,22%	3,03%	3,12%	4,96%
similaritymatrix_050_30_2	10	10	10	0,00%	2,04%	5,44%	5,76%	0,60%
similaritymatrix_050_30_2	5	10	15	2,55%	0,59%	1,89%	1,92%	1,24%
similaritymatrix_050_30_3	10	10	10	5,20%	0,14%	6,18%	6,59%	0,47%
similaritymatrix_050_30_3	5	10	15	7,53%	0,32%	8,39%	9,15%	5,19%
similaritymatrix_050_30_4	10	10	10	8,61%	4,25%	8,57%	9,38%	4,13%
similaritymatrix_050_30_4	5	10	15	3,05%	1,41%	2,59%	2,66%	0,37%
similaritymatrix_050_30_5	10	10	10	4,08%	1,58%	3,19%	4,25%	1,74%
similaritymatrix_050_30_5	5	10	15	5,87%	3,05%	3,78%	4,25%	1,84%
similaritymatrix_075_30_1	10	10	10	2,07%	2,94%	1,59%	1,30%	0,40%
similaritymatrix_075_30_1	5	10	15	4,48%	1,62%	2,15%	2,20%	3,19%
similaritymatrix_075_30_2	10	10	10	2,11%	0,71%	0,82%	1,00%	0,00%
similaritymatrix_075_30_2	5	10	15	6,74%	0,53%	1,66%	5,65%	3,46%
similaritymatrix_075_30_3	10	10	10	3,91%	1,76%	3,08%	3,18%	0,02%
similaritymatrix_075_30_3	5	10	15	3,29%	1,06%	3,29%	1,38%	1,36%
similaritymatrix_075_30_4	10	10	10	4,02%	2,10%	4,90%	0,07%	1,90%
similaritymatrix_075_30_4	5	10	15	4,53%	0,23%	4,10%	7,35%	0,00%
similaritymatrix_075_30_5	10	10	10	1,62%	1,55%	1,62%	5,93%	2,35%
similaritymatrix_075_30_5	5	10	15	0,65%	0,49%	0,65%	0,66%	0,99%

List of Tables

List of Figures

Bibliography

[1] L. De Giovanni, *Ricerca Operativa,* Dipartimento di matematica pura e applicata (Torre Archimede), Università di Padova.
https://www.math.unipd.it/~luigi/courses/rodid/m00.intro.01.pdf

[2] K. Burger, L. White, M. Yearworth, 2019, *Developing a smart operational research with hybrid practice theories*, European Journal of Operational Research.

[3] Dispense di ricerca operativa.
http://www.ing.unisannio.it/boccia/files/ricerca_operativa.pdf

[4] M.Roma, 2015, *Appunti delle lezioni di ricerca operativa*.
http://www.diag.uniroma1.it//~roma/didattica/RO15-16/cap1-2-3.pdf

[5] G. Marco, *The essentials of combinatorial optimization,* Slides A.A 2020/2021, Politecnico di Torino.

[6] Y. Bengio, A. Lodi, A Prouvost, 2021, *Machine learning for combinatorial optimization: A methodological tour d'horizon*, European Journal of Operational Research.

[7] IBM ILOG CPLEX Optimization Studio CPLEX User's Manual, 2017.
https://www.ibm.com/docs/en/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/usrcplex.pdf

[8] D. Vigo, 2005, *Programmazione Lineare Intera: III – Algoritmo Branch and Cut,* DEIS – Università di Bologna.
http://www.or.deis.unibo.it/didatt_pages/mmsd_ce/Pmat_ILP_BC_rev2.pdf

[9] L. De Giovanni, G. Zambelli, *Metodi e modelli per l'ottimizzazione combinatoria*, Dipartimento di matematica pura e applicata (Torre Archimede), Università di Padova.
https://www.math.unipd.it/~luigi/courses/metmodoc0809/c12-16.01.ip.pdf

[10] D. Salvagnin, 2011, *Cenni di Programmazione Lineare Intera.*
http://www.dei.unipd.it/~salvagni/didattica/mip

[11] M. Gnagi, P. Baumann, 2021, *A matheuristic for large-scale capacitated clustering,* Computers & Operations Research.

[12] S. Orlando, *Clustering,* Università di Venezia.
https://www.dsi.unive.it/~dm/Slides/2_Cluster.pdf

[13] G. Calafiore, *Clustering,* Optimization for machine learning, A.A 2020/2021, Politecnico di Torino.

[14] A. Billionnet, 2005, *Different formulations for solving the heaviest k-subgraph problem*, Information Systems and Operational Research.

[15] G.M. Goncalves, L.L. Lourenco, 2019 *Mathematical formulations for the K clusters with fixed cardinality problem,* Computers & Industrial Engineering.

[16] CEDRIC'S LIBRARY
cedric.cnam.fr/~lamberta/Library/k-cluster/index.php

[17] A. Agnetis, C. Meloni, *Appunti su metodi metaeuristici di ricerca,* Dipartimento di ingegneria dell'Informazione, Università di Siena.
https://www3.diism.unisi.it/~agnetis/locals.pdf

[18] R.A Lopes, A.R.R Freitas, R.C.P. Silva, 2021, *Local Search with Groups of Step Sizes*, Operations Research Letters.