Daniel Lemes Gribel

Hybrid Genetic Algorithm for the Minimum Sum-of-Squares Clustering Problem

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Thibaut Victor Gaston Vidal

Rio de Janeiro
March 2017

Daniel Lemes Gribel

Hybrid Genetic Algorithm for the Minimum Sum-of-Squares Clustering Problem

Dissertation presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the undersigned Examination Committee.

Prof. Thibaut Victor Gaston Vidal
Advisor
Departamento de Informática – PUC-Rio

Prof. Marcus Vinicius Soledade Poggi de Aragão
Departamento de Informática – PUC-Rio

Prof. Marco Serpa Molinaro
Departamento de Informática – PUC-Rio

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies
Centro Técnico Científico – PUC-Rio

Rio de Janeiro, March 20th, 2017

Daniel Lemes Gribel

Obtained a bachelor's degree in Information Systems from the Federal University of Rio de Janeiro State (Unirio, Rio de Janeiro, Brazil). He earned a scholarship from CNPq in his Masters at PUC-Rio.

Bibliographic data

Gribel, Daniel Lemes

Hybrid Genetic Algorithm for the Minimum Sum-of-Squares Clustering Problem / Daniel Lemes Gribel; advisor: Thibaut Victor Gaston Vidal. – 2017.

54 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2017.

Inclui bibliografia

1. Informática – Teses. 2. Clusterização. 3. Meta-heurística. 4. Aprendizado não-supervisionado. 5. Mínima soma dos quadrados. 6. Mineração de dados. I. Vidal, Thibaut Victor Gaston. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor, Thibaut Vidal, for his great support during the Master's program. His patience, motivation and immense knowledge helped me a lot in conducting this research.

I would also like to thank CNPq for the funding received through a scholarship during the entire duration of this degree.

Last but not least, I would like to thank my parents, my brothers and my friends for providing me with unfailing support and continuous encouragement.

Abstract

Gribel, Daniel Lemes; Vidal, Thibaut Victor Gaston (Advisor). **Hybrid Genetic Algorithm for the Minimum Sum-of-Squares Clustering Problem**. Rio de Janeiro, 2017. 54p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Clustering plays an important role in data mining, being useful in many fields that deal with exploratory data analysis, such as information retrieval, document extraction, and image segmentation. Although they are essential in data mining applications, most clustering algorithms are adhoc methods. They have a lack of guarantee on the solution quality, which in many cases is related to a premature convergence to a local minimum of the search space. In this research, we address the problem of data clustering from an optimization perspective, where we propose a hybrid genetic algorithm to solve the Minimum Sum-of-Squares Clustering (MSSC) problem. This meta-heuristic is capable of escaping from local minima and generating near-optimal solutions to the MSSC problem. Results show that the proposed method outperformed the best current literature results – in terms of solution quality – for almost all considered sets of benchmark instances for the MSSC objective.

Keywords

Clustering; Meta-heuristic; Unsupervised learning; Minimum sum-of-squares; Data mining.

Resumo

Gribel, Daniel Lemes; Vidal, Thibaut Victor Gaston (Orientador). **Algoritmo Genético Híbrido para o Problema de Clusterização Minimum Sum-of-Squares**. Rio de Janeiro, 2017. 54p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Clusterização desempenha um papel importante em data mining, sendo útil em muitas áreas que lidam com a análise exploratória de dados, tais como recuperação de informações, extração de documentos e segmentação de imagens. Embora sejam essenciais em aplicações de data mining, a maioria dos algoritmos de clusterização são métodos ad-hoc. Eles carecem de garantias na qualidade da solução, que em muitos casos está relacionada a uma convergência prematura para um mínimo local no espaço de busca. Neste trabalho, abordamos o problema de clusterização a partir da perspectiva de otimização, onde propomos um algoritmo genético híbrido para resolver o problema Minimum Sum-of-Squares Clustering (MSSC, em inglês). A meta-heurística proposta é capaz de escapar de mínimos locais e gerar soluções quase ótimas para o problema MSSC. Os resultados mostram que o método proposto superou os resultados atuais da literatura – em termos de qualidade da solução – para quase todos os conjuntos de instâncias considerados para o problema MSSC.

Palavras-chave

Clusterização; Meta-heurística; Aprendizado não-supervisionado; Mínima soma dos quadrados; Mineração de dados.

Table of Contents

List of Figures

List of Tables

1 Introduction

Clustering plays an important role in data mining, being useful in many fields that require the separation of objects into meaningful groups. Clustering arises whenever one has a collection of objects or patterns – say, a set of photographs, documents, or micro-organisms – and is trying to classify or organize them into coherent groups (20). It has found many applications in fields that deal with exploratory data analysis, such as information retrieval, document extraction, and image segmentation.

Due to permanent social and physical interactions, human beings have developed the ability of clustering objects, once we are required to do it and draw conclusions from it in a large variety of daily situations. Although the human brain works very well (and quickly) on clustering some objects in two or three dimensions, it is not capable of performing the same task in a very large number of dimensions or in the presence of a large amount of data. Thus, one of the fundamental questions on clustering is: "How can we propose some means to lead the human cognitive capacity of quickly separating objects in a low dimension of features to a high dimension?". Most of research on clustering analysis is spending efforts on answering this question in a satisfactory way (8).

Numerous definitions of clustering analysis have been proposed in the literature. Clustering is typically oriented to study the internal structure of data (19), but it can assume different definitions and roles according to the domain of study and goals. Even finding a good criteria to validate the output of a clustering algorithm is not trivial. As clustering is a generalization of what humans perceive, it is depicted by many informal definitions and guided by subjective criteria, revealing the difficulty of providing a singular and sufficiently broad definition. Beyond this, clustering criteria are sometimes known and well defined, but results should be given according to a specific and adequate perspective that better attends someone view of the problem. For example, some criteria to validate clusters consider the cohesion and/or separability of patterns. The former asserts that patterns in one cluster should be as similar to each other as possible. The latter asserts that clusters should be well separated, i.e., patterns in different clusters should be as different to each other as possible. These different perspectives have led to the development of different classes of algorithms.

Although they are essential in data mining applications, most clustering algorithms are ad-hoc methods, which can be assimilated to a greedy construction procedure or local search for some optimization problem. These methods are designed for specific clustering criteria, being not adapted to different purposes. In addition, they have no optimality guarantee and usually have volatile performance in the domain of the optimization problem to which they are attached. Thus, these methods work relatively well for certain applications but fail in others.

K-means is an example of a very popular algorithm for clustering, based on iterative updates of cluster centers until convergence. Although iterative algorithms are easy to implement and usually demand low computational time (linear in the number of samples in K-means), they have typically two disadvantages (8): (i) they are likely to converge to a local optimum; (ii) their performance is usually very sensitive to the initial conditions of the search; and (iii) they are highly dependent of the shape of clusters.

The main motivation of this work is to overcome the lack of guarantee on the solution quality of classical clustering algorithms. For this purpose, this research addresses the problem of data clustering from an optimization perspective, where we explore a particular formulation, the Minimum sum-of-squares clustering (MSSC) in the Euclidean space. In addition, the limitation imposed by classical ad-hoc methods is in many cases related to premature convergence. To overcome these issues, the present work proposes a meta-heuristic capable of escaping from local minima and generating near-optimal solutions to the MSSC problem.

The MSSC is the most treated formulation in data clustering literature, since it expresses both homogeneity and separation (16). The Euclidean MSSC corresponds to the minimization of the sum-of-squares of Euclidean distances of objects to their cluster means, or equivalently, to the minimization of within-group sum-of-squares (30). In this work we refer to objects or samples to be clustered as data points.

The method proposed in this document is a hybrid genetic algorithm that combines a K-means local improvement procedure with crossover, mutation and diversification operators. This allows to efficiently escape from local minima and reach high quality solutions. The method outperforms the best current literature results for all considered sets of benchmark instances in terms of solution quality for the MSSC model. In some cases, the results of the K-means algorithm are more than 1100% off the solution quality of the proposed method. Additionally, as this work is an accurate and stable method for the MSSC objective, it is also an important starting point to solve different objectives in clustering. Although the scope of the present work is focused on solving the MSSC problem, the long-term goal in this research is to have a general framework that solves any classical clustering model. As we tackle the clustering problem from an optimization perspective through a meta-heuristic, we can elaborate a neighborhood structure that is general enough to deal with different models, where the evaluation cost of a solution varies according to the objective. Thus, the proposed method is an important instrument to the coming step of this work, which consists in testing different clustering models and verifying their impact on clustering results.

This document is structured as follows: in Chapter 2, we discuss the differences between the early ad-hoc and the model-based approaches for clustering, focusing in the case of MSSC. In Chapter 3, we describe the proposed methodology, a meta-heuristic designed to solve the MSSC problem. In Chapter 4, we present the experimental results and the analysis we draw from them. Finally, the final remarks and the future perspectives are addressed in Chapter 5.

2
Problem Statement and Literature Review

Most algorithms proposed for data clustering are based on heuristics with intuitive procedures (11). These algorithms are also referred as ad-hoc methods, since they are solutions designed for specific clustering criteria, being non-generalizable and not adaptive. As these ad-hoc methods are based on greedy constructions or local improvements, they have some limitation regarding solution quality and local optimum convergence.

There is little systematic guidance in such algorithms for solving important questions in cluster analysis (11). K-means is an example of a very popular ad-hoc algorithm for clustering, based on iterative updates of cluster centers until convergence. Although K-means is easy to implement and usually demand low computational time, it has not any optimality guarantee and is highly dependent on initial conditions to deliver good outcomes.

However, in the recent years there has been a considerable growth of clustering methods designed in a model-oriented way. This approach is based on formal models that usually produce more accurate solutions than ad-hoc methods as they clearly state an objective to be achieved.

2.1
Ad-hoc methods

The ad-hoc methods proposed for data clustering usually belong to two large classes: hierarchical and partitional algorithms.

Hierarchical clustering works by successively agglomerating or separating clusters at each stage according to some distance measure. The methods of this class can be roughly separated into two groups: agglomerative and divisive methods. In an agglomerative approach, each data point starts at its own cluster and at each stage of the algorithm it joins the two most similar clusters, until a single cluster containing all data points is reached. The divisive method is the opposite. The method proceeds by separating n data points successively into finer groups.

Several algorithms for data clustering are based on hierarchical constructions, once they do not require the number of clusters a priori and are independent from initial conditions, as strategic starting points. However, hierarchical methods are computationally expensive – usually O(n^2 log n) for computational complexity – and all decisions about joining/separating sets of points are definitive, i.e., it is impossible to revert any merge or split operation. BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies) (32) and CURE (Clustering Using Representatives) (12) are examples of popular hierarchical clustering algorithms.

The methods belonging to partitional clustering directly divide data points into some pre-specified number of clusters without any hierarchical structure (31). It is composed of methods based on iterative relocation of data points between clusters that, at each iteration, reduce the value of some criteria function until convergence (19). The minimum sum-of-squares function is one of the most widely used criteria, which aims to minimize the sum of each data point to the center of its cluster. K-means and PAM (Partitioning Around Medoids) are quite widespread partitioning clustering methods. Although partitional clustering algorithms are suitable for large data sets due to efficiency in computational time, they require the number of clusters in advance and are heavily dependent on the initial conditions, which can lead algorithms like K-means to a premature convergence (8).

Due to the popularity of K-means, several works have been dedicated to proposing more efficient implementations, speed up techniques and smart data structures for it. The work of Hamerly (13), in particular, provides an efficient K-means algorithm which works with a complexity of $O(nmd + md^2)$ per iteration, where n, m and d are the number of data points, the number of clusters and the dimensionality of data, respectively. However, this implementation is much faster in practice as it avoids most of the K-means innermost loops.

2.2 Model-driven clustering

The above classes of ad-hoc methods are not guided by a model, but by iterative constructions or re-assignments according to some underlying criteria. For instance, in K-means algorithm, the criteria is to assign data points to the closest center after the calculation of centers position. In hierarchical methods, the criteria to merge/split a solution is based on a distance metric between pairs of clusters. Due to these characteristics, it is difficult to assess if erratic results come from inappropriate methods, or from a data that does not intuitively leads to the result that we desire.

In order to detect this kind of behaviour, some solutions have been designed to face data clustering from a model-oriented perspective, which means giving a formal definition of the clustering problem as an optimization problem, where clear objective functions and restrictions are set. Therefore, one can apply a method that minimizes/maximizes an objective function to solve the model.

The work of Hansen and Jaumard (14) states that most of clustering tasks are composed by the following elements:

– *Data.* Observation of d characteristics in n entities (patterns). This yields a $n \times d$ data matrix.

– *Similarities.* Computation of the similarities between entities, i.e., the derived measures from features that indicates how similar (dissimilar) are each pair of entities.

– *Constraints.* Specification of existing constraints in the clustering task. For instance: number of clusters, maximum cardinality per cluster, number of clusters an entity can be assigned to, etc.

– *Criterion.* Selection of the criterion to express homogeneity or separability of the clusters.

– *Algorithm.* Design of the algorithm to solve the clustering problem.

– *Interpretation.* Analysis on how meaningful are the generated clusters. For instance: use of descriptive statistics and data mining indicators.

Except for *Interpretation*, the above steps that characterize a general clustering task are profitably addressed by the mathematical programming perspective, as it is capable of explicitly define and delimit the problem.

Many formulations can be done to express a clustering task. Besides that, some optimization problems are more suitable for specific clustering tasks, or specific data sets. In this work, we consider a particular formulation for data clustering: the Minimum sum-of-squares clustering (MSSC). The MSSC is among the most studied problems in cluster analysis and has been the focus of an extensive literature. Due to the large number of works considering this formulation and to its intuitive way of understanding a solution for data clustering, the MSSC was chosen to be tackled. Actually, the long-term purpose of this work is to extend the proposed method to a range of formulations. Thus, the MSSC is good starting point to validate the method, as much material is available in the literature for comparison.

2.3 The MSSC problem

The MSSC problem – also referred as the *discrete clustering* problem or the *hard clustering* problem – has been extensively studied in the literature for data clustering, possibly because this is the natural model which is addressed by the K-means algorithm. It was first formulated mathematically by Vinod (28), where the MSSC is defined as a problem that assumes integer variables that can take values 0 or 1 only, recognizing it as an Integer programming problem.

The objective in the MSSC problem is to minimize the total sum of the distances of each data point to the mean point in its cluster. We are given a set X of n data points in a d-dimensional space $\mathbb{R}^d$ (4):

$$X = \{x_1, x_2, ..., x_n\}, \text{ where } x_i \in \mathbb{R}^d, \quad i = 1, ..., n.$$

The clustering procedure must partition the set X into m disjoint and non empty subsets $\{S_1, \ldots, S_m\}$, such that $S_i \cap S_j = \emptyset$ for all (i, j) and $S_1 \cup \cdots \cup S_m = X$, where m is the number of desired clusters. In other words, a clustering algorithm aims to generate a partition that groups data consistently, with most similar data points belonging to the same group and dissimilar data points belonging to different groups.

In order to define the similarity between any two points in the space, a function d assigns to any pair $x, y \in \mathbb{R}^d$ a distance metric $d(x, y) \in \mathbb{R}$. Many distance measures could be used to characterize how similar two points or patterns are. A popular one in the domain of data mining is the Euclidean distance, that can often be used to reflect the similarity between two patterns when considering multiple dimensions (features) (18):

$$d(x, y) = \sqrt{\sum_{q=1}^{d} (x_q - y_q)^2} = \|x - y\|, \quad x, y \in \mathbb{R}^d \tag{2-1}$$

Here, x_q is the value of the q-th feature of a point x, d is the number of features and $q = \{1, ..., d\}$. Analogously, the set of data points $x_i \in X$ can be described as a matrix $X'_{n \times d}$, where an entry x_{iq} is the value of the q-th feature for the i-th data point. In this work we consider that points (including the centroids) are in the Euclidean space, so the measure above (2-1) is used to express the distance between any two points. Therefore, the MSSC problem can be formulated as the following set partitioning problem:

$$\text{Minimize} \sum_{k=1}^{|S|} z_k y_k \tag{2-2}$$

$$\sum_{k=1}^{|S|} a_{ik} y_k = 1, \quad \forall i \tag{2-3}$$

$$\sum_{k=1}^{|S|} y_k = m \tag{2-4}$$

$$y_k \in \{0,1\}, \quad \forall k \tag{2-5}$$

where S is the set of all possible subsets obtained from elements in X, $|S|$ is the size of S; $a_{ik} = 1$ if $x_i \in S_k$ and $a_{ik} = 0$ otherwise; y_k are the decision variables, with $y_k = 1$ if the subset S_k is chosen and $y_k = 0$ otherwise; and z_k is the cost function on subset S_k, i.e., the cost within the k-th subset. In the MSSC problem, the centroid is the mean point c_k of cluster S_k. Thus, the contribution z_k of each subset to the objective function is:

$$z_k = \sum_{x_i \in S_k} d(x_i, c_k) \tag{2-6}$$

$$\text{where } c_k = \frac{\sum_{i=1}^{n} a_{ik} x_i}{\sum_{i=1}^{n} a_{ik}}$$

2.3.1
Computational complexity

Regarding computational complexity, the MSSC can be solved in $O(n^3)$ time for $m \geq 2$ and one dimensional data (27). For general $m \geq 2$ and general dimension d, the MSSC is NP-hard (2). If both m and d are fixed, the problem can be solved in $O(n^{md+1})$ time (17), which may be very time-consuming. However, the hardness of MSSC is not measured only by the number of points (samples), dimensions and clusters, it also depends on the distribution of points (3).

2.3.2
Solution techniques

Many solution techniques have been developed in recent years to solve the MSSC problem. These techniques can be separated into different categories, based on their exact or heuristic nature, whether they are deterministic or probabilistic, whether they process complete solutions or construct solutions during the search, and finally whether they maintain a single candidate solution or have a population of solutions (8). This range of methods includes construction methods and local searches, which aims to repeatedly seek for better solutions according to some fitness; meta-heuristic algorithms;

and mathematical programming techniques. In this section, we review some techniques and previous works that have been designed for the resolution of the MSSC problem.

Hansen (15) proposed a local search called J-MEANS for the MSSC problem. The neighbourhood of a current solution is defined by all possible centroid-to-object relocations followed by corresponding changes of assignments. Thus, a move in J-MEANS corresponds to replacing an existing centroid x_i by a data point x_j. The move is done by selecting the pair of indices (i, j) that brings the highest gain in the objective function. After the replacement of x_i by x_j, the corresponding assignment updates are done. Moves are made in these neighbourhoods whenever the value of a neighbour's objective function is better than the current solution, until a local optimum is reached. As the J-MEANS is a general local search, it is also applied to fit into meta-heuristics as the Variable Neighbourhood Search (VNS). The J-MEANS local search seems to work well in some problem instances where the number of clusters is large, so that data points could be the centroids of some clusters in the current solution.

Recently, different incremental algorithms have been developed to address the choice of initial solutions in the K-means algorithm. Incremental clustering algorithms start from an initial solution with $k-1$ centers for the $(k-1)$-clustering problem and attempt to optimally add one new cluster by placing the $k-$th center in an appropriate position. The global K-means (23) is an incremental algorithm that solves the MSSC problem by considering each data point as a candidate for the $k-$th cluster center.

A modified version of the global K-means (MGKM) proposed by (5) tackles the MSSC by producing initial solutions through the resolution of an auxiliary clustering problem, rather then testing all data points as candidate centers. Experimental results demonstrate that MGKM is able to find better solutions than the global K-means, although it requires more computational effort.

The work of Ordin and Bagirov (26) considered the MSSC as a global optimization problem and introduced a multi-start modified global K-means (MS-MGKM) algorithm to improve the accuracy of MGKM. The MS-MGKM is an adaptation of MGKM that produces a set of initial solutions that undergo K-means, rather than considering only one initial solution at each K-means stage. The proposed method was applied on 16 real-world data sets and experiments show that it is more accurate than MGKM.

More recently, (6) presented an incremental algorithm based on the difference of convex representations for solving the MSSC problem, where a method is designed to solve non-smooth optimization problems at each iteration of the incremental algorithm. Large data sets ranging from tens to hundreds of thousands data points are used, and results show that this approach is efficient for solving large data sets when compared to previous incremental algorithms like MGKM and global K-means.

Many authors proposed some hybrid methods, that combine both classical algorithms with meta-heuristics. (21) and (25) uses K-means as a search operator inside a genetic algorithm (GA) on the MSSC problem, in such a way that each GA candidate solution is used as a starting point for K-means. Thus, K-means assumes the role of a local search inside a broader optimization process guided by the GA.

The work of Festa (10) has also treated the MSSC problem with emphasis on combinatorial optimization perspective. The proposed approach considers a biased random-key genetic algorithm (BRKGA), which was applied to biological data clustering. In random-key genetic algorithms (RKGA), candidate solutions are represented as vectors of randomly generated real numbers in the interval (0, 1]. Each vector can thus be associated to a solution in the combinatorial optimization problem, for which an objective value or fitness can be computed. To evolve the population of solutions, additional individuals are produced in order to complete the new population. This is done by crossing parent solutions. The fundamental difference between BRKGA and RKGA resides in the way parents are selected for crossing. RKGA uses any two solutions to produce a child individual, once BRKGA crosses a elite solution (good regarding fitness) with a non-elite to produce a child.

3
Proposed Methodology

This chapter describes the proposed meta-heuristic to solve the MSSC problem. The designed method is based on a genetic algorithm (GA) with local improvements combined with mechanisms that allow the diversification of the population (local minima escape) and the propagation of good solutions. In this method, the local improvement procedure is performed via the running of the K-means algorithm, which takes one candidate solution in the GA population as a starting point. In other words, the method may be defined as a multi-start K-means inside a GA framework, which in turn is guided by the MSSC objective.

3.1 General structure

The GA here proposed works through the following simple cycle of stages:

1. Creation of an initial population of individuals (candidate solutions);
2. Selection of parents;
3. Genetic manipulation to create new individuals (crossover and mutation);
4. Enhancement of the produced individual (local improvement);
5. Selection of survivors individuals for propagation.

This general scheme of the meta-heuristic is described in more details by Algorithm 1. Initially, the method generates a random population of individuals (see 3.3). A population is a set of solutions, where each individual represents a point in a search space of the optimization problem. Then, it applies successively a number of operators to evolve this population. Firstly, it selects two parent individuals (see 3.4.1) from the population and combines them by a crossover procedure (see 3.4.2), yielding to a new individual (offspring) that is added to the population. Secondly, the offspring is mutated (see 3.4.3) and enhanced by a local improvement (see 3.4.4), generating a new individual solution that is also added to the population. These two steps –

crossover and mutation with enhancement – are performed many times until a termination criteria is reached.

In addition to the genetic operators mentioned above, a mechanism to select the survivor individuals is applied when a pre-defined criteria is reached (see 3.5). This mechanism allows the method to propagate the best individuals and keep the diversity of the population. The following sub-sections describe how these operators and mechanisms work to manage the population of solutions.

Algorithm 1 Genetic algorithm framework

```
 1: Initialize population
 2: while number of iterations without improvement < $I_S$ and $I_{max}$ not
    reached do
 3:     Select parents $p_1$ and $p_2$
 4:     Generate an offspring $\theta$ from $p_1$ and $p_2$ (crossover)
 5:     Generate an individual $\theta'$ by mutating $\theta$ (mutation)
 6:     Apply local improvement on $\theta'$
 7:     Add $\theta$ and $\theta'$ to the population
 8:     if population size is equal to the maximum size $\Pi$ of population then
 9:         Select survivors
10:     end if
11: end while
12: Return best solution
```

3.2 Solution representation

A solution (clustering partition) is represented by two direct encodings: (i) the point-cluster assignment and (ii) the centroids description. The idea of the point-cluster assignment is to use a genetic encoding that allocates directly n data points to m clusters, such that each candidate solution consists of a n-vector (n genes) with integer values in the interval [1, m]. Thus, for n = 4 and m = 3, the encoding (1,1,3,2) allocates the first and the second data points to cluster 1; the third data point to cluster 3 and the fourth data point to cluster 2, generating the partition ({1,2}, {4}, {3}).

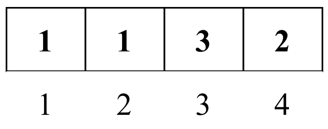


Figure 1: Assignment encoding for n = 4 and m = 3

The encoding based on the centroids description defines a solution through the feature vectors existing in each centroid. A solution is represented by a matrix C of size $m \times d$, where m is the number of centroids (clusters), d is the number of features and an entry c_{ij} of C is the value of the j-th feature of the i-th centroid ($i = 1 \ldots m$; $j = 1 \ldots d$) (see Figure 2).

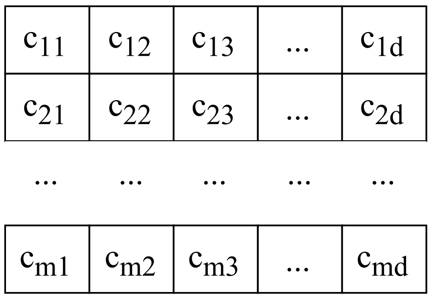

c_{11} c_{12} c_{13} ... c_{1d}
c_{21} c_{22} c_{23} ... c_{2d}
... ...
c_{m1} c_{m2} c_{m3} ... c_{md}

Figure 2: Encoding based on centroids features

3.3 Initial population

The first step in a GA is the generation of an initial population. It has been recognized that if the initial population of the GA is good, then the algorithm has a greater possibility of finding a good solution (7, 33). In this work, the initial population is randomly generated, by assigning each data point to a cluster according to a discrete uniform distribution, i.e., where each outcome is equally likely to happen. To compose the initial population, 100 individuals are created. Then, each initial individual is submitted to the local improvement. Some factors can influence the initial population or should be taken into account when an initial population is generated randomly: the search space, the fitness function, the diversity, and the number of individuals (9).

3.4 Individuals management

In order to increase the population and propagate good solutions, the algorithm needs to keep introducing new individuals. The generation of a new individual (offspring) begins with the random selection of two parents, p_1 and p_2, which are submitted to a crossover procedure that generates a child individual θ. Then, θ is added to the population. As highly fit solutions have more chances to be selected for reproduction, the offspring – which combines characteristics from each parent – is likely to have a good fitness. In addition to the crossover, the method also considers the mutation operator, which generates a new individual θ' that is similar to θ and tends to be good regarding the fitness.

3.4.1 Selection

The selection is the stage where individuals from the population are chosen to mate and generate a new individual. Due to the evolutionary behaviour of GA, the most likely individuals to be chosen for selection are the ones with good fitness. That way, good fragments of solutions can be propagated to generate good children solutions. In the proposed method, the parent selection is done through a w-tournament, which randomly selects w individuals (in a discrete uniform distribution) from the population and keeps the one with the best fitness among the w individuals to set the first parent. The fitness here considered is the value of the objective function (cost) of a solution. Then, the same selection scheme is performed for the second parent. The value of w was chosen according to a calibration process described in Section 4.2.

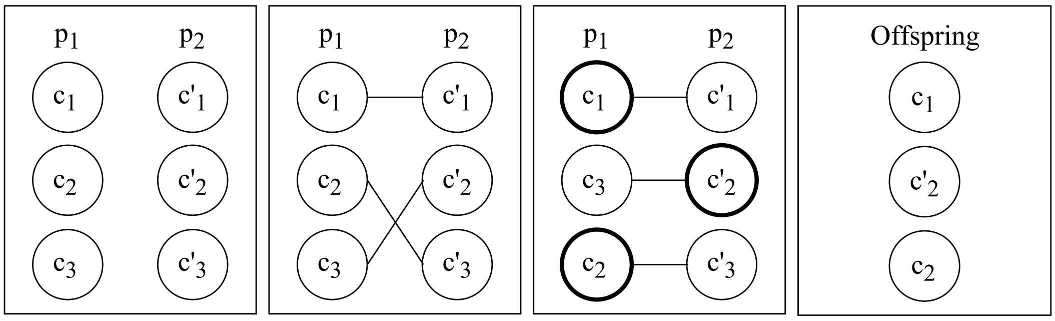
3.4.2 Crossover

After selecting two parents p_1 and p_2, these solutions are submitted to a crossover procedure in order to produce an offspring (child) solution. The crossover operator works as follows:

1. Firstly, a solution for the minimum weighted bipartite matching between centroids of p_1 and p_2 is found (Figure 3 (b)). Finding such a matching is known as the assignment problem. Let $G = (V, E)$ be a complete bipartite graph whose node set can be partitioned as $V = X \cup Y$, with the property that every edge $e \in E$ has one end in X and the other in Y, and every node of X is connected to every node of Y. Consider also that $|X| = |Y| = m$. An assignment M in G is the subset of edges $M \subseteq E$ of minimum cost, such that each node appears in exactly one edge in M (20). In our case, the nodes in X and Y correspond to the centroids of p_1 and p_2, respectively. The cost of an edge $e \in E$ is the Euclidean distance between two centroids $c' \in X$ and $c'' \in Y$. Thus, the objective is to produce the one-to-one assignment of centroids, in such a way that the sum of all distances considered in the assignment is minimized. In other words, the goal of this step is to join the similar centroids of different individuals. We solve this matching problem with the Hungarian algorithm (22), using the specific implementation of *dlib* C++ library (King, 2011), which works with a complexity of $O(m^3)$ and leads to m pairs of centroids, where m is the number of clusters (centroids).

2. For each pair of centroids resulted from the assignment solution, one of them is randomly selected (Figure 3 (c)) and set as a centroid of the offspring, resulting in a solution with m centroids, each coming from p_1 or p_2 (Figure 3 (d)).

3. Finally, data points are assigned to the closest offspring centroid.



3(a): Parents p_1 and p_2 3(b): Assignment 3(c): Random selection 3(d): New solution

Figure 3: Crossover based on centroids matching: (a) two parent solutions; (b) the assignment between centroids of p_1 and p_2; (c) random selection of matched centroids and (d) the produced offspring

3.4.3 Mutation

The use of mutation is typically motivated by the possible permanent loss of genetic material during the execution of a GA. It is possible that after several generations performing the selection followed by crossover, the same value is applied to a specific genetic material (fragment) in all solutions of the population. If this happens, selection and crossover will not be able to restore the lost genetic material, leading the algorithm to a premature convergence (29). In this context, mutation plays the role of recovering the lost genetic materials by modifying one or more portions of a solution. If crossover is supposed to combine existing solutions to find better ones, mutation is supposed to explore different regions of the search space (1), being useful on maintaining the genetic diversity from one generation of a GA population to the next.

In the proposed method, the mutation of a solution is performed by a biased relocation of a centroid, which is done by randomly selecting a centroid and relocating it to a position of a data point, where data points far from their centroids are more likely to be chosen. Then the respective re-assignments are performed. The proposed mutation can be described as follows:

1. Randomly select a centroid c^* and remove it from the solution (Figure 4 (a)).

2. Re-assign each data point to the closest centroid among the $m - 1$ remaining centroids (Figure 4 (b)). In this step, the centroids remain in their current positions.

3. Randomly select a data point x_u and re-insert c^* in the position of x_u (Figure 4 (c)). This position is selected as in a roulette wheel, such that data points far from their current centroids are more likely to be chosen.

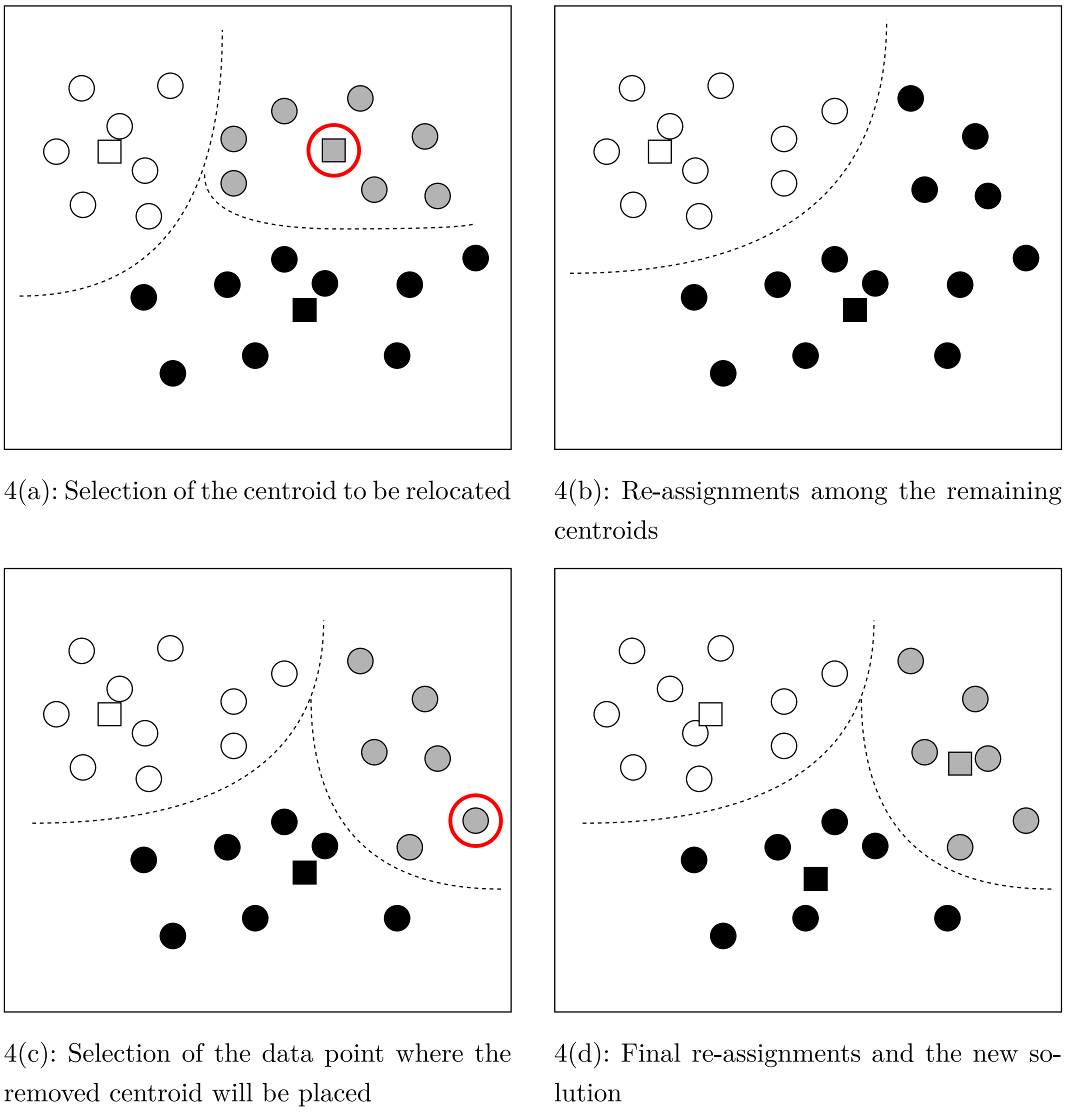
   Consider that $C(x_j)$ is the centroid of the cluster where a data point x_j is assigned to. Let $P(x_j)$ be the probability that x_j is selected as the new centroid.

   $$P(x_j) = \frac{d(x_j, C(x_j))}{\sum_{i=1}^{n} d(x_i, C(x_i))} \tag{3-1}$$

   As we can easily observe, data points far from their centroids are more likely to be chosen, as their probabilities are higher. Thus, we select the data point x_u to be the new centroid by randomly choosing a value according to the probabilities distribution, coming back to a solution with m centroids.

4. Among the m resulting centroids, re-assign each data point to the closest centroid and update the position of centroids (Figure 4 (d)).

This relocation proved to be effective as it promotes the introduction of a large move in the position of a centroid, allowing substantial changes in the structure of the overall solution that are not achieved by the local improvement. Thus, the mutation, in addition to generating a solution that is similar to the offspring – which is typically a good solution due to elitism in parental selection – also contributes significantly to the population diversity through the biased relocation of a centroid.


4(a): Selection of the centroid to be relocated

4(b): Re-assignments among the remaining centroids

4(c): Selection of the data point where the removed centroid will be placed

4(d): Final re-assignments and the new solution

Figure 4: Mutation based on centroid relocation

3.4.4 Local improvement

After the generation process in crossover and mutation, θ' is enhanced by means of a local improvement procedure. The local improvement aims to find a local optimum by applying local changes to the current solution. Here, the adopted local improvement is one run of the K-means algorithm. The K-means starts with the initial solution θ' with m centroids c_1, c_2, ..., c_m, and proceeds by alternating between two steps:

1. Assignment step: Assign each data point x_i to the closest cluster.

$$cluster(x_i) = min_j \quad d(x_i, c_j), \quad j = 1, ..., m \tag{3-2}$$

2. Update step: Calculate the new centroids c_j to be the mean (average) point of the data points in the new clusters.

$$c_j = \frac{\sum_{x \in S_j} x}{|S_j|}, \quad j = 1, ..., m \tag{3-3}$$

Then the algorithm keeps repeating these two steps until the assignments no longer change, converging to a local optimum.

In our experiments, we use the K-means implementation of Hamerly (13), which gives the same answer of the standard Lloyd's K-means (24) but is much faster in practice. This implementation avoids distance computations by using the triangle inequality and lower bounds on distances. The time per K-means iteration of the algorithm is O(nmd+md^2), where n is the number of data points, m is the number of clusters and d is the number of data dimensions (features). However, the calculated lower bounds allow to eliminate the innermost K-means loop in around 80% of the time, which in practise is much faster than the standard K-means.

3.4.5 Treatment of incomplete solutions

When generating new individuals, it is possible that a cluster is left empty, i.e., no data point is allocated to it. This can happen in two situations:

– when generating a solution in crossover or mutation, one or more centroids could not be the closest centroid to any data point;

– when generating a solution in the population initialization, it could happen that some numbers in the interval [1, ..., m] are not chosen for the assignment, as it is a random selection.

To work around this issue and deal only with solutions with exactly m clusters, the following procedure is adopted just after the execution of crossover, mutation and population initialization if a solution with less than m clusters is generated: while there is an empty cluster, select the data point farthest from its current centroid (that belongs to a cluster with more than one element) and allocate it to an empty cluster.

3.5 Population management

One of the main challenges in population-based algorithms is avoiding premature convergence of the population. The selection of parents based on elitism favors good individuals, reducing the diversity of the genetic material in the coming generations of the population. To overcome this issue, we propose a mechanism that propagates good solutions while ensuring diversity. Thus, the search procedure can be led to unexplored regions of the search space without losing promising individuals. We call *Survivors selection* as this mechanism to complement the selection, crossover and local improvement operators.

3.5.1 Survivors selection

The *Survivors selection* aims to select the best individuals to propagate when the maximum population size Π is reached. This procedure determines the μ individuals that will go on to the next generation, by discarding λ individuals ($\lambda = \Pi - \mu$) that are either clones (identical to other solution) or bad regarding the fitness, according to Algorithm 2.

The detection of clones is done by a hash table that uses the following hash function:

$$H(s) = \sum_{i=1}^{m} i \cdot g(i) \tag{3-4}$$

where s is a solution for the MSSC, and $g(i)$ denotes the cardinality of the i-th least populated cluster of s. To detect if a solution s' with a hash function $H(s')$ and ordered cardinalities $g'(i)$ is a clone of a given solution s, the following conditions must hold:

i) $H(s) = H(s')$

ii) $g(i) = g'(i), \quad \forall i = 1...m$

iii) $f(s') - \epsilon < f(s) < f(s') + \epsilon$

where $f(s)$ is the objective function value (cost) of a solution s and ϵ is a small error. Therefore, if the three conditions above are satisfied, then there is a collision in the hash table and the clone is identified.

This characteristic of eliminating clones and poor solutions reveals two aspects of the *Survivors selection* mechanism. The first one is that population diversity is maintained, as we favor the removal of clones first. The second is related to elitism, as the individuals with good fitness are preserved.

Algorithm 2 Survivors selection

```
1: for $i = 1...\lambda$ do
2:     $X \leftarrow$ all individuals having a clone
3:     if $X \neq \emptyset$ then
4:         Remove $p \in X$ with minimum fitness
5:     else
6:         Remove $p$ in the population with minimum fitness
7:     end if
8: end for
```

3.6 Computational complexity

As stated in previous sections, after the initialization of the population, the proposed meta-heuristic operates in an external loop that calls some internal operators. Among these operators, the main bottleneck is the local improvement phase, where the K-means is performed to improve the newly mutated solution. As the computational complexity of K-means is $O(nmd + md^2)$ (13), the overall complexity of the proposed algorithm is $O(I_{max}(nmd + md^2))$, as the local improvement is applied after the mutation in each iteration (the maximum number of iterations in the external loop is I_{max}). Thus, although the proposed meta-heuristic is of the same asymptotic complexity of K-means, it is in practice slower than K-means due to the constant I_{max} set in the external loop and to other internal operators, like crossover, mutation, survivors selection and diversification. In the following, the computational complexity of each internal operator:

- Selection: $O(w)$ (w constant)
- Crossover: $O(nmd + m^3)$
- Mutation: $O(nmd)$
- Local improvement (K-means): $O(nmd + md^2)$

In addition to the above internal operators, a mechanism to select the best individuals for propagation is also performed in the general loop. However, unlike the other operators, the *Survivors selection* is not performed on each iteration, but only when the population reaches Π individuals. Therefore, its contribution to the computational time of the algorithm is small in practice, being $O(n\frac{I_{max}}{\Pi})$.

4 Computational Experiments and Analysis

In this chapter, we discuss the computational experiments and the analysis that emerge from them. From now on, all results regarding the proposed algorithm will be identified as HG-means (for Hybrid Genetic K-Means). The results are presented in terms of objective function value and time, and are compared to current state-of-the-art results in the MSSC literature.

The experiments were conducted on an Intel Core i5 2.6 GHz processor machine with 8 GB of RAM memory. The source codes were written in C++, using the UNIX g++ compiler on a Linux Ubuntu 14.04 LTS 64-bit operating system.

Three different analysis are presented in this chapter. The first one analyses the general performance of the proposed algorithm in terms of objective function value and computational time, where we compare our results with the current literature. The second one analyses the impact of instances characteristics, where the objective is to verify if the proposed algorithm is affected by the number of clusters and the size of instances. Finally, we analyse the contribution of crossover and mutation components in the performance of the method.

In the coming result tables, the following notation is used:

- n is the size of the instance (number of data points);
- m is the number of clusters;
- d is the number of features;
- f_{best} is the best known value for the MSSC objective found so far;
- E is the error from the best known solution, calculated as:

$$E = \tfrac{f - f_{best}}{f_{best}} \times 100$$

  where f is the value of the MSSC objective found by an algorithm. For HG-means, E_{med} and E_{avg} report the error from the median and average values, respectively;
- t is the CPU time in seconds;

4.1 Instances

The selection of instances was done in order to cover different types of real data, considering both the instance size and the number of features as important indicators to describe the nature of data. Table 1 summarizes the characteristics of the adopted instances, which correspond to 24 important benchmarks in recent clustering literature, as reported in (26) and (6). For all considered instances, every feature value is a real or integer number and there is no missing values. The number of data points (instances size) ranges from 59 (smallest) to 434,874 (largest); and the number of features ranges from 2 (smallest) to 128 (largest).

In order to elucidate the coming analysis, we propose the discrimination of these instances according to their sizes, that from now on will by identified as Group A1, Group A2, Group B and Group C of instances.

Groups A1 and A2 correspond to small instances reported in the work of (26). The difference between them is that A1 is composed by really small instances with up to 150 data points, whereas instances in A2 have some hundreds of data points. This distinction is useful to choose a reasonable number of clusters to be tested in each group. For instances A1, tests are performed considering up to 10 clusters, whereas in A2 the number of clusters is extended. Group B of instances correspond to medium/large instances reported also by (26), comprising data where the number of points ranges from 1,060 to 20,000. Group C of instances are the same reported in (6), where instance sizes ranges from 13,910 to 434,874 data points. This last group is especially relevant as it consider large real world instances, which allows a deeper analysis in the scaling of algorithms.

	Instances	n	d
Group A1	German towns	59	2
	Bavaria postal 1	89	3
	Bavaria postal 2	89	4
	Fisher's Iris Plant	150	4
Group A2	Heart Disease	297	13
	Liver Disorders	345	6
	Ionosphere	351	34
	Congressional Voting	435	16
	Breast Cancer	683	9
	Pima Indians Diabetes	768	8
Group B	TSPLIB1060	1060	2
	Image Segmentation	2310	19
	TSPLIB3038	3038	2
	Page Blocks	5473	10
	Pendigit	10992	16
	Letters	20000	16
Group C	Gas sensor	13910	128
	EEG eye state	14980	14
	D15112	15112	2
	KEGG metabolic relation	53413	20
	Shuttle control	58000	9
	Pla85900	85900	2
	Skin Segmentation	245057	3
	3D road network	434874	3

Table 1: Instances description

4.2 Parameters calibration

As in most meta-heuristics, the values chosen for parameters directly affect the results. In this work, we consider five main parameters: w (the tournament size for selection), μ (the population size), Π (the maximum size of population), I_{max} (the maximum number of iterations) and I_S (the number of iterations without improvement that causes the algorithm stop). For the parameters related to the management of individuals in the population (w, μ and Π) we performed a calibration to choose the best configuration for the experiments. For the parameters related to the number of iterations the algorithm takes (I_{max} and I_S), we directly set their values to 4000 and 2500, respectively. These values were set in order to obtain results in a time comparable to previous authors.

In preliminary experiments, we started with the following configuration for the free variables, as they produced good and stable results after an initial and manually exploration: $w = 3$, $\mu = 80$ and $\Pi = 300$. From this baseline, we expanded the range of these values by an one-factor-at-a-time (OFAT) approach. Table 2 presents the ranges of tested values for each parameter and the final values achieved after increasing/decreasing each parameter value at a time.

Parameter		Range of values	Final value
w	Tournament size for selection	{2, 3, 4}	3
μ	Population size	{60, 70, 80, 90, 100}	80
Π	Maximum size of population	{200, 250, 300, 350, 400}	200

Table 2: Parameters calibration

We considered a subset of five instances for calibration, that were chosen based on their sizes and number of features (Liver disorders, Ionosphere, Breast cancer, Pima Indians diabetes and TSPLIB1060). Small instances were not considered because in almost all of them the different configurations for calibration achieved the best known result (possibly the global optimal), so they are not so informative. As we consider 3 parameters independently, the combination of parameters resulted in 11 possible configurations, as shown in Table 3. For this reason, large instances were also not considered. We chose medium-sized instances and tested 4 different number of clusters (20, 30, 40, 50) for each instance. After measuring the offset between the average error and the execution time for each configuration, we took the best one among the 11 options.

In Table 3, the first three columns report the values of parameters w, μ and Π, whereas the remaining columns report, in turn, the errors from the best, worst, median and average solution, and finally the average time taken to run the considered instances for calibration.

w	μ	Π	E_{bst}	E_{wst}	E_{med}	E_{avg}	t_{avg} (s)
3	80	300	-4.70	0.28	-0.88	-1.24	188.13
2	80	300	-4.83	0.27	-0.81	-1.25	200.58
4	80	300	-4.53	0.19	-0.90	-1.22	170.92
3	60	300	-4.86	0.34	-0.93	-1.21	177.14
3	70	300	-4.85	0.25	-0.95	-1.23	183.56
3	90	300	-4.77	0.61	-0.81	-1.23	191.54
3	100	300	-4.80	0.34	-0.73	-1.23	195.69
3	**80**	**200**	**-4.80**	**0.19**	**-0.88**	**-1.23**	**175.04**
3	80	250	-4.87	0.40	-0.78	-1.24	200.25
3	80	350	-4.85	0.41	-0.79	-1.25	203.47
3	80	400	-4.91	0.32	-0.76	-1.26	199.84

Table 3: Results of calibration

4.3 Experimental results

4.3.1 General performance and computational time

The results of numerical experiments regarding solution performance and computational time for groups A1, A2, B and C of instances are presented in Tables 4, 5, 6 and 7, respectively. Due to the stochastic nature of the proposed meta-heuristic, the results of HG-means correspond to the average of 10 runs, where a run is the execution of an instance with a specific m.

For comparison purposes, we analyse the results considering the best known solutions found so far and the results of global K-means (GKM) (23), the modified global K-means (MGKM) (5), the multi-start modified global K-means (MS-MGKM) (26) and the difference of convex clustering (DCClust and MS-DCA) (6) algorithms, which are recent works in MSSC literature, corresponding to the state-of-the-art for this problem. For Tables 4, 5, 6, the computational time of MS-MGKM, MGKM and GKM algorithms correspond to the measurements performed by an Intel Core 2 Dual 2.50 GHz processor machine with 3 GB of RAM memory, as reported in (26). For Table 7, an Intel Core i5 2.90 GHz processor machine with 8 GB of RAM was used to measure DCClust and MS-DCA algorithms, according to (6).

Tables 4 and 5 presents the results for instances of group A1 and A2. As we can observe, HG-means finds the new best solution or achieves the current best one in all instances for most of m values. In some cases, as in Ionosphere instance, HG-means results reach more than 4% of the error, a significant improvement. On the other hand, the proposed algorithm requires more computational effort than MS-MGKM, MGKM and GKM algorithms for most of these small instances – in many cases the computational time of HG-means is similar to MS-MGKM. It is also important to note that in some instances like Bavaria postal 2, Liver disorders, Ionosphere and Pima Indians diabetes, the gap of HG-means results to the best known solutions increases for large m.

Table 6 presents the results for instances of group B. As in results for small instances, HG-means finds the new best solution or achieves the current best one in all instances for most of m values, outperforming MS-MGKM, MGKM and GKM algorithms in terms of solution quality. Regarding computational time, HG-means is faster than MS-MGKM and GKM in TSPLIB3038, faster than MS-MGKM in TSPLIB1060 and has nearly the same time of MS-MGKM and MGKM in Pendigit. For Image segmentation and Page Block – the latter, only for large m – HG-means requires more computational effort.

Table 7 presents the results for instances of group C – the group with the largest instances. As in the previous results, HG-means finds the new best solution or achieves the current best one in all instances for most of m values. Regarding computational time, HG-means is faster than DCClust for almost all instances, except for EEG eye state and D15112.

m	f_{best}	MS-MGKM		MGKM		GKM		HG-means		
		E	t	E	t	E	t	E_{med}	E_{avg}	t
German towns										
2	121430	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07
3	77009	0.00	0.02	1.45	0.00	1.45	0.00	0.00	0.00	0.07
4	49601	0.24	0.02	0.72	0.00	0.72	0.00	0.00	0.00	0.08
5	38716	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.08
6	30536	0.00	0.02	0.27	0.00	0.00	0.00	0.00	0.00	0.08
7	24433	0.08	0.02	0.00	0.00	0.09	0.00	0.00	0.00	0.09
8	21631	0.00	0.02	0.54	0.00	0.64	0.00	**-0.68**	**-0.68**	0.10
9	18550	2.13	0.02	4.46	0.00	2.13	0.00	0.00	0.00	0.10
10	16307	1.81	0.02	1.52	0.00	1.81	0.00	0.01	0.01	0.11
Bavaria postals 1										
2	6.0255E+11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07
3	2.9451E+11	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.08
4	1.0447E+11	0.05	0.00	0.05	0.00	0.05	0.00	0.00	0.00	0.09
5	5.9762E+10	0.06	0.00	0.54	0.00	0.54	0.00	0.00	0.00	0.10
6	3.5909E+10	0.07	0.00	1.44	0.00	1.44	0.00	0.00	0.00	0.11
7	2.1983E+10	0.01	0.00	3.17	0.02	3.17	0.00	0.00	0.00	0.13
8	1.3385E+10	0.25	0.02	1.71	0.02	1.71	0.00	0.00	0.00	0.13
9	8.4237E+9	0.28	0.02	2.85	0.02	2.85	0.00	0.00	0.00	0.14
10	6.4465E+9	0.07	0.02	3.55	0.02	3.55	0.00	0.00	0.00	0.15
Bavaria postals 2										
2	4.8631E+10	0.00	0.00	0.00	0.00	7.75	0.00	0.00	0.00	0.08
3	1.7399E+10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09
4	7.5591E+9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10
5	5.3429E+9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12
6	3.1876E+9	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.12
7	2.2159E+9	0.61	0.00	1.50	0.02	1.50	0.00	**-0.04**	**-0.04**	0.14
8	1.7045E+9	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.16
9	1.4030E+9	0.00	0.02	0.00	0.02	0.00	0.00	**-0.14**	**-0.14**	0.18
10	1.1841E+9	0.00	0.02	0.00	0.02	0.00	0.00	**-0.26**	**-0.26**	0.20
Fisher Iris										
2	152.348	0.00	0.02	7.32	0.00	7.32	0.00	0.00	0.00	0.11
3	78.851	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.18
4	57.228	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.20
5	46.446	0.00	0.03	1.86	0.02	1.86	0.02	0.00	0.00	0.21
6	39.040	0.00	0.03	1.21	0.02	1.21	0.02	0.00	0.00	0.28
7	34.298	0.00	0.05	0.51	0.02	0.51	0.02	0.00	0.00	0.28
8	29.989	0.18	0.06	0.73	0.02	0.73	0.02	0.00	0.00	0.31
9	27.786	1.07	0.08	0.00	0.02	0.00	0.02	0.00	0.00	0.38
10	25.834	0.00	0.09	0.73	0.02	0.73	0.02	0.00	0.00	0.44

Table 4: Objective value and computational time of algorithms (Instances A1)

m	f_{best}	MS-MGKM		MGKM		GKM		HG-means		
		E	t	E	t	E	t	E_{med}	E_{avg}	t
Heart disease										
2	598900	0.00	0.05	93.96	0.02	93.96	0.02	0.00	0.00	0.48
5	327970	0.01	0.27	0.08	0.05	0.08	0.02	0.00	0.00	1.01
10	200650	0.00	0.94	6.96	0.09	6.93	0.03	**-0.06**	**-0.06**	1.91
15	147650	0.00	1.47	0.06	0.17	1.69	0.06	**-0.55**	**-0.55**	2.45
20	117780	0.00	1.94	0.77	0.23	1.07	0.09	**-0.86**	**-0.86**	3.15
25	99292	0.00	2.47	1.74	0.33	1.98	0.11	**-0.82**	**-0.78**	3.66
30	86216	0.00	3.08	1.36	0.44	1.57	0.14	**-0.54**	**-0.48**	4.38
40	67701	0.00	4.05	1.05	0.69	4.68	0.20	**-1.69**	**-1.44**	5.15
50	54878	0.00	5.20	3.33	1.03	9.01	0.27	**-1.06**	**-1.01**	6.18
Liver disorders										
2	423980	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.27
5	218260	0.06	0.13	0.18	0.03	0.07	0.03	0.00	0.00	0.65
10	119400	0.00	0.63	0.64	0.08	2.38	0.05	6.71	6.71	1.46
15	97405	0.00	1.20	0.81	0.13	5.16	0.08	**-0.67**	**-0.67**	1.84
20	81192	0.00	1.77	1.52	0.19	7.33	0.11	**-1.41**	**-1.40**	2.72
25	69212	0.00	2.61	0.68	0.28	7.49	0.13	**-1.90**	**-1.74**	3.30
30	60325	0.00	3.28	0.37	0.39	7.77	0.16	**-2.25**	**-2.11**	3.66
40	47336	0.87	4.67	0.00	0.66	7.82	0.23	**-1.13**	**-1.04**	4.55
50	38305	0.28	6.06	0.00	0.97	6.63	0.28	**-1.55**	**-1.50**	5.20
Ionosphere										
2	2419.4	0.00	0.08	0.00	0.05	0.00	0.03	0.00	0.00	1.12
5	1891.5	0.00	0.39	1.86	0.13	2.28	0.05	**-0.09**	**-0.09**	2.76
10	1559.4	0.00	0.81	0.13	0.27	0.11	0.08	**-0.60**	**-0.58**	3.97
15	1390.1	0.00	1.39	0.92	0.42	0.88	0.11	**-2.18**	**-2.16**	5.54
20	1252.4	0.00	1.86	1.17	0.77	2.99	0.13	**-2.75**	**-2.69**	7.04
25	1140.8	0.00	2.42	0.31	1.39	4.45	0.16	**-3.34**	**-3.28**	8.38
30	1043.0	0.00	3.09	1.28	2.20	4.75	0.20	**-4.44**	**-4.39**	9.88
40	856.6	0.00	4.98	2.36	4.77	6.14	0.30	**-3.58**	**-3.46**	13.11
50	702.6	0.00	6.89	3.68	8.39	8.05	0.38	**-3.72**	**-3.51**	16.79
Congressional vote										
2	1640.9	0.00	0.11	0.00	0.03	0.00	0.02	0.00	0.00	0.72
5	1335.8	0.03	0.56	0.14	0.11	0.14	0.05	0.00	0.00	2.28
10	1123.3	0.00	1.50	1.86	0.20	1.86	0.08	**-0.32**	**-0.31**	4.28
15	992.07	0.00	2.25	0.36	0.31	0.33	0.13	**-0.84**	**-0.75**	5.31
20	906.47	0.00	3.28	0.71	0.44	0.34	0.17	**-1.51**	**-1.47**	5.62
25	839.75	0.00	4.36	0.92	0.58	0.37	0.22	**-2.28**	**-2.22**	6.57
30	776.77	0.00	5.61	0.98	0.73	2.83	0.27	**-1.45**	**-1.48**	6.93
40	689.35	0.00	7.75	1.81	1.16	1.28	0.38	**-2.88**	**-2.84**	8.48
50	617.5	0.00	10.19	1.73	1.84	1.98	0.48	**-3.39**	**-3.32**	10.18

Breast cancer										
2	19323	0.00	0.09	0.00	0.06	0.00	0.05	0.00	0.00	0.58
5	13705	0.00	0.58	0.52	0.17	0.52	0.09	0.00	0.00	1.64
10	10205	0.00	1.06	2.70	0.33	0.75	0.17	**-0.14**	**-0.14**	3.02
15	8704.7	0.00	1.72	0.72	0.48	0.04	0.23	**-0.80**	**-0.74**	4.74
20	7695.2	0.39	2.48	1.34	0.66	0.00	0.31	**-1.19**	**-1.13**	6.79
25	6946.4	0.00	3.38	2.86	0.83	3.35	0.38	**-0.88**	**-0.81**	7.55
30	6360.3	0.00	4.27	3.31	0.98	2.99	0.45	**-0.40**	**-0.32**	8.79
40	5487.8	0.00	5.59	1.39	1.39	3.13	0.61	**-0.62**	**-0.64**	9.87
50	4812.3	0.00	6.75	1.85	1.83	3.95	0.77	**-1.21**	**-1.21**	11.60
Pima Indians diabetes										
2	5.1424E+6	0.00	0.13	0.12	0.09	0.12	0.06	0.00	0.00	0.73
5	1.7369E+6	0.00	0.38	1.02	0.22	1.02	0.13	0.00	0.00	1.45
10	9.3046E+5	0.00	0.91	0.70	0.41	2.04	0.20	**-0.01**	**-0.01**	3.47
15	6.9499E+5	0.00	1.73	1.87	0.59	1.69	0.30	**-0.04**	**-0.04**	6.43
20	5.7241E+5	0.00	2.98	0.88	0.80	2.29	0.39	**-0.01**	**-0.01**	8.53
25	4.8869E+5	0.00	4.97	1.26	0.98	3.31	0.52	**-0.08**	**-0.08**	10.16
30	4.3219E+5	0.00	8.75	0.69	1.22	3.44	0.59	**-0.29**	**-0.28**	11.92
40	3.5656E+5	0.00	12.61	0.69	1.70	4.03	0.83	**-0.39**	**-0.40**	13.38
50	3.0903E+5	0.00	15.17	1.14	2.28	5.53	1.06	**-0.14**	**-0.12**	17.91

Table 5: Objective value and computational time of algorithms (Instances A2)

m	f_{best}	MS-MGKM		MGKM		GKM		HG-means		
		E	t	E	t	E	t	E_{med}	E_{avg}	t
TSPLIB1060										
2	9.8319E+9	0.00	0.06	0.00	0.08	0.00	0.08	0.00	0.00	0.54
10	1.7548E+9	0.22	1.14	0.05	0.34	0.23	0.36	0.00	0.00	2.96
20	7.9179E+8	0.14	3.19	1.88	0.66	1.88	0.69	0.00	0.00	4.52
30	4.8125E+8	0.24	5.39	3.37	0.97	3.34	1.03	0.00	0.00	5.11
40	3.4342E+8	0.00	8.28	2.82	1.30	4.00	1.38	**-0.60**	**-0.58**	6.84
50	2.5551E+8	1.16	10.20	2.53	1.69	3.10	1.73	0.11	0.11	7.03
60	1.9960E+8	0.00	13.95	2.42	2.06	3.16	2.08	**-1.06**	**-0.98**	9.12
80	1.2967E+8	0.00	18.17	4.44	2.89	4.38	2.80	**-0.57**	**-0.44**	11.82
100	9.7019E+7	0.00	23.61	3.50	3.75	3.60	3.53	**-0.16**	**-0.13**	15.08
Image segmentation										
2	3.5606E+7	0.00	0.52	0.00	1.39	0.00	1.06	0.00	0.00	5.27
10	9.7952E+6	0.64	6.00	1.76	6.75	1.76	3.97	0.00	0.00	20.41
20	5.1283E+6	0.77	14.41	1.49	13.11	0.09	7.58	0.00	0.00	37.22
30	3.5074E+6	0.00	27.98	0.07	20.89	0.07	11.36	**-0.01**	**-0.01**	45.18
40	2.7398E+6	0.18	46.63	1.24	28.92	1.25	16.67	0.10	0.07	62.52
50	2.2249E+6	0.55	61.17	2.41	37.72	2.41	18.73	**-0.02**	**-0.01**	76.21
60	1.8818E+6	0.00	77.19	2.34	46.91	1.47	22.50	**-0.39**	**-0.42**	92.88
80	1.4207E+6	0.00	97.61	1.64	68.81	2.59	30.19	**-0.35**	**-0.32**	120.04
100	1.1419E+6	0.00	122.92	0.81	93.69	1.74	38.00	**-0.66**	**-0.63**	154.15

TSPLIB3038										
2	3.1688E+9	0.00	0.56	0.00	0.86	0.00	1.38	0.00	0.00	1.72
10	5.6025E+8	0.57	4.92	0.58	3.30	2.78	8.41	0.00	0.00	5.75
20	2.6681E+8	0.20	11.89	0.48	5.77	2.00	16.63	0.00	0.00	12.31
30	1.7557E+8	0.39	29.17	0.67	8.25	1.45	25.00	**-0.02**	**-0.02**	16.02
40	1.2548E+8	0.33	43.59	1.35	10.70	1.35	33.23	**-0.41**	**-0.41**	19.49
50	9.8400E+7	0.57	61.75	1.41	13.23	1.19	41.52	**-0.08**	**-0.09**	25.92
60	8.1180E+7	0.00	85.42	2.01	15.75	1.02	49.75	**-0.78**	**-0.77**	30.85
80	6.0642E+7	0.00	147.11	1.58	20.94	0.95	66.42	**-0.25**	**-0.22**	45.54
100	4.8182E+7	0.00	211.00	1.52	26.11	2.11	83.16	**-0.86**	**-0.82**	56.92
Page blocks										
2	5.7937E+10	0.00	1.88	0.00	6.92	0.24	8.19	0.00	0.00	3.84
10	4.5662E+9	0.00	12.08	0.00	34.09	0.80	49.62	**-0.73**	**-0.73**	10.46
20	1.6742E+9	0.00	20.75	2.57	62.09	2.37	92.30	**-0.26**	**-0.25**	30.50
30	9.3523E+8	0.00	29.83	0.62	89.42	1.38	132.41	**-1.55**	**-1.55**	71.93
40	6.2570E+8	0.67	37.14	0.00	118.55	0.17	172.13	**-2.49**	**-2.41**	117.52
50	4.2024E+8	0.00	43.42	2.17	149.77	2.21	212.27	**-0.93**	**-0.92**	168.89
60	3.0850E+8	0.00	57.25	1.42	184.06	1.09	254.88	0.00	0.00	233.99
80	2.0436E+8	0.00	90.89	0.69	258.69	2.16	334.36	**-1.33**	**-1.30**	427.42
100	1.4428E+8	0.00	114.59	0.91	346.94	0.81	415.19	**-0.58**	**-0.56**	589.40
Pendigit										
2	1.2812E+8	0.00	6.94	0.00	16.73	0.39	9.42	0.00	0.00	25.58
10	4.9302E+7	0.00	58.58	0.00	137.17	0.00	81.94	0.00	0.00	71.49
20	3.4123E+7	0.22	144.13	0.39	281.33	0.22	168.36	**-0.30**	**-0.30**	182.63
30	2.7157E+7	0.13	254.30	0.00	425.77	0.00	255.52	**-0.25**	**-0.25**	305.88
40	2.3446E+7	0.00	387.47	0.00	570.50	0.11	343.11	**-0.03**	**-0.03**	490.22
50	2.1090E+7	0.00	523.19	0.20	713.95	0.56	430.71	**-0.18**	**-0.18**	671.02
60	1.9357E+7	0.00	678.02	0.36	858.09	0.22	518.97	**-0.21**	**-0.20**	823.97
80	1.6961E+7	0.00	1021.52	0.11	1148.34	0.25	696.94	**-0.48**	**-0.48**	1163.43
100	1.5258E+7	0.00	1456.25	0.35	1438.78	0.45	876.94	**-0.18**	**-0.18**	1591.32
Letter										
2	1.3819E+6	0.00	14.48	0.00	72.89	0.00	40.17	0.00	0.00	67.89
10	8.5752E+5	0.00	180.42	0.00	583.36	0.00	363.78	0.00	0.00	423.15
20	6.7263E+5	0.00	463.38	0.53	1202.36	0.72	732.23	**-0.01**	**-0.01**	819.44
30	5.7977E+5	0.08	741.50	0.00	1811.45	0.08	1094.59	**-0.19**	**-0.19**	1145.33
40	5.1925E+5	0.00	1057.08	0.00	2457.58	0.78	1453.70	**-0.15**	**-0.15**	1582.46
50	4.7727E+5	0.00	1384.72	0.23	3065.69	0.29	1812.06	**-0.42**	**-0.43**	1880.03
60	4.4166E+5	0.00	1781.64	0.12	3678.17	0.37	2173.91	**-0.24**	**-0.24**	2124.35
80	3.9129E+5	0.00	2594.13	0.25	4907.64	0.64	2892.00	**-0.25**	**-0.25**	2626.61
100	3.5644E+5	0.00	3465.92	0.04	6130.53	0.29	3611.05	**-0.47**	**-0.46**	3230.49

Table 6: Objective value and computational time of algorithms (Instances B)

m	f_{best}	MS-MGKM	GKM	MS-DCA	DCClust		HG-means		
		E	E	E	E	t	E_{med}	E_{avg}	t
Gas sensor									
2	7.9119E+14	0.00	0.00	0.00	0.00	42.73	**-90.00**	**-90.00**	121.37
3	5.0241E+13	0.00	0.00	0.00	0.00	111.29	0.00	0.00	151.02
5	3.2273E+13	0.00	0.00	0.00	0.00	286.45	**-0.10**	**-0.10**	275.17
10	1.6552E+13	0.00	0.00	0.00	0.00	951.37	**-0.18**	**-0.18**	641.13
12	1.4066E+13	0.01	0.00	0.01	0.00	1279.65	**-0.20**	**-0.20**	784.63
15	1.1380E+13	0.00	0.36	0.00	0.35	1847.30	**-0.94**	**-0.94**	790.37
20	8.7916E+12	0.00	0.62	0.16	0.62	2810.51	**-0.21**	**-0.21**	1396.77
25	7.2348E+12	0.00	0.16	0.00	0.47	3783.46	**-0.28**	**-0.28**	1588.62
EEG eye state									
2	8.1781E+11	0.00	0.00	0.00	0.00	0.62	**-4.07**	**-4.07**	13.32
3	1.8339E+11	0.00	0.00	0.00	0.00	1.23	0.00	0.00	13.92
5	1.3386E+8	0.00	0.00	0.00	0.00	4.34	0.00	0.00	20.89
10	4.5669E+7	0.00	0.00	0.00	0.00	40.39	**-0.80**	**-0.80**	171.96
12	4.0251E+7	0.00	0.00	0.00	0.01	60.65	**-1.43**	**-1.43**	269.36
15	3.4653E+7	0.05	0.00	0.05	0.26	105.16	0.00	0.00	360.76
20	2.8987E+7	0.00	0.00	0.03	0.96	187.72	**-0.01**	**-0.01**	647.22
25	2.5995E+7	0.12	0.63	0.12	0.00	285.23	**-0.18**	**-0.18**	891.46
D15112									
2	3.6840E+11	0.00	0.00	0.00	0.00	2.53	0.00	0.00	10.52
3	2.5324E+11	0.00	0.00	0.00	0.00	5.02	0.00	0.00	18.82
5	1.3271E+11	0.00	0.00	0.00	0.00	8.53	0.00	0.00	15.95
10	6.4892E+10	0.00	0.78	0.00	0.00	19.56	**-0.62**	**-0.62**	45.31
12	5.4500E+10	0.01	0.02	0.92	0.00	27.47	0.00	0.00	45.13
15	4.3138E+10	0.00	0.02	0.00	0.24	39.47	0.00	0.00	70.13
20	3.2177E+10	0.25	0.01	0.25	0.00	65.38	0.00	0.00	74.85
25	2.5309E+10	0.00	0.49	0.00	0.00	98.69	**-0.04**	**-0.04**	123.21
KEGG metabolic									
2	1.1385E+9	0.00	0.00	0.00	0.00	21.40	0.00	0.00	76.26
3	4.9006E+8	0.00	0.00	0.00	0.00	56.68	0.00	0.00	73.83
5	1.8837E+8	0.00	0.00	0.00	0.00	216.12	0.00	0.00	111.56
10	6.3515E+7	0.00	0.00	0.00	0.09	1049.09	**-4.73**	**-4.73**	244.11
12	4.7825E+7	0.00	0.00	0.41	0.49	1432.95	**-0.95**	**-0.95**	356.62
15	3.5484E+7	3.28	3.28	0.00	0.01	2148.77	**-1.54**	**-1.54**	487.20
20	2.5430E+7	0.00	1.26	0.29	0.47	3254.46	**-2.83**	**-2.83**	976.88
25	1.9289E+7	0.51	0.67	0.00	0.37	4544.48	**-1.03**	**-1.02**	1389.41

Shuttle control									
2	2.1343E+9	0.00	0.00	0.00	0.00	2.85	0.00	0.00	42.26
3	1.0854E+9	0.00	0.00	0.00	0.00	9.14	0.00	0.00	46.07
5	7.2448E+8	0.00	0.00	0.01	0.28	58.19	0.00	0.00	59.82
10	2.8322E+8	0.32	0.00	0.32	0.20	224.92	**-0.02**	**-0.02**	126.55
12	2.1414E+8	0.44	0.00	0.45	4.04	442.96	0.00	0.00	161.36
15	1.5315E+8	0.00	0.00	0.01	0.72	760.13	0.00	0.00	238.19
20	1.0601E+8	0.00	0.00	0.02	0.58	1405.66	**-3.67**	**-3.67**	529.78
25	7.8727E+7	1.50	1.50	1.54	0.00	2164.50	**-3.60**	**-3.60**	880.90
Pla85900									
2	3.7491E+15	0.00	0.00	0.00	0.00	83.51	0.00	0.00	72.39
3	2.2806E+15	0.00	0.00	0.00	0.00	159.76	0.00	0.00	101.53
5	1.3397E+15	0.00	0.00	0.00	0.00	318.26	0.00	0.00	119.61
10	6.8294E+14	0.00	0.00	0.00	0.00	731.32	0.00	0.00	290.93
12	5.7504E+14	0.19	0.00	0.19	0.00	911.50	0.00	0.00	311.70
15	4.6249E+14	0.00	0.03	0.00	0.00	1211.55	**-0.47**	**-0.47**	457.37
20	3.4988E+14	0.00	0.29	0.00	0.52	1715.45	**-0.02**	**-0.02**	614.96
25	2.8265E+14	0.11	0.94	0.11	0.00	2293.84	**-0.15**	**-0.15**	819.37
Skin segmentation									
2	1.3224E+9	0.00	0.00	0.00	0.00	705.36	0.00	0.00	143.78
3	8.9362E+8	0.00	0.00	0.00	0.00	1320.25	0.00	0.00	211.71
5	5.0203E+8	0.00	0.00	0.00	0.00	2401.20	0.00	0.00	218.41
10	2.5122E+8	0.00	0.00	0.00	0.00	5109.63	0.00	0.00	321.14
12	2.1416E+8	0.55	0.00	0.55	0.00	6090.59	**-0.41**	**-0.41**	447.09
15	1.6964E+8	0.18	0.00	0.19	0.18	7733.34	**-1.63**	**-1.63**	481.07
20	1.2770E+8	0.17	0.00	0.17	0.14	10309.53	**-2.09**	**-2.09**	677.60
25	1.0299E+8	0.00	0.00	0.00	0.00	12959.91	**-1.07**	**-1.07**	935.14
3D road network									
2	4.9133E+7	0.00	0.00	0.00	0.00	1672.56	0.00	0.00	383.45
3	2.2778E+7	0.00	0.00	0.00	0.00	3811.46	0.00	0.00	544.78
5	8.8257E+6	0.00	0.00	0.01	0.00	7701.69	0.00	0.00	708.47
10	2.5671E+6	-	-	0.21	0.00	17506.87	**-0.02**	**-0.02**	3658.75
12	1.8498E+6	-	-	0.05	0.00	21534.91	**-0.07**	**-0.07**	4196.57
15	1.2707E+6	-	-	0.26	0.00	27696.03	0.00	0.00	5122.38
20	8.0872E+5	-	-	0.73	0.00	38245.56	**-0.01**	**-0.01**	6878.53
25	6.0334E+5	-	-	0.00	1.93	49311.45	**-1.78**	**-1.78**	10768.587

Table 7: Objective value and computational time of algorithms (Instances C)

4.3.2 Impact of the number of clusters

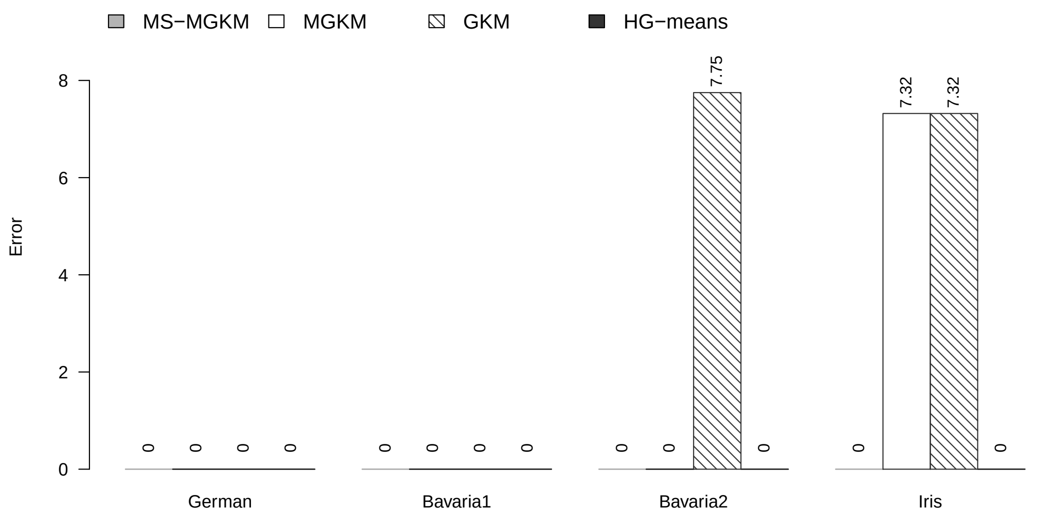
From the performance observed in the results of HG-means algorithm in both solution quality and computational time, we sought to verify if the algorithm works well by increasing the number of clusters. This analysis is very important to confirm the robustness of the method, since we are dealing with a larger combinatorial problem when m is large.

Table 8 presents how HG-means performs for different numbers of clusters. For each group of instances, it reports the average error (E_{avg}) to the best known solution when varying the number of clusters (m). For instances A1, A2 and B, HG-means has the most significant improvements – compared to the best known solution – in cases where m is large. For instances of group C, HG-means increased the gap to the best known solution when m = (2, 20, 25), confirming the robustness of the method.

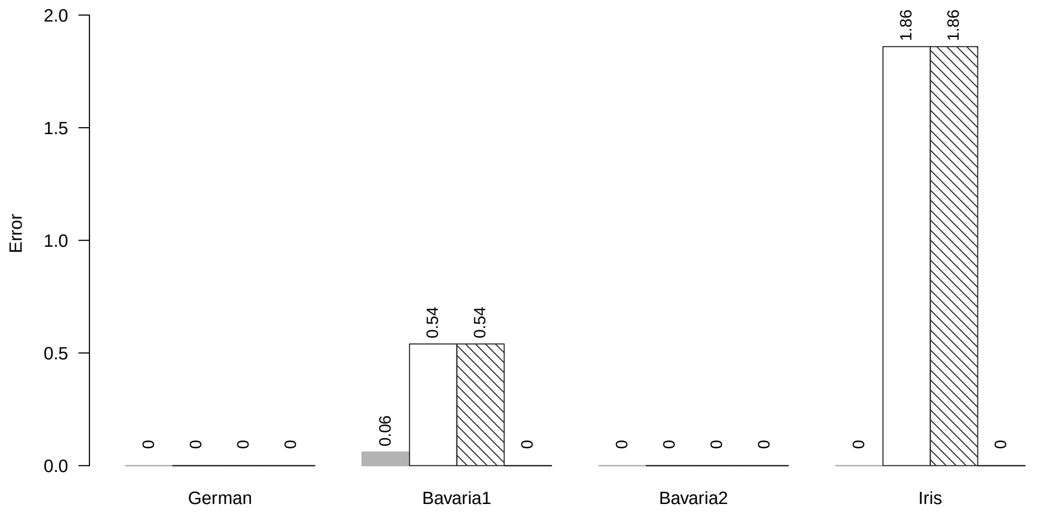
Instances A1									
m	2	3	4	5	6	7	8	9	10
E_{avg}	0.00	0.00	0.00	0.00	0.00	-0.01	-0.17	-0.03	-0.06
Instances A2									
m	2	5	10	15	20	25	30	40	50
E_{avg}	0.00	-0.02	0.94	-0.82	-1.26	-1.48	-1.51	-1.64	-1.78
Instances B									
m	2	10	20	30	40	50	60	80	100
E_{avg}	0.00	-0.12	-0.09	-0.33	-0.59	-0.25	-0.43	-0.50	-0.46
Instances C									
m	2	3	5	10	12	15	20	25	
E_{avg}	-11.76	0.00	-0.01	-0.79	-0.38	-0.57	-1.11	-1.02	

Table 8: HG-means average performance for different numbers of clusters

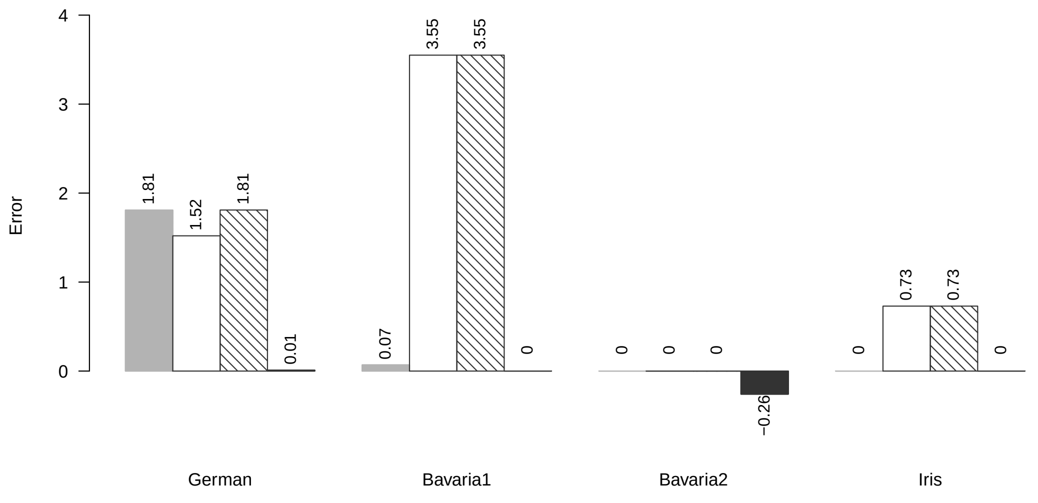
Figures 5 to 8 display the percentage gap of the methods when considering different number of clusters. In these figures, results are shown for each instance, rather than the aggregated values of Table 8.



5(a): $m = 2$

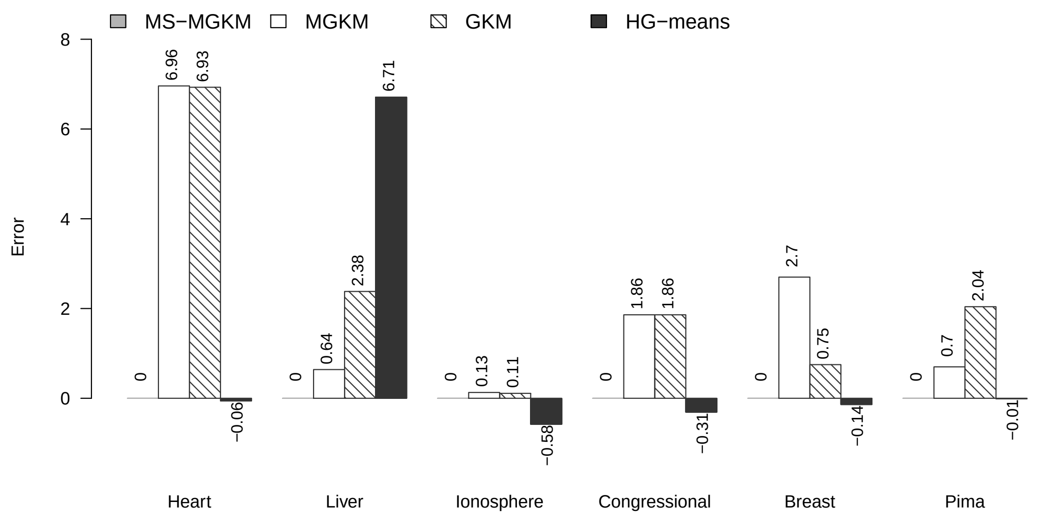


5(b): $m = 5$

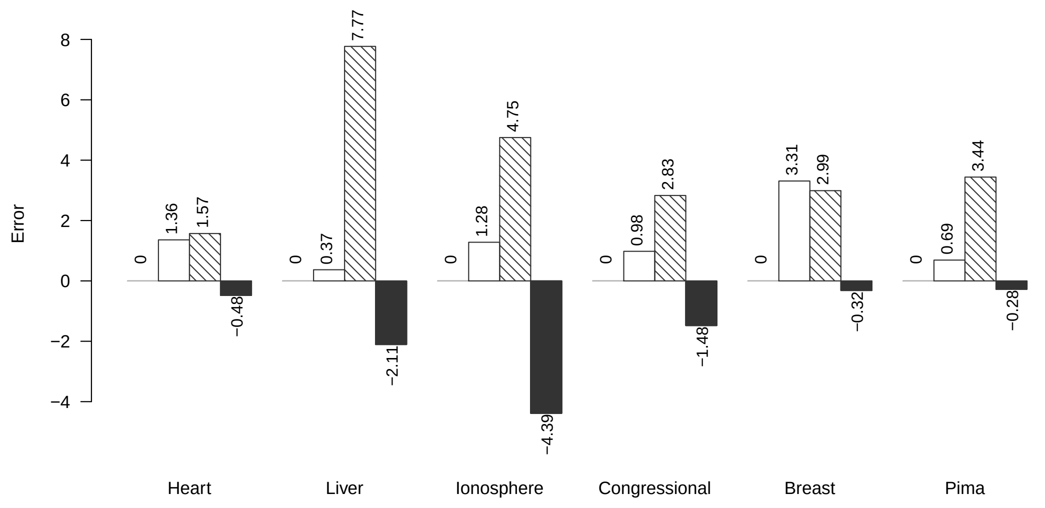


5(c): $m = 10$

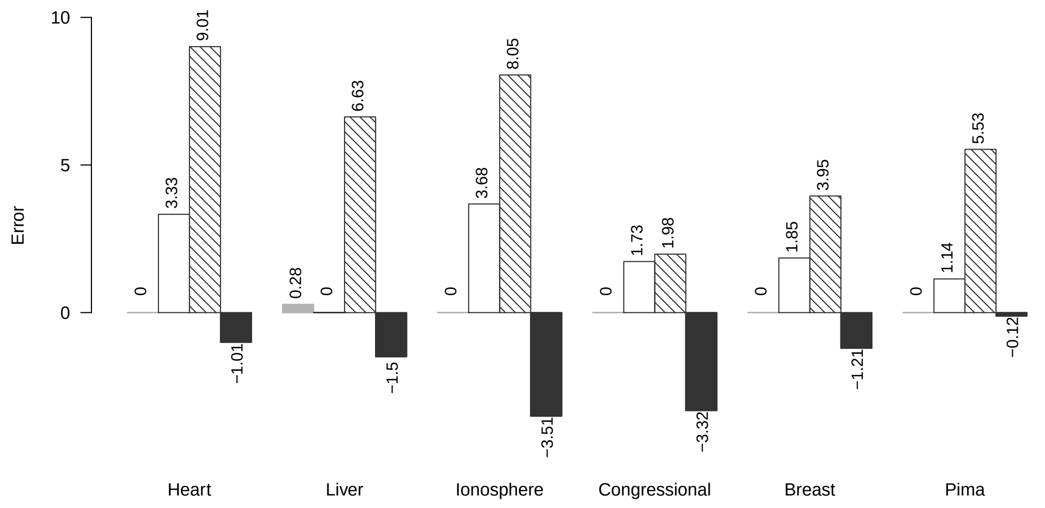
Figure 5: Algorithms performance for different number of clusters (Instances A1)



6(a): $m = 10$

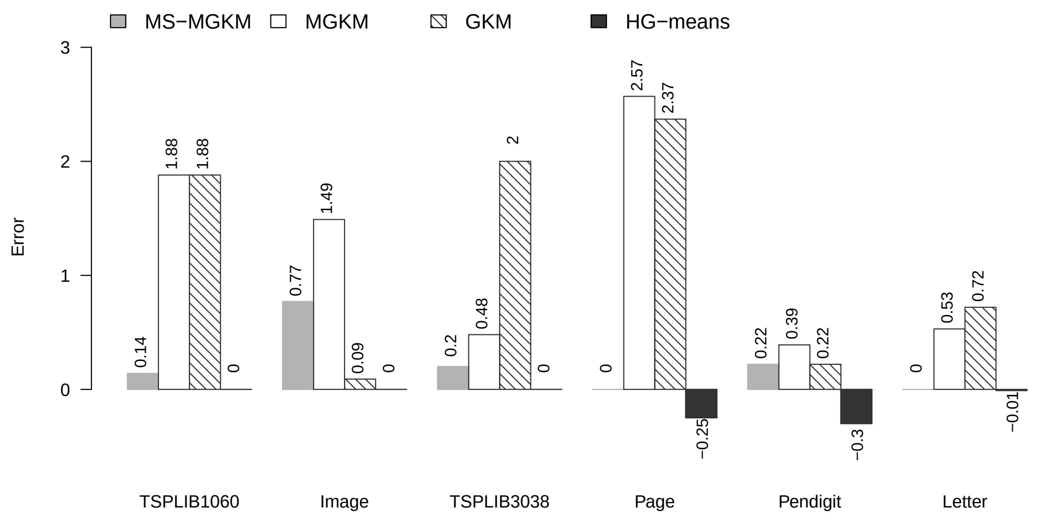


6(b): $m = 30$

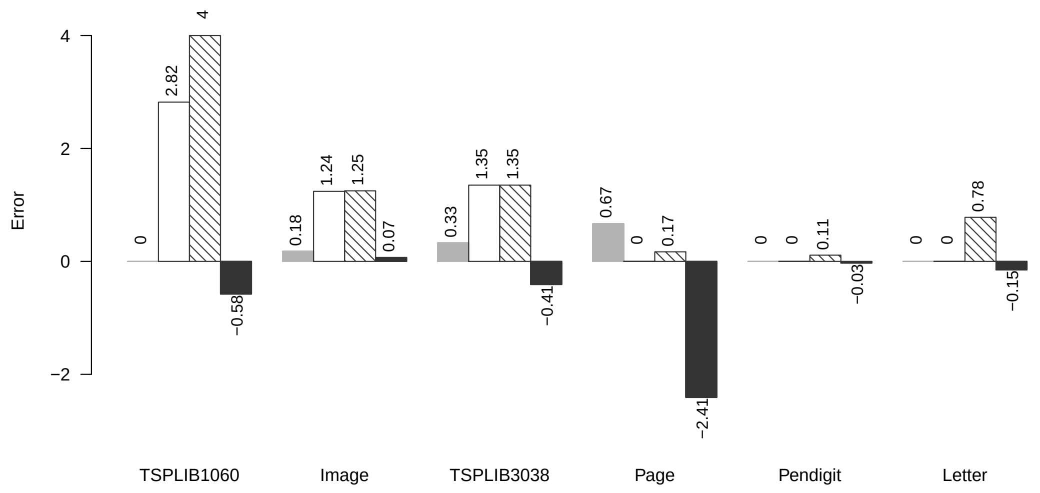


6(c): $m = 50$

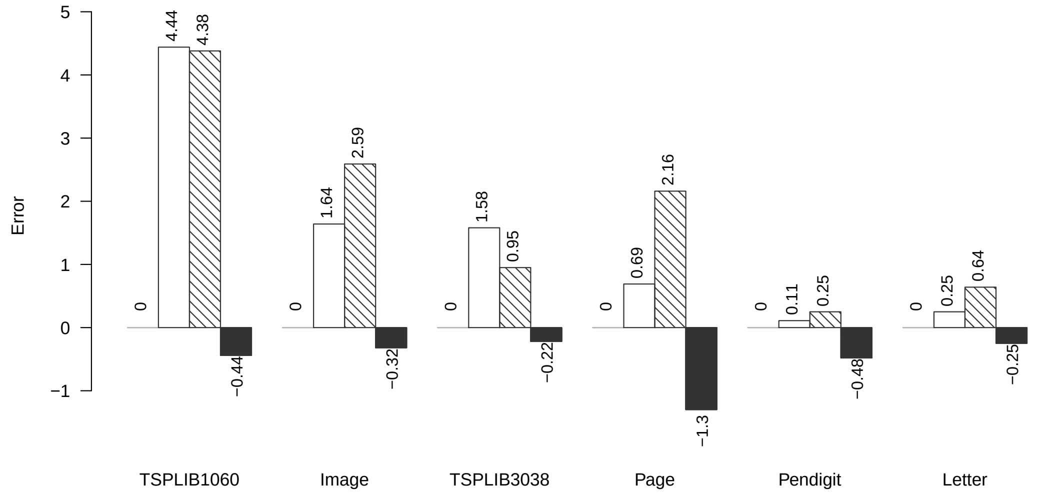
Figure 6: Algorithms performance for different number of clusters (Instances A2)



7(a): $m = 20$

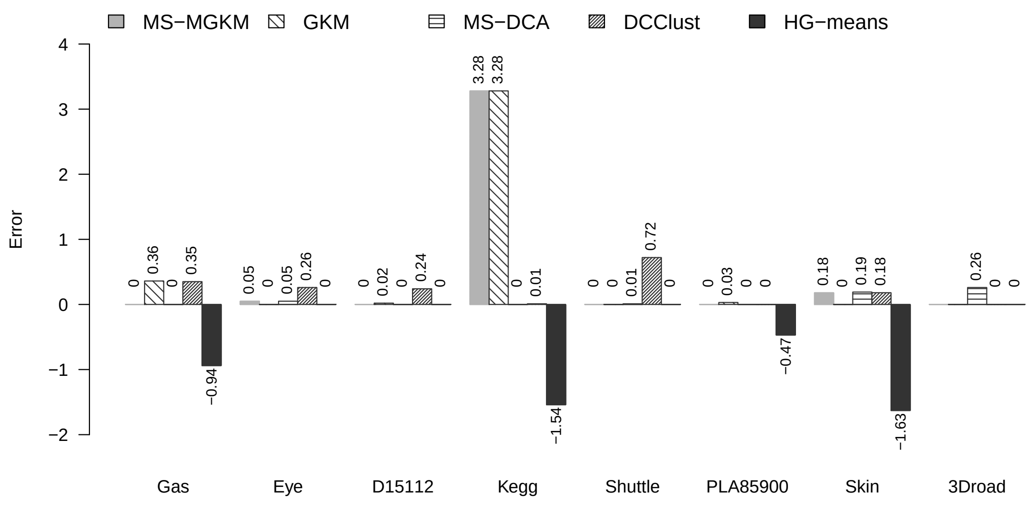


7(b): $m = 40$

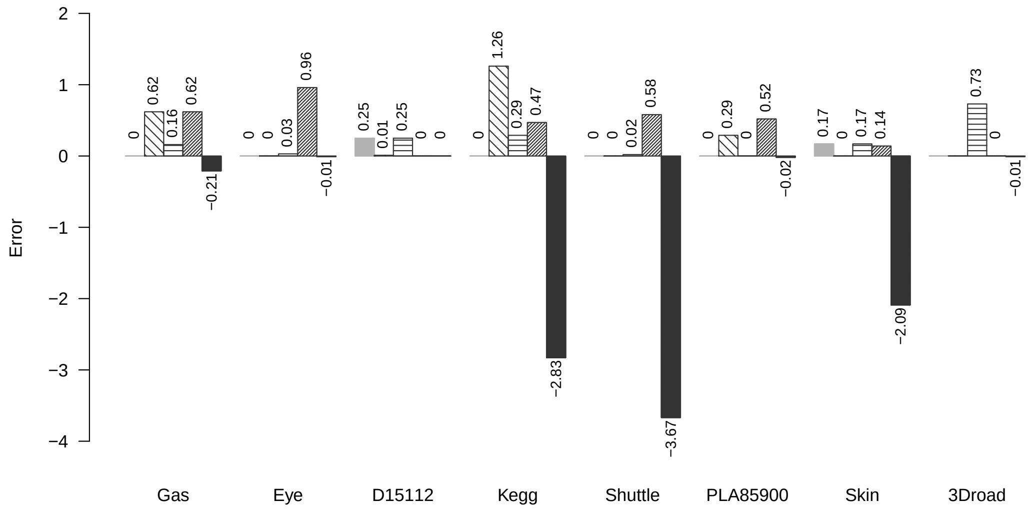


7(c): $m = 80$

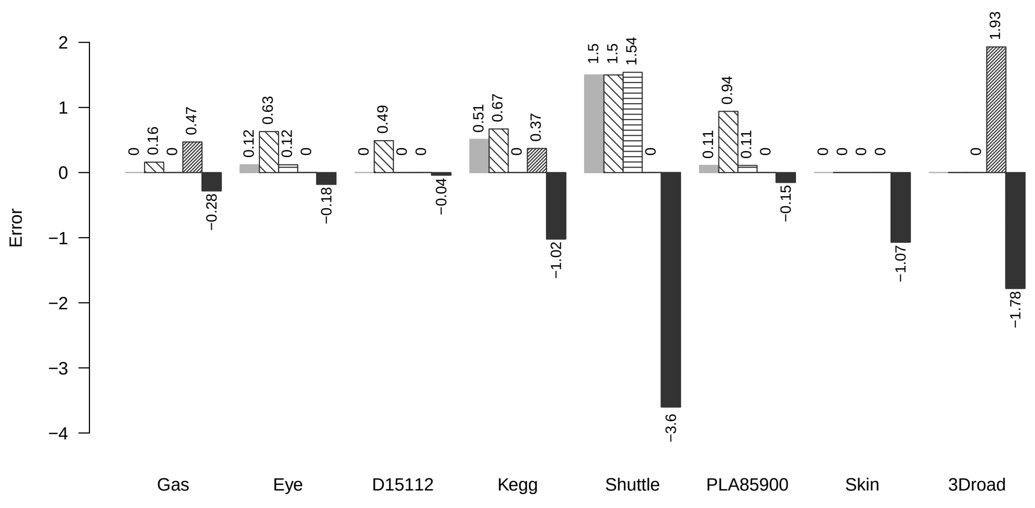
Figure 7: Algorithms performance for different number of clusters (Instances B)



8(a): $m = 15$



8(b): $m = 20$



8(c): $m = 25$

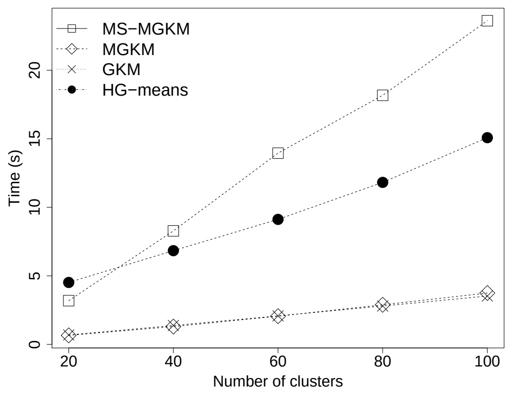
Figure 8: Algorithms performance for different number of clusters (Instances C)

4.3.3 Impact of instances size

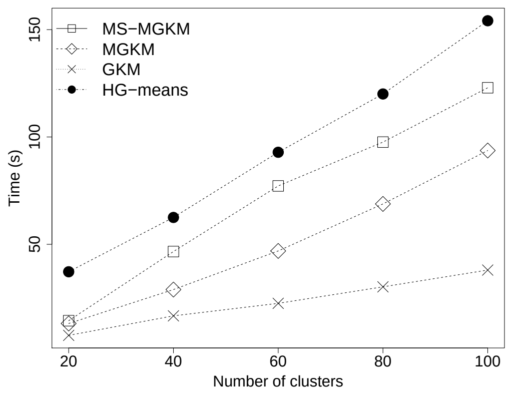
Section 4.3.1 shows that the objective value of solutions produced by HG-means outperformed the compared algorithms in most experiments. This section presents some analysis regarding the HG-means scaling, i.e., how much computational time does the algorithm require in large data. Figures 9 and 10 compare HG-means with MS-MGKM, MGKM, GKM and DCClust in terms of CPU time as a function of the number of clusters m. The elapsed time was measured in seconds and the results are presented for groups B and C of instances. For small instances, there is a very small difference in the computational time of the considered algorithms. Therefore, the results for groups A1 and A2 are not presented in this analysis.

For group B of instances, HG-means takes a similar amount of time when compared to other algorithms, and has its most competitive performance when m is increased. For instances of group C, HG-means is faster than the other algorithms exactly in the largest instances (Pla85900, Skin segmentation and 3D road), having a similar computational time for the remaining instances in this group.

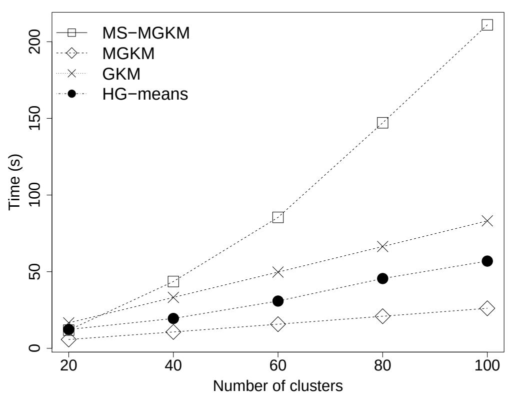
Therefore, HG-means is very promising regarding scalability, as it produces better solutions and requires less computational time than the compared algorithms on large instances. In addition, there is room to adjust the number of iterations the algorithm takes and to use more efficient data structures that can improve runtime.



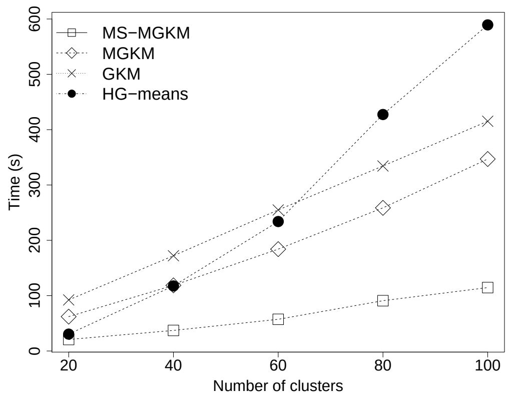
9(a): TSPLIB1060 (n = 1060)



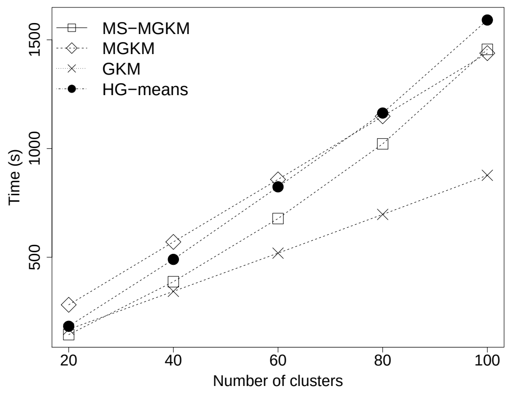
9(b): Image segmentation (n = 2310)



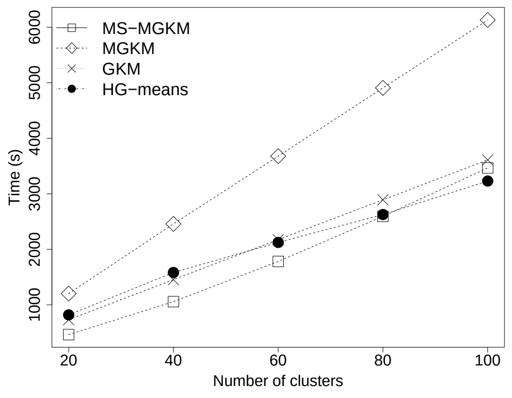
9(c): TSPLIB3038 (n = 3038)



9(d): Page blocks (n = 5473)

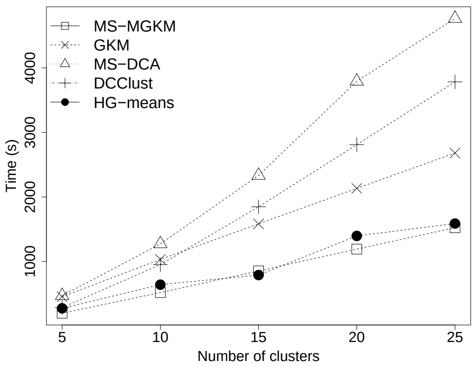


9(e): Pendigit (n = 10992)

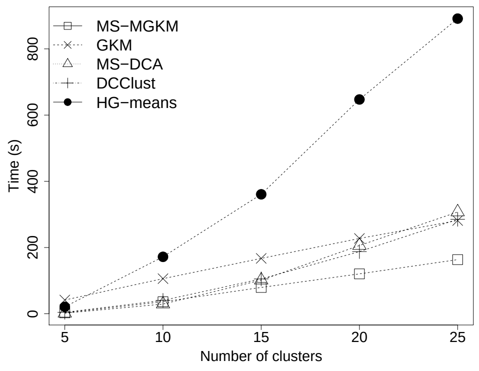


9(f): Letter (n = 20000)

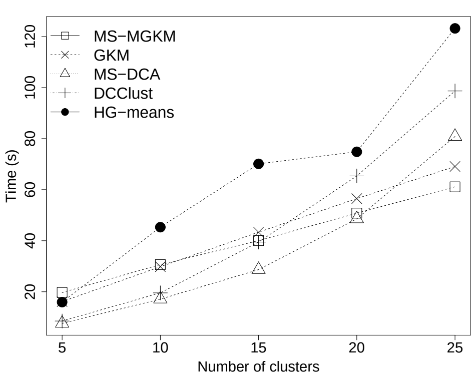
Figure 9: The CPU time of algorithms (Instances B)



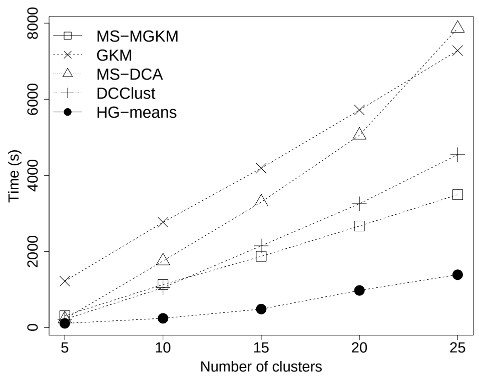
10(a): Gas sensor ($n = 13910$)



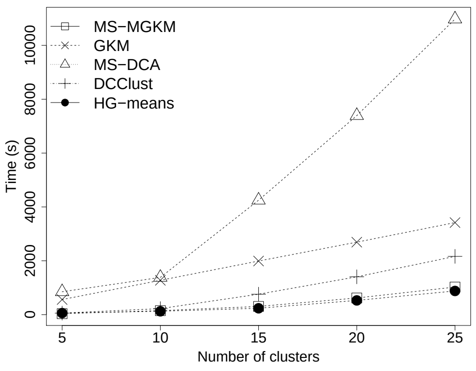
10(b): EEG eye state ($n = 14980$)



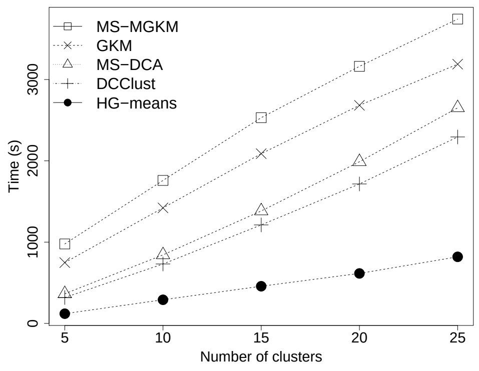
10(c): D15112 ($n = 15112$)



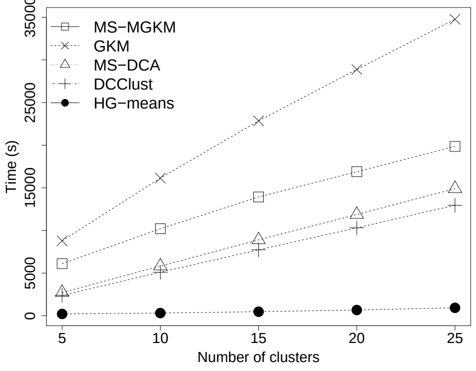
10(d): KEGG metabolic ($n = 53413$)



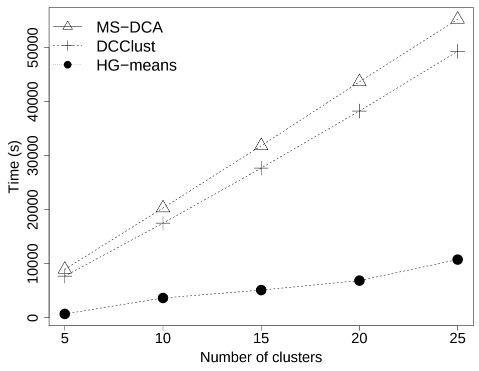
10(e): Shuttle control ($n = 58000$)



10(f): Pla85900 ($n = 85900$)



10(g): Skin segmentation ($n = 245057$)



10(h): 3D road network ($n = 434874$)

Figure 10: The CPU time of algorithms (Instances C)

4.3.4 Contribution of components

In order to understand the contribution of components in the performance of HG-means, we performed some experiments by removing two important operators at a time: crossover and mutation. For comparison purposes, Table 9 reports the average error with respect to the best known solution in three scenarios: (i) complete HG-means with all components (E_{avg}), (ii) HG-means without mutation (E_{avg} -M) and (iii) HG-means without crossover (E_{avg} -C).

Instances A1									
m	2	3	4	5	6	7	8	9	10
E_{avg}	0.00	0.00	0.00	0.00	0.00	-0.01	-0.17	-0.03	-0.06
E_{avg} -M	1.94	32.57	117.64	118.07	158.28	34.66	54.85	80.79	95.18
E_{avg} -C	0.00	0.00	0.00	0.00	0.00	-0.01	-0.17	-0.03	-0.06
Instances A2									
m	2	5	10	15	20	25	30	40	50
E_{avg}	0.00	-0.02	0.94	-0.82	-1.26	-1.48	-1.51	-1.64	-1.78
E_{avg} -M	0.00	0.45	3.67	3.75	5.20	6.21	6.03	4.80	3.93
E_{avg} -C	0.00	-0.02	0.95	-0.61	-0.85	-0.82	-0.59	-0.26	0.26
Instances B									
m	2	10	20	30	40	50	60	80	100
E_{avg}	0.00	-0.18	-0.06	-0.39	-0.83	-0.23	-0.54	-0.57	-0.54
E_{avg} -M	0.06	53.51	125.25	47.97	115.16	39.70	36.84	44.20	51.91
E_{avg} -C	0.00	-0.18	-0.06	-0.39	-0.78	-0.08	-0.24	0.13	0.59

Table 9: HG-means performance in different scenarios: with all components, without mutation and without crossover

Mutation and crossover are both essential to find high-quality solutions. For the case where we have HG-means without mutation, it does not find (in average) a solution better than the best known so far, while in the two other scenarios (complete HG-means and HG-means without crossover), it has a significantly better performance. Additionally, the complete HG-means algorithm performs better than the HG-means without crossover, especially when m increases, indicating that the crossover is an essential operator when dealing with more complex clustering tasks.

5
Conclusions and Future work

In this work, we have addressed the clustering task as an optimization problem, where the Minimum sum-of-squares (MSSC) formulation is investigated. Among the many existing formulations of clustering problems, the MSSC in the Euclidean space is the most treated one.

In order to solve the Euclidean MSSC problem, we proposed a hybrid genetic algorithm (HG-means) that combines the K-means local improvement with diversification strategies. The proposed method allows to efficiently escape from local minima and reach high quality solutions, outperforming the best current literature results for all considered sets of benchmark instances in terms of solution quality. In terms of computational time, the proposed method outperformed some of state-of-the-art algorithms in large instances.

The effectiveness of the proposed method resides in the crossover, mutation and local improvement components, that contribute with elitism and diversification, favoring the propagation of good solutions and making possible the exploration of new regions of the search space. As the proposed meta-heuristic is based on a general genetic framework – with classical components as crossover and mutation – the algorithm is easy to understand and implement.

For future work, we will investigate some efficient data structures and speed up techniques to improve the computational time of HG-means. We also aim to solve different objectives in data clustering with the proposed method. As we tackle the clustering problem from an optimization perspective through a meta-heuristic, we can elaborate a neighbourhood structure that is general enough to deal with different models, where the evaluation cost of a solution varies according to the objective. Thus, with this algorithmic framework we can easily extend the scope of this research to verify the suitability of different clustering models when considering different types of data.

Bibliography

1 ABDOUN, O.; ABOUCHABAKA, J. ; TAJANI, C.. **Analyzing the performance of mutation operators to solve the travelling salesman problem**. arXiv preprint arXiv:1203.3099, 2012.

2 ALOISE, D.; DESHPANDE, A.; HANSEN, P. ; POPAT, P.. **NP-hardness of euclidean sum-of-squares clustering**. Machine learning, 75(2):245–248, 2009.

3 ALOISE, D.; HANSEN, P.. **A branch-and-cut SDP-based algorithm for minimum sum-of-squares clustering**. Pesquisa Operacional, 29(3):503–516, 2009.

4 BAGIROV, A. M.; YEARWOOD, J.. **A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems**. European Journal of Operational Research, 170(2):578–596, 2006.

5 BAGIROV, A. M.. **Modified global k-means algorithm for minimum sum-of-squares clustering problems**. Pattern Recognition, 41(10):3192–3199, 2008.

6 BAGIROV, A. M.; TAHERI, S. ; UGON, J.. **Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems**. Pattern Recognition, 53:12–24, 2016.

7 BURKE, E. K.; GUSTAFSON, S. ; KENDALL, G.. **Diversity in genetic programming: An analysis of measures and correlation with fitness**. Trans. Evol. Comp, 8(1):47–62, 2004.

8 DAS, S.; ABRAHAM, A. ; KONAR, A.. **Metaheuristic Clustering (Studies in Computational Intelligence)**. Springer, 2009.

9 DIAZ-GOMEZ, P. A.; HOUGEN, D. F.. **Initial population for genetic algorithms: A metric approach**. In: Proceedings of the 2007 International Conference on Genetic and Evolutionary Methods, GEM 2007, June 25-28, 2007, Las Vegas, Nevada, USA, p. 43–49, 2007.

10 FESTA, P.. **A biased random-key genetic algorithm for data clustering**. Mathematical Biosciences, 245(1):76–85, 2013.

11 FRALEY, C.; RAFTERY, A. E.. **Model-based clustering, discriminant analysis, and density estimation**. Journal of the American statistical Association, 97(458):611–631, 2002.

12 GUHA, S.; RASTOGI, R. ; SHIM, K.. **Cure: An efficient clustering algorithm for large databases**. SIGMOD Rec., 27(2):73–84, 1998.

13 HAMERLY, G.. **Making k-means even faster**. In: SDM, p. 130–140. SIAM, 2010.

14 HANSEN, P.; JAUMARD, B.. **Cluster analysis and mathematical programming**. Mathematical programming, 79(1-3):191–215, 1997.

15 HANSEN, P.; MLADENOVIĆ, N.. **J-means: a new local search heuristic for minimum sum of squares clustering**. Pattern Recognition, 34(2):405–413, 2001.

16 HANSEN, P.; ALOISE, D.. **A survey on exact methods for minimum sum-of-squares clustering**, 2009.

17 INABA, M.; KATOH, N. ; IMAI, H.. **Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering**. In: Proceedings of the tenth annual symposium on Computational geometry, p. 332–339. ACM, 1994.

18 JAIN, A. K.; MURTY, M. N. ; FLYNN, P. J.. **Data clustering: A review**. ACM Comput. Surv., 31(3):264–323, 1999.

19 KARKKAINEN, T.; AYRAMO, S.. **Introduction to partitioning-based clustering methods with a robust example**. 2006.

20 KLEINBERG, J.; TARDOS, E.. **Algorithm design**. Pearson/Addison Wesley, Boston (Mass.), London, Sydney, 2006.

21 KRISHNA, K.; MURTY, M. N.. **Genetic k-means algorithm**. IEEE Trans. Syst., Man, Cybern. B, 29(3):433–439, 1999.

22 KUHN, H. W.. **The Hungarian method for the assignment problem**. Naval Research Logistics Quarterly, 2:83 – 97, 1955.

23 LIKAS, A.; VLASSIS, N. ; VERBEEK, J. J.. **The global k-means clustering algorithm**. Pattern recognition, 36(2):451–461, 2003.

24 LLOYD, S.. **Least squares quantization in PCM**. IEEE Transactions on Information Theory, 28(2):129 – 137, 1982.

25 LU, Y.; LU, S.; FOTOUHI, F.; DENG, Y. ; BROWN, S. J.. **FGKA**. In: Proceedings of the 2004 ACM symposium on Applied computing - SAC'04. Association for Computing Machinery (ACM), 2004.

26 ORDIN, B.; BAGIROV, A. M.. **A heuristic algorithm for solving the minimum sum-of-squares clustering problems**. Journal of Global Optimization, 61(2):341–361, 2014.

27 SPÄTH, H.. **Cluster analysis algorithms for data reduction and classification of objects**. 1980.

28 VINOD, H. D.. **Integer programming and the theory of grouping**. Journal of the American Statistical Association, 64(326):506–519, 1969.

29 WHITLEY, D.. **A genetic algorithm tutorial**. Statistics and Computing, 4(2):65–85, 1994.

30 XAVIER, A. E.; XAVIER, V. L.. **Solving the minimum sum-of-squares clustering problem by hyperbolic smoothing and partition into boundary and gravitational regions**. Pattern Recognition, 44(1):70 – 77, 2011.

31 XU, R.; WUNSCHII, D.. **Survey of clustering algorithms**. IEEE Trans. Neural Netw., 16(3):645–678, 2005.

32 ZHANG, T.; RAMAKRISHNAN, R. ; LIVNY, M.. **Birch: An efficient data clustering method for very large databases**, 1996.

33 ZITZLER, E.; DEB, K. ; THIELE, L.. **Comparison of multiobjective evolutionary algorithms: Empirical results**. Evol. Comput., 8(2):173–195, 2000.