Pattern Recognition 88 (2019) 569–583

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

HG-means: A scalable hybrid genetic algorithm for minimum sum-of-squares clustering

Daniel Gribel, Thibaut Vidal*

Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro - RJ, Rua Marquês de São Vicente, 225 - Gávea, 22451-900, Brazil

ARTICLE INFO

Article history:
Received 25 April 2018
Revised 30 November 2018
Accepted 16 December 2018
Available online 17 December 2018

Keywords:
Clustering
Minimum sum-of-squares
Global optimization
Hybrid genetic algorithm
K-means
Unsupervised learning

ABSTRACT

Minimum sum-of-squares clustering (MSSC) is a widely used clustering model, of which the popular K-means algorithm constitutes a local minimizer. It is well known that the solutions of K-means can be arbitrarily distant from the true MSSC global optimum, and dozens of alternative heuristics have been proposed for this problem. However, no other algorithm has been predominantly adopted in the literature. This may be related to differences of computational effort, or to the assumption that a near-optimal solution of the MSSC has only a marginal impact on clustering validity.

In this article, we dispute this belief. We introduce an efficient population-based metaheuristic that uses K-means as a local search in combination with problem-tailored crossover, mutation, and diversification operators. This algorithm can be interpreted as a multi-start K-means, in which the initial center positions are carefully sampled based on the search history. The approach is scalable and accurate, outperforming all recent state-of-the-art algorithms for MSSC in terms of solution quality, measured by the depth of local minima. This enhanced accuracy leads to clusters which are significantly closer to the ground truth than those of other algorithms, for overlapping Gaussian-mixture datasets with a large number of features. Therefore, improved global optimization methods appear to be essential to better exploit the MSSC model in high dimension.

1. Introduction

Broadly defined, clustering is the problem of organizing a collection of elements into coherent groups in such a way that similar elements are in the same cluster and different elements are in different clusters. Of the models and formulations for this problem, the Euclidean minimum sum-of-squares clustering (MSSC) is prominent in the literature. MSSC can be formulated as an optimization problem in which the objective is to minimize the sum-of-squares of the Euclidean distances of the samples to their cluster means. This problem has been extensively studied over the last 50 years, as highlighted by various surveys and books [see, e.g., 14,19,22].

The NP-hardness of MSSC [2] and the size of practical datasets explain why most MSSC algorithms are heuristics, designed to produce an approximate solution in a reasonable computational time. K-means [16] (also called Lloyd's algorithm [32]) and K-means++ [5] are two popular local search algorithms for MSSC that differ in the construction of their initial solutions. Their simplicity and low computational complexity explain their extensive use in practice. However, these methods have two significant disadvantages: (i) their solutions can be distant from the global optimum, especially in the presence of a large number of clusters and dimensions, and (ii) their performance is sensitive to the initial conditions of the search.

To circumvent these issues, a variety of heuristics and metaheuristics have been proposed with the aim of better escaping from shallow local minima (i.e., poor solutions in terms of the MSSC objective). Nearly all the classical metaheuristic frameworks have been applied, including simulated annealing, tabu search, variable neighborhood search, iterated local search, evolutionary algorithms [1,15,21,34,37,39], as well as more recent incremental methods and convex optimization techniques [4,7,8,23]. However, these sophisticated methods have not been predominantly used in machine learning applications. This may be explained by three main factors: (1) data size and computational time restrictions, (2) the limited availability of implementations, or (3) the belief that a near-optimal solution of the MSSC model has little impact on clustering validity.

* Corresponding author.
E-mail addresses: dgribel@inf.puc-rio.br (D. Gribel), vidalt@inf.puc-rio.br, thibaut.vidal@cirrelt.ca (T. Vidal).

https://doi.org/10.1016/j.patcog.2018.12.022
0031-3203/

To help remove these barriers, we introduce a simple and efficient hybrid genetic search for the MSSC called HG-MEANS, and conduct extensive computational analyses to measure the correlation between solution quality (in terms of the MSSC objective) and clustering performance (based on external measures). Our method combines the improvement capabilities of the K-MEANS algorithm with a problem-tailored crossover, an adaptive mutation scheme, and population-diversity management strategies. The overall method can be seen as a multi-start K-MEANS algorithm, in which the initial center positions are sampled by the genetic operators based on the search history. HG-MEANS' crossover uses a minimum-cost matching algorithm as a subroutine, with the aim of inheriting genetic material from both parents without excessive perturbation and creating *child* solutions that can be improved in a limited number of iterations. The adaptive mutation operator has been designed to help cover distant samples without being excessively attracted by outliers. Finally, the population is managed so as to prohibit *clones* and favor the discovery of diverse solutions, a feature that helps to avoid premature convergence toward low-quality local minima.

As demonstrated by our experiments on a variety of datasets, HG-MEANS produces MSSC solutions of significantly higher quality than those provided by previous algorithms. Its computational time is also lower than that of recent state-of-the-art optimization approaches, and it grows linearly with the number of samples and dimension. Moreover, when considering the reconstruction of a mixture of Gaussians, we observe that the standard repeated K-MEANS and K-MEANS++ approaches remain trapped in shallow local minima which can be very far from the ground truth, whereas HG-MEANS consistently attains better local optima and finds more accurate clusters. The performance gains are especially pronounced on datasets with a larger number of clusters and a feature space of higher dimension, in which more independent information is available, but also in which pairwise distances are known to become more uniform and less meaningful. Therefore, some key challenges associated to high-dimensional data clustering may be overcome by improving the optimization algorithms, before even considering a change of clustering model or paradigm.

The remainder of this article is structured as follows. Section 2 formally defines the MSSC and reviews the related literature. Section 3 describes the proposed HG-MEANS algorithm. Section 4 reports our computational experiments, and Section 5 provides some concluding remarks.

2. Problem statement

In a clustering problem, we are given a set $P = \{p_1, \ldots, p_n\}$ of n samples, where each sample p_i is represented as a point in $\mathbb{R}^d$ with coordinates $(p_i^1, \ldots, p_i^d)$, and we seek to partition P into m disjoint clusters $\mathcal{C} = (C_1, \ldots, C_m)$ so as to minimize a criterion $f(\mathcal{C})$. There is no universal objective suitable for all applications, but $f(\cdot)$ should generally promote homogeneity (similar samples should be in the same cluster) and separation (different samples should be in different clusters). MSSC corresponds to a specific choice of objective function, in which one aims to form the clusters and find a center position $y_k \in \mathbb{R}^d$ for each cluster, in such a way that the sum of the squared Euclidean distances of each point to the center of its associated cluster is minimized. This problem has been the focus of extensive research: there are many applications [22], and it is the natural problem for which K-MEANS finds a local minimum.

A compact mathematical formulation of MSSC is presented in Eqs. (1)–(4). For each sample and cluster, the binary variable x_{ik} takes the value 1 if sample i is assigned to cluster k, and 0 otherwise. The variables $y_k \in \mathbb{R}^d$ represent the positions of the centers.

$$\min \quad \sum_{i=1}^{n}\sum_{k=1}^{m} x_{ik}||p_i - y_k||^2 \tag{1}$$

$$\text{s.t.} \quad \sum_{k=1}^{m} x_{ik} = 1 \qquad i \in \{1, \ldots, n\} \tag{2}$$

$$x_{ik} \in \{0, 1\} \qquad i \in \{1, \ldots, n\}, k \in \{1, \ldots, m\} \tag{3}$$

$$y_k \in \mathbb{R}^d \qquad k \in \{1, \ldots, m\} \tag{4}$$

In the objective, $||\cdot||$ represents the Euclidean norm. Eq. (2) forces each sample to be associated with a cluster, and Eqs. (3)–(4) define the domains of the variables. Note that in this model, and in the remainder of this paper, we consider a fixed number of clusters m. Indeed, from the MSSC objective viewpoint, it is always beneficial to use the maximum number of available clusters. For some applications such as color quantization and data compression [38], the number of clusters is known in advance (desired number of colors or compression factor). Analytical techniques have been developed to find a suitable number of clusters [42] when this information is not available. Finally, it is common to solve MSSC for a range of values of m and select the most relevant result a-posteriori.

Regarding computational complexity, MSSC can be solved in $\mathcal{O}(n^3)$ time when $d = 1$ using dynamic programming. For general m and d, MSSC is NP-hard [2,3]. Optimal MSSC solutions are known to satisfy at least two necessary conditions:

Property 1. In any optimal MSSC solution, for each $k \in \{1, \ldots, m\}$, the position of the center y_k coincides with the centroid of the points belonging to C_k:

$$y_k = \frac{1}{|C_k|}\sum_{i \in C_k} p_i. \tag{5}$$

Property 2. In any optimal MSSC solution, for each $i \in \{1, \ldots, n\}$, the sample p_i is associated with the closest cluster $C_{k_{\text{MIN}}(i)}$ such that:

$$k_{\text{MIN}}(i) = arg\,min_{k=1}^{n}||p_i - y_k||^2. \tag{6}$$

These two properties are fundamental to understand the behavior of various MSSC algorithms. The K-MEANS algorithm, in particular, iteratively modifies an incumbent solution to satisfy first Property 1 and then Property 2, until both are satisfied simultaneously. Various studies have proposed more efficient data structures and speed-up techniques for this method. For example, Hamerly [13] provides an efficient K-MEANS algorithm that has a complexity of $\mathcal{O}(nmd + md^2)$ per iteration. This algorithm is faster in practice than its theoretical worst case, since it avoids many of the innermost loops of K-MEANS.

Other improvements of K-MEANS have focused on improving the choice of initial centers [41]. K-MEANS++ is one such method. This algorithm places the first center y_1 in the location of a random sample selected with uniform probability. Then, each subsequent center y_k is randomly placed in the location of a sample p_j, with a probability proportional to the distance of p_j to its closest center in $\{y_1, \ldots, y_{k-1}\}$. Finally, K-MEANS is executed from this starting point. With this rule, the expected solution quality is within a factor $8(\log m + 2)$ of the global optimum.

Numerous other solution techniques have been proposed for MSSC. These algorithms can generally be classified according to whether they are exact or heuristic, deterministic or probabilistic, and hierarchical or partitional. Some process complete solutions whereas others construct solutions during the search, and some maintain a single candidate solution whereas others work with

population of solutions [22]. The range of methods includes construction methods and local searches, metaheuristics, and mathematical programming techniques. Since this work focuses on the development of a hybrid genetic algorithm with population management, the remainder of this review focuses on other evolutionary methods for this problem, adaptations of K-means, as well as algorithms that currently report the best known solutions for the MSSC objective since these methods will be used for comparison purposes in Section 4.

Hruschka et al. [19] provide a comprehensive review and analysis of evolutionary algorithms for MSSC, comparing different solution encoding, crossover, and mutation strategies. As is the case for other combinatorial optimization problems, many genetic algorithms do not rely on random mutation and crossover only, but also integrate a local search to stimulate the discovery of high-quality solutions. Such algorithms are usually called *hybrid genetic* or *memetic* algorithms [9]. The algorithms of Fränti et al. [11], Kivijärvi et al. [26], Krishna and Murty [27], Lu et al. [33], Merz and Zell [35] are representative of this type of strategy and exploit K-means for solution improvement. In particular, Fränti et al. [11] and Kivijärvi et al. [26] propose a hybrid genetic algorithm based on a portfolio of six crossover operators. One of these, which inspired the crossover of the present work, pairs the centroids of two solutions via a greedy nearest-neighbor algorithm and randomly selects one center from each pair. The mutation operator relocates a random centroid in the location of a random sample, with a small probability. Although this method has some common mechanisms with HG-means, it also misses other key components: an exact matching-based crossover, population-management mechanisms favoring the removal of *clones*, and an adaptive parameter to control the attractiveness of outliers in the mutation. The variation (crossover and mutation) operators of Krishna and Murty [27], Lu et al. [33], Merz and Zell [35] are also different from those of HG-means. In particular, Krishna and Murty [27], Lu et al. [33] do not rely on crossover but exploit random mutation to reassign data points to clusters. Finally, Merz and Zell [35] considers an application of clustering for gene expression data, using K-means as a local search along with a crossover operator that relies on distance proximity to exchange centers between solutions.

Besides evolutionary algorithms and metaheuristics, substantial research has been conducted on incremental variants of the K-means algorithm [6,7,23,31,36], leading to the current state-of-the-art results for large-scale datasets. Incremental clustering algorithms construct a solution of MSSC iteratively, adding one center at a time. The global K-means algorithm [31] is such a method. Starting from a solution with k centers, the complete algorithm performs n runs of K-means, one from each initial solution containing the k existing centers plus sample $i \in \{1, \ldots, n\}$. The best solution with $k+1$ centers is stored, and the process is repeated until a desired number of clusters is attained. Faster versions of this algorithm can be designed, by greedily selecting a smaller subset of solutions for improvement at each step. For example, the modified global K-means (MGKM) of Bagirov [6] solves an auxiliary clustering problem to select one good initial solution at each step instead of considering all n possibilities. This algorithm was improved in Ordin and Bagirov [36] into a multi-start modified global K-means (MS-MGKM) algorithm, which generates several candidates at each step. Experiments on 16 real-world datasets show that MS-MGKM produces more accurate solutions than MGKM and the global K-means algorithm. These methods were also extended in Bagirov et al. [7] and Karmitsa et al. [23], by solving an optimization problem over a difference of convex (DC) functions in order to choose candidate initial solutions. Finally, Karmitsa et al. [24] introduced an incremental nonsmooth optimization algorithm based on a limited memory bundle method, which produces solutions in a short time. To this date, the MS-MGKM, DCClust and DCD-Bundle algorithms represent the current state-of-the-art in terms of solution quality.

Despite this extensive research, producing high-quality MSSC solutions in a consistent manner remains challenging for large datasets. Our algorithm, presented in the next section, helps to fill this gap.

3. Proposed methodology

HG-means is a hybrid metaheuristic that combines the exploration capabilities of a genetic algorithm and the improvement capabilities of a local search, along with general population-management strategies to preserve the diversity of the genetic information. Similarly to Kivijärvi et al. [26], Krishna and Murty [27] and some subsequent studies, the K-means algorithm is used as a local search. Moreover, the proposed method differs from previous work in its main variation operators: it relies on an exact bipartite matching crossover, uses a sophisticated adaptive mechanism in the mutation operator, as well as population-diversity management techniques.

The general scheme is given in Algorithm 1. Each individual P in the population is represented as a triplet $(\psi_P, \phi_P, \alpha_P)$ containing a membership chromosome ψ_P and a coordinate chromosome ϕ_P to represent the solution and a mutation parameter α_P to help balance the influence of outliers. The algorithm first generates a randomized initial population (Section 3.1) and then iteratively applies variation operators (selection, recombination, mutation) and local search (K-means) to evolve this population. At each iteration, two parents P_1 and P_2 are selected and crossed (Section 3.2), yielding the coordinates ϕ_C and mutation parameter α_C of an offspring C. A mutation operator is then applied to ϕ_C (Section 3.3), leading to an individual that is improved using the K-means algorithm (Section 3.4) and then included in the population.

```
1  Initialize population with Π_MAX individuals/solutions
2  while (number of iterations without improvement < N_1) ∧
   (number of iterations < N_2) do
3  |  Select parents P_1 and P_2 by binary tournament
4  |  Apply crossover to P_1 and P_2 to generate an offspring C
5  |  Mutate C to obtain C'
6  |  Apply local search (K-means) to C' to obtain an individual C''
7  |  Add C'' to the population
8  |  if the size of the population exceeds Π_MAX then
9  |  |  Eliminate clones and select Π_MIN survivors
10 Return best solution
```

Algorithm 1: HG-means – general structure.

Finally, each time the population exceeds a prescribed size Π_{MAX}, a survivor selection phase is triggered (Section 3.5), to retain only a diverse subset of Π_{MIN} good individuals. The algorithm terminates after N_1 consecutive iterations (generation of new individuals) without improvement of the best solution or a total of N_2 iterations have been performed. The remainder of this section describes each component of the method, in more detail.

3.1. Solution representation and initial population

Each individual P contains two chromosomes encoding the solution: a *membership* chromosome ϕ_P with n integers, specifying for each sample the index of the cluster with which it is associated; and a *coordinate* chromosome ψ_P with m real vectors in $\mathbb{R}^d$, representing the coordinates of the center of each cluster. The individual is completed with a mutation parameter, $\alpha_P \in \mathbb{R}$. Fig. 1 illustrates this solution representation for a simple two-dimensional example with three centers.

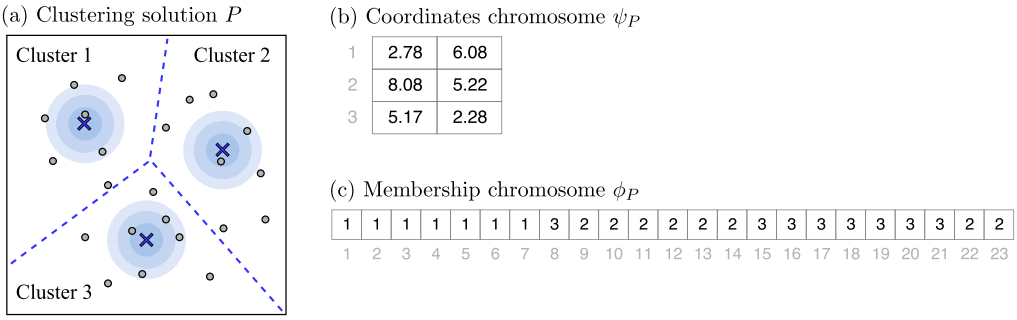

Fig. 1. Representation of an MSSC solution as a chromosome pair.

Observe that either of the two chromosomes is sufficient to characterize an MSSC solution. If only the membership chromosome ϕ_P is known, then Property 1 states that the center of each cluster should be located at the centroid of the points associated with it, and a trivial calculation gives the coordinates of each centroid in $\mathcal{O}(nd)$. If only the coordinate chromosome ψ_P is known, then Property 2 states that each sample should be associated with its closest center, and a simple linear search in $\mathcal{O}(nmd)$, by calculating the distances of all the centers from each sample, gives the membership chromosome. Finally, note that the iterative decoding of one chromosome into the other, until convergence, is equivalent to the K-means algorithm.

3.2. Selection and crossover

The generation of each new individual begins with the selection of two parents P_1 and P_2. The parent selection is done by binary tournament. A binary tournament selects two random solutions in the population with uniform probability and retains the one with the best fitness. The fitness considered in HG-means is simply the value of the objective function (MSSC cost).

Then, the coordinate chromosomes ψ_{P_1} and ψ_{P_2} of the two parents serve as input to the *matching crossover* (MX), which generates the coordinate chromosome ψ_C of an offspring in two steps:

- Step 1) The MX solves a bipartite matching problem to pair-up the centers of the two parents. Let $G = (U, V, E)$ be a complete bipartite graph, where the vertex set $U = (u_1, \ldots, u_m)$ represents the centers of parent P_1, and the vertex set $V = (v_1, \ldots, v_m)$ represents the centers of parent P_2. Each edge $(u_i, v_j) \in E$, for $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, m\}$ represents a possible association of center i from parent P_1 with center j from parent P_2. Its cost $c_{ij} = ||\psi_{P_1}(i) - \psi_{P_2}(j)||$ is calculated as the Euclidean distance between the two centers. A minimum-cost bipartite matching problem is solved in the graph G using an efficient implementation of the Hungarian algorithm [28], returning m pairs of centers in $\mathcal{O}(m^3)$ time.
- Step 2) For each pair obtained at the previous step, the MX randomly selects one of the two centers with equal probability, leading to a new *coordinate* chromosome with m centers and inherited characteristics from both parents.

Finally, the mutation parameter of the offspring is obtained as a simple average of the parent values: $\alpha_C = \frac{1}{2}(\alpha_{P_1} + \alpha_{P_2})$.

The MX is illustrated in Fig. 2 on the same example as before. This method can be viewed as an extension of the third crossover of Fränti et al. [11], using an exact matching algorithm instead of a greedy heuristic. The MX has several important properties. First, each center in ψ_C belongs to at least one parent, therefore promoting the transmission of good building blocks [18]. Second, although any MSSC solution admits $m!$ symmetrical representations, obtained by reindexing its clusters or reordering its centers, the coordinate chromosome generated by the MX will contain the same centers, regardless of this order. In combination with population management mechanisms (Section 3.5), this helps to avoid the propagation of similar solutions in the population and prevents premature convergence due to a loss of diversity.

3.3. Mutation

The MX is deterministic and strictly inherits existing solution features from both parents. In contrast, our mutation operator aims to introduce new randomized solution characteristics. It receives as input the coordinate chromosome ψ_C of the offspring and its mutation parameter α_C. It has two steps:

- Step 1) It "mutates" the mutation parameter as follows:

$$\alpha_{C'} = \max\{0, \min\{\alpha_C + X, 1\}\}, \tag{7}$$

  where X is a random number selected with uniform probability in $[-0.2, 0.2]$. Such a mechanism is typical in evolution strategies and allows an efficient parameter adaptation during the search.
- Step 2) It uses the newly generated $\alpha_{C'}$ to mutate the coordinate chromosome ψ_C by the biased relocation of a center:
  - Select a random center with uniform probability for removal.
  - Re-assign each sample p_i to the closest remaining center (but do not modify the positions of the centers). Let d_i^c be the distance of each sample p_i from its closest center.
  - Select a random sample p_Y and add a new center in its position, leading to the new coordinate chromosome $\psi_{C'}$. The selection of p_Y is done by roulette wheel, based on the following mixture of two probability distributions:

$$P(Y = i) = \left((1 - \alpha_{C'}) \times \frac{1}{n}\right) + \left(\alpha_{C'} \times \frac{||p_i - d_i^c||}{\sum_{j=1}^{n} ||p_j - d_j^c||}\right) \tag{8}$$

The value of $\alpha_{C'}$ in Eq. (8) drives the behavior of the mutation operator. In the case where $\alpha_{C'} = 0$, all the samples have an equal chance of selection, independently of their distance from the closest center. When $\alpha_{C'} = 1$, the selection probability is proportional to the distance, similarly to the initialization phase of K-means++ [5]. Considering sample-center distances increases the odds of selecting a good position for the new center. However, this could also drive the center positions toward outliers and reduce the solution diversity. For these reasons, $\alpha_{C'}$ is adapted instead of remaining fixed.

The mutation operator requires $\mathcal{O}(nmd)$ time and is illustrated in Fig. 3. The top-right center is selected for removal, and a new center is reinserted on the bottom-right of the figure. These new center coordinates constitute a starting point for the K-means local search discussed in the following section. The first step of

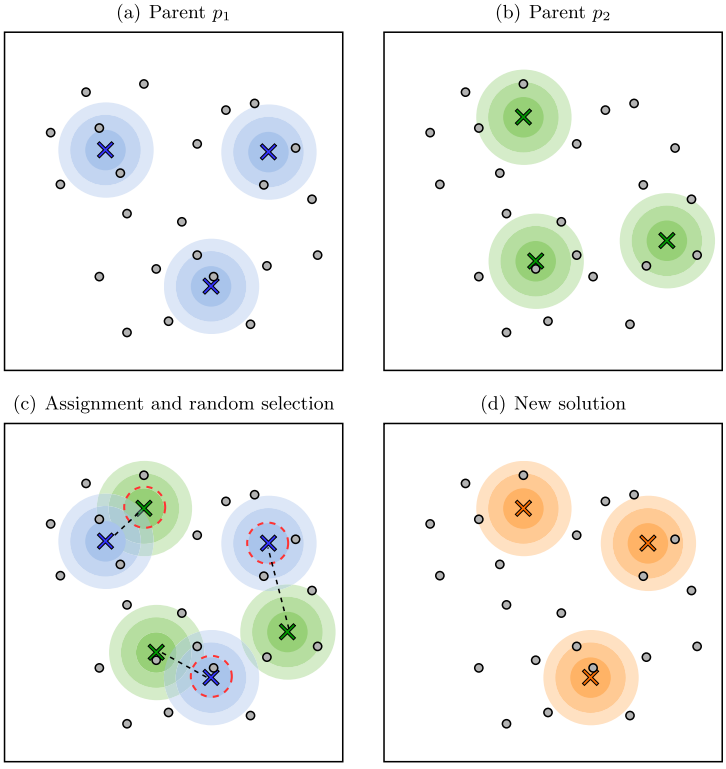

Fig. 2. Crossover based on centroid matching: (a) First parent; (b) Second parent; (c) Assignment of centroids and random selection; (d) Resulting offspring.

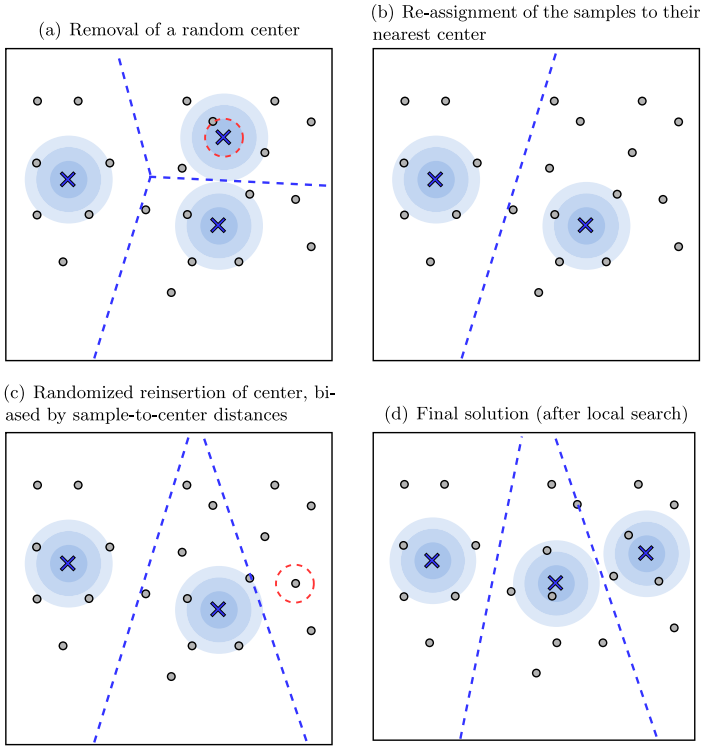

Fig. 3. Mutation operator by relocation of a single centroid.

K-MEANS will be to reassign each sample to its closest center (Fig. 3(c)), which is equivalent to recovering the membership chromosome $\phi_{C'}$. After a few iterations, K-MEANS converges to a new local minimum (Fig. 3(d)), giving a solution that is added to the population.

3.4. Local search

Each coordinate chromosome generated through selection, crossover, and mutation serves as a starting point for a local search based on K-MEANS. This algorithm iteratively (1) reassigns each sample to its closest center and (2) moves each center position

to the centroid of the samples to which it is assigned. These two steps are iterated until convergence to a local optimum. For this purpose, we use the fast K-means of [13]. This algorithm has a worst-case complexity of $\mathcal{O}(nmd + md^2)$ per loop, and it exploits lower bounds on the distances to eliminate many distance evaluations while retaining the same results as the classical K-means. Moreover, in the exceptional case in which some clusters are left empty after the application of K-means (i.e., one center has no allocated sample), the algorithm selects new center locations using Eq. (8) and restarts the K-means algorithm to ensure that all the solutions have exactly m clusters.

3.5. Population management

One critical challenge for population-based algorithms is to avoid the premature convergence of the population. Indeed, the elitist selection of parents favors good individuals and tends to reduce the population diversity. This effect is exacerbated by the application of K-means as a local search, since it concentrates the population on a smaller subset of local minima. Once diversity is lost, the chances of generating improving solutions via crossover and mutation are greatly reduced. To overcome this issue, HG-means relies on diversity management operators that find a good balance between elitism and diversification and that allow progress toward unexplored regions of the search space without excluding promising individuals. Similar techniques have been shown to be essential to progress toward high-quality solutions for difficult combinatorial optimization problems [40,43].

Population management. The initial population of HG-means is generated by Π_{MAX} runs of the standard K-means algorithm, in which the initial center locations are randomly selected from the set of samples. Moreover, the mutation parameter of each individual is randomly initialized in [0,1] with an uniform distribution. Subsequently, each new individual is directly included in the population and thus has *some chance* to be selected as a parent by the binary tournament operator, regardless of its quality. Whenever the population reaches a maximum size Π_{MAX}, a *survivor selection* is triggered to retain a subset of Π_{MIN} individuals.

Survivors selection. This mechanism selects $\Pi_{\text{MAX}} - \Pi_{\text{MIN}}$ individuals in the population for removal. To promote population diversity, the removal first focuses on clone individuals. Two individuals P and P' are *clones* if they have the same center positions. When a pair of clones is detected, one of the two individuals is randomly eliminated. When no more clones remain, the survivors selection proceeds with the elimination of the worst individuals, until a population of size Π_{MIN} is obtained.

Complexity analysis. HG-means has an overall worst-case complexity of $\mathcal{O}(\Pi_{\text{MAX}}\Phi_{\text{KM}} + N_2\Phi_{\text{CROSS}} + N_2\Phi_{\text{MUT}} + N_2\Phi_{\text{KM}})$, where Φ_{CROSS}, Φ_{MUT}, and Φ_{KM} represent the time spent in the crossover, mutation, and K-means procedures. The mutation and crossover methods are much faster than the K-means local search in practice, so HG-means' CPU time is proportional to the product of $(\Pi_{\text{MAX}} + N_2)$ with the time of K-means. Under strict CPU time limits, Π_{MAX} and N_2 could be set to small constants to obtain fast results (see Section 4.4). Moreover, since HG-means maintains a population of complete solutions with m clusters, it has good "anytime behavior" since it can be interrupted whenever necessary to return the current best solution.

4. Experimental analysis

We conducted extensive computational experiments to evaluate the performance of HG-means. After a description of the datasets and a preliminary parameter calibration (Sections 4.1–4.2), our first analysis focuses on solution quality from the perspective of the MSSC optimization problem (Section 4.3). We compare the solution quality obtained by HG-means with that of the current state-of-the-art algorithms, in terms of objective function value and computational time, and we study the sensitivity of the method to changes in the parameters. Our second analysis concentrates on the scalability of the method, studying how the computational effort grows with the dimensionality of the data and the number of clusters (Section 4.4). Finally, our third analysis evaluates the correlation between solution quality from an optimization viewpoint and clustering performance via external cluster validity measures. It compares the performance of HG-means, K-means, and K-means++ on a fundamental task, that of recovering the parameters of a non separable mixture of Gaussians (Section 4.5). Yet, we consider feature spaces of medium to high dimensionality (20 to 500) in the presence of a medium to large number of Gaussians (50 to 1,000). We show that the improved solution quality of HG-means directly translates into better clustering performance and more accurate information retrieval.

The algorithms of this paper were implemented in C++ and compiled with G++ 4.8.4. The source code is available at https://github.com/danielgribel/hg-means. The experiments were conducted on a single thread of an Intel Xeon X5675 3.07 GHz processor with 8 GB of RAM.

4.1. Datasets

We evaluated classical and recent state-of-the-art algorithms [6,7,23,24,26,36], in terms of solution quality for the MSSC objective, on a subset of datasets from the UCI machine learning repository (http://archive.ics.uci.edu/ml/). We collected all the recent datasets used in these studies for a thorough experimental comparison. The resulting 29 datasets, listed in Table 1, arise from a large variety of applications (e.g. cognitive psychology, genomics, and particle physics) and contain numeric features without missing values.

Their dimensions vary significantly, from 59 to 434,874 samples, and from 2 to 5,000 features. Each dataset has been considered with different values of m (number of clusters), leading to a variety of MSSC test set-ups. These datasets are grouped into four classes. Classes A1 and A2 have small datasets with 59 to 768 samples, while Class B has medium datasets with 1060 to 20,000 samples. These three classes were considered in Ordin and Bagirov [36]. Class C has larger datasets collected in Karmitsa et al. [24], with 13,910 to 434,874 samples and sometimes a large number of features (e.g., *Isolet* and *Gisette*).

4.2. Parameter calibration

HG-means is driven by four parameters: two controlling the population size (Π_{MIN} and Π_{MAX}) and two controlling the algorithm termination (maximum number of consecutive iterations without improvement N_1, and overall maximum number of iterations N_2). Changing the termination criteria leads to different nondominated trade-offs between solution quality and CPU time. We therefore set these parameters to consume a smaller CPU time than most existing algorithms on large datasets, while allowing enough iterations to profit from the enhanced exploration abilities of HG-means, leading to $(N_1, N_2) = (500, 5000)$. Subsequently, we calibrated the population size to establish a good balance between exploitation (number of generations before termination) and exploration (number of individuals). We compared the algorithm versions with $\Pi_{\text{MIN}} \in \{5, 10, 20, 40, 80\}$ and $\Pi_{\text{MAX}} \in \{10, 20, 50, 100, 200\}$ on medium-sized datasets of class A2 and B with $m \in \{2, 10, 20, 30, 50\}$. The setting $(\Pi_{\text{MIN}}, \Pi_{\text{MAX}}) = (10, 20)$ had the best performance and forms our baseline configuration. The impact of deviations from this parameter setting will be studied in the next section.

Table 1
Datasets used for performance comparisons on the MSSC optimization problem.

Group	Dataset	n	d	$n \times d$	Clusters
A1	German Towns	59	2	118	$m \in \{2, 3, 4 \ldots$
	Bavaria Postal 1	89	3	267	5, 6, 7, 8, 9, 10$\}$
	Bavaria Postal 2	89	4	356	
	Fisher's Iris Plant	150	4	600	
A2	Liver Disorders	345	6	2k	$m \in \{2, 5, 10, 15 \ldots$
	Heart Disease	297	13	4k	20, 25, 30, 40, 50$\}$
	Breast Cancer	683	9	6k	
	Pima Indians Diabetes	768	8	6k	
	Congressional Voting	435	16	7k	
	Ionosphere	351	34	12k	
B	TSPLib1060	1,060	2	2k	$m \in \{2, 10, 20, 30 \ldots$
	TSPLib3038	3,038	2	6k	40, 50, 60, 80, 100$\}$
	Image Segmentation	2,310	19	44k	
	Page Blocks	5,473	10	55k	
	Pendigit	10,992	16	176k	
	Letters	20,000	16	320k	
C	D15112	15,112	2	30k	$m \in \{2, 3, 5, 10 \ldots$
	Pla85900	85,900	2	172k	15, 20, 25$\}$
	EEG Eye State	14,980	14	210k	
	Shuttle Control	58,000	9	522k	
	Skin Segmentation	245,057	3	735k	
	KEGG Metabolic Relation	53,413	20	1M	
	3D Road Network	434,874	3	1M	
	Gas Sensor	13,910	128	2M	
	Online News Popularity	39,644	58	2M	
	Sensorless Drive Diagnosis	58,509	48	3M	
	Isolet	7,797	617	5M	
	MiniBooNE	130,064	50	7M	
	Gisette	13,500	5,000	68M	

4.3. Performance on the MSSC optimization problem

We tested HG-means on each MSSC dataset and number of clusters m. Tables 2 and 3 compare its solution quality and CPU time with those of classical and recent state-of-the-art algorithms:

- GKM [31] – global K-means;
- SAGA [26] – self-adaptive genetic algorithm;
- MGKM [6] – modified global K-means;
- MS-MGKM [36] – multi-start modified global K-means;
- DCClust and MS-DCA [7] – clustering based on a difference of convex functions;
- DCD-Bundle [23] – diagonal bundle method;
- LMBM-Clust [24] – nonsmooth optimization based on a limited memory bundle method.

We also report the results of the classical K-means and K-means++ algorithms (efficient implementation of Hamerly [13]) for both a single run and the best solution of 5000 repeated runs with different initial solutions.

The datasets and numbers of clusters m indicated in Table 1 lead to 235 test set-ups. For ease of presentation, each line of Tables 2 and 3 is associated with one dataset and displays averaged results over all values of m. The detailed results of HG-means are available at https://w1.cirrelt.ca/~vidalt/en/research-data.html. For each dataset and value of m, the solution quality is measured as the percentage gap from the best-known solution (BKS) value reported in all previous articles (from multiple methods, runs, and parameter settings). This gap is expressed as $\text{Gap}(\%) = 100 \times (z - z_{\text{BKS}})/z_{\text{BKS}}$, where z represents the solution value of the method considered, and z_{BKS} is the BKS value. A negative gap means that the solutions for this dataset are better than the best solutions found previously. Finally, the last two lines indicate the CPU model used in each study, along with the time-scaling factor (based on the Passmark benchmark) representing the ratio between its speed and that of our processor. All time values in this article have been multiplied by these scaling factors to account for processor differences.

HG-means produces solutions of remarkable quality, with an average gap of -0.40% and -0.26% on the small and large datasets, respectively. This means that its solutions are better, on average, than the best solutions ever found. For all datasets, HG-means achieved the best gap value. The statistical significance of these improvements is confirmed by pairwise Wilcoxon tests between the results of HG-means and those of other methods (with p-values $< 10^{-8}$). Over all 235 test set-ups (dataset $\times$ number of cluster combinations), HG-means found 113 solutions better than the BKS, 116 solutions of equal quality, and only five solutions of lower quality. We observe that the improvements are also correlated with the size of the datasets. For the smallest ones, all methods perform relatively well. However, for more complex applications involving a larger number of samples, a feature space of higher dimension, and more clusters, striking differences in solution quality can be observed between the state-of-the-art methods.

These experiments also confirm the known fact that a single run of K-means or K-means++ does not usually find a good local minimum of the MSSC problem, as shown by gap values that can become arbitrarily high. For the Eye and Miniboone datasets, in particular, a misplaced center can have large consequences in terms of objective value. The probability of falling into such a case is high in a single run, but it can be reduced by performing repeated runs and retaining the best solution. Nevertheless, even 5000 independent runs of K-means or K-means++ are insufficient to achieve solutions of a quality comparable to that of HG-means.

In terms of computational effort, HG-means is generally faster than SAGA, MS-MGKM, DCClust, MS-DCA, and DCD-Bundle (the current best methods in terms of solution quality), but slower than LMBM-Clust, since this method is specifically designed and calibrated to return quick solutions. It is also faster than a repeated K-means or K-means++ algorithm with 5000 restarts, i.e., a number of restarts equal to the maximum number of iterations of the

Table 2
Performance comparison for small and medium MSSC datasets.

	K-means				K-means^{++}				GKM		SAGA		MGKM		MS-MGKM		HG-means	
	Single run		5000 runs		Single run		5000 runs											
	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)
German	20.34	0.00	0.00	0.15	15.12	0.00	−**0.08**	0.14	0.76	0.00	−**0.08**	0.22	1.00	0.00	0.47	0.01	−**0.08**	0.02
Bavaria1	789.66	0.00	8.28	0.36	10.34	0.00	**0.00**	0.19	1.03	0.00	**0.00**	0.26	0.17	0.01	0.07	0.00	**0.00**	0.02
Bavaria2	738.96	0.00	5.63	0.49	37.14	0.00	−**0.05**	0.25	1.37	0.00	−**0.05**	0.29	1.37	0.01	0.14	0.00	−**0.05**	0.03
Iris	20.47	0.00	**0.00**	0.48	10.92	0.00	**0.00**	0.49	1.48	0.01	**0.00**	0.53	1.48	0.01	0.09	0.03	**0.00**	0.09
Liver	26.34	0.00	6.90	11.54	8.85	0.00	1.22	9.00	13.44	0.08	-0.06	5.92	12.15	0.21	0.00	1.59	−**0.94**	1.82
Heart	15.24	0.00	3.48	11.08	7.39	0.00	1.52	11.12	1.64	0.07	0.55	16.66	1.63	0.24	0.04	1.51	−**0.55**	2.17
Breast	20.31	0.01	4.95	27.63	5.43	0.00	1.61	22.22	3.29	0.24	1.15	19.21	1.30	0.52	0.00	2.02	−**0.45**	5.56
Pima	21.75	0.01	2.60	35.56	6.12	0.01	0.82	24.72	1.01	0.32	0.87	18.58	0.95	0.64	0.00	3.70	−**0.11**	5.60
Congressional	7.53	0.00	2.34	20.93	5.89	0.01	1.64	23.71	2.61	0.14	0.74	16.34	0.93	0.42	0.00	2.77	−**0.88**	3.91
Ionosphere	14.60	0.01	5.40	30.77	15.56	0.01	3.24	27.97	4.96	0.11	1.05	21.08	0.47	1.43	0.13	1.70	−**1.59**	5.54
TSPLib1060	16.83	0.00	4.04	20.98	11.03	0.00	2.35	18.20	2.63	1.06	0.81	30.83	2.33	1.07	0.20	6.53	−**0.15**	4.15
TSPLib3038	5.54	0.02	1.02	89.16	4.54	0.02	0.74	80.96	1.43	25.32	0.00	35.77	1.07	8.16	0.23	46.31	−**0.24**	16.66
Image	41.67	0.08	16.45	369.38	11.01	0.06	1.63	259.30	1.26	11.67	1.54	143.90	1.31	24.75	0.24	35.34	−**0.01**	57.51
Page	2960.65	0.80	911.35	4538.52	14.13	0.08	1.11	340.92	1.25	129.99	61.58	134.88	0.93	97.26	0.07	31.72	−**0.96**	143.67
Pendigit	3.31	0.50	0.36	2504.67	2.50	0.50	0.22	2275.14	0.24	263.04	0.74	607.68	0.16	434.83	0.04	352.36	−**0.18**	461.13
Letters	1.98	1.26	0.14	6209.46	1.35	1.35	0.10	6751.95	0.35	1102.38	0.42	1114.36	0.13	1859.64	0.01	908.70	−**0.18**	1326.05
Avg. Gap	294.07		60.81		10.46		1.00		2.42		4.33		1.71		0.11		−**0.40**	
CPU	Xe 3.07 GHz				Xe 3.07 GHz				Core2 2.5 GHz		Xe 3.07 GHz		Core2 2.5 GHz		Core2 2.5 GHz		Xe 3.07 GHz	
Passmark	1403 (1.00)				1403 (1.00)				976 (0.70)		1403 (1.00)		976 (0.70)		976 (0.70)		1403 (1.00)	

Table 3
Performance comparison for large MSSC datasets.

	K-means				K-means^{++}				GKM		SAGA		LMBM		MS-MGKM		DCClust		MS-DCA		DCD-Bundle		HG-means	
	Single run		5000 runs		Single run		5000 runs																	
	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)
D15112	1.60	0.04	**0.00**	160.69	1.18	0.03	**0.00**	146.53	0.34	43.25	0.19	47.29	0.34	4.58	0.13	11.28	0.12	16.26	0.13	35.73	0.13	9.60	**0.00**	17.52
Pla85900	0.46	0.31	**−0.02**	1115.56	0.79	0.26	**−0.02**	1060.07	0.25	2023.84	0.15	260.18	0.95	22.36	0.10	2094.58	0.14	200.30	0.09	1416.24	0.15	185.61	**−0.02**	198.14
Eye	880402.99	0.39	0.00	1522.87	48.95	0.23	0.58	961.63	0.81	161.18	0.04	212.00	0.75	6.62	0.74	17.59	0.89	41.23	0.75	121.77	0.98	19.45	**−0.02**	196.43
Shuttle	181.15	0.81	121.42	2832.02	22.49	0.25	−0.66	911.64	0.35	1954.72	−0.90	475.43	0.10	4.55	0.41	89.27	0.46	227.47	0.41	4722.90	1.71	312.10	**−0.91**	97.84
Skin	9.63	0.60	0.05	2304.23	7.71	0.43	−0.38	1728.25	0.28	22518.92	0.13	844.14	3.93	14.41	0.63	3021.33	0.32	1233.00	0.33	8774.03	0.32	1259.01	**−0.41**	230.25
Kegg	94.45	4.21	76.24	17988.47	6.96	0.58	−0.49	1204.77	1.85	4147.46	3.54	686.18	1.52	10.78	1.52	445.03	1.18	488.31	1.02	3442.98	0.89	576.76	**−0.51**	244.37
3Droad	0.23	5.55	**0.00**	35237.41	0.28	2.99	**0.00**	13437.58	**0.00**†	42431.72†	0.02	1233.00	**0.00**	63.21	0.54	60180.29	0.56	4924.72	0.44	31325.93	0.01	4872.71	**0.00**	2862.09
Gas	21.42	1.57	1.87	5182.01	7.27	1.12	−0.17	2482.24	0.24	1550.29	−0.20	983.33	0.86	86.16	0.02	256.74	0.24	814.34	0.05	2404.14	0.55	627.13	**−0.22**	521.57
Online	18.65	1.57	0.51	5245.66	19.74	1.33	**−0.17**	2653.76	–	–	−0.15	1160.02	4.54	96.18	0.12	795.22	0.00	1509.81	–	–	0.26	1600.50	**−0.17**	473.23
Sensorless	155.50	4.91	46.02	18798.70	19.06	2.50	**−0.44**	8237.59	–	–	1.16	2004.10	1.18	25.24	0.27	1249.96	2.58	2133.54	–	–	–	–	−0.41	1077.67
Isolet	1.93	3.61	**−0.21**	11082.28	1.33	3.74	**−0.21**	12587.68	–	–	**−0.21**	3961.39	0.49	97.59	0.39	677.14	0.32	1672.82	–	–	–	–	**−0.21**	1846.70
Miniboone	40992.86	15.63	−0.07	57148.73	1.75	9.63	**−0.10**	27611.08	–	–	−0.07	4883.13	3.50	88.41	0.29	7559.34	0.24	9656.14	–	–	0.23	9291.40	**−0.10**	2941.52
Gisette	−0.47	77.17	–	–	−0.47	96.63	–	–	–	–	–	–	0.03	1871.67	0.01‡	39504.13‡	0.00†	49847.13†	–	–	–	–	**−0.52**	22279.47
Avg. Gap*	110088.99		24.94		11.96		−0.14		0.52		0.37		1.05		0.51		0.49		0.40		0.59		**−0.26**	
CPU	Xe 3.07 GHz				Xe 3.07 GHz				I5 2.9 GHz		Xe 3.07 GHz		I7 4.0 GHz		I7 4.0 GHz		I7 4.0 GHz		I5 2.9 GHz		I5 1.6/2.7 GHz		Xe 3.07 GHz	
Passmark	1403 (1.00)				1403 (1.00)				1859 (1.32)		1403 (1.00)		2352 (1.68)		2352 (1.68)		2352 (1.68)		1859 (1.32)		1432 (1.02)		1403 (1.00)	

* Considering the subset of 8 instances which is common to all methods. † Considering $m \in \{2, 3, 5\}$. ‡ Considering $m \in \{2, 3, 5, 10\}$.

Table 4
Sensitivity of HG-MEANS to changes of population-size parameters.

	Average gap (%)						Median time (s)				
	Π_{MAX}	= 10	20	50	100	200	10	20	50	100	200
$\Pi_{MIN} = 5$		−0.31	−0.31	−0.28	−0.24	−0.20	12.03	11.81	13.54	14.73	17.79
10			−0.32	−0.28	−0.26	−0.23		12.03	11.53	14.29	16.76
20				−0.31	−0.25	−0.22			15.27	15.94	13.44
40					−0.25	−0.14				13.35	15.19
80						−0.11					19.48

Table 5
Sensitivity of HG-MEANS to changes of the termination criterion.

$N_2 = 10 \times N_1$	Average gap (%)					Median time (s)				
	$N_1 = 50$	100	250	500	1000	50	100	250	500	1000
$(\Pi_{MIN}, \Pi_{MAX}) = (5,10)$	0.20	−0.07	−0.25	−0.31	−0.35	1.87	3.31	7.40	12.03	29.10
(10,20)	0.52	−0.02	−0.22	−0.32	−0.36	1.54	4.13	8.25	12.03	29.90

Table 6
Performance of HG-MEANS as a function of the number of clusters.

	Gap (%)							Time (s)						
	$m = 2$	3	5	10	15	20	25	2	3	5	10	15	20	25
D15112	0.00	0.00	0.00	0.00	0.00	0.00	−0.03	2.80	5.23	4.97	9.36	37.12	22.45	40.74
Pla85900	0.00	0.00	0.00	0.00	0.00	−0.02	−0.13	23.94	34.96	95.62	135.77	131.92	349.98	614.79
Eye	0.00	0.00	0.00	−0.01	0.00	0.00	−0.16	4.38	5.22	13.60	91.01	121.20	509.55	630.08
Shuttle	0.00	0.00	0.00	−0.02	0.00	−3.67	−2.68	17.53	18.93	22.91	45.74	63.10	175.70	340.98
Skin	0.00	0.00	0.00	0.00	−1.63	−0.89	−0.38	66.23	90.92	96.21	176.57	336.70	213.78	631.37
Kegg	0.00	0.00	0.00	0.00	−1.29	−1.26	−1.03	41.45	66.68	90.63	117.53	226.37	424.89	743.02
3Droad	0.00	0.00	0.00	0.00	0.00	0.00	0.00	444.22	535.59	498.42	2824.24	2582.08	5888.17	7261.91
Gas	0.00	0.00	0.00	−0.18	−0.94	−0.21	−0.18	93.20	87.71	156.56	222.77	827.35	920.49	1342.93
Online	0.00	0.00	0.00	0.00	0.00	−0.01	−1.20	109.80	87.27	190.75	333.71	285.21	1154.91	1150.95
Sensorless	0.00	0.00	0.00	−2.42	0.00	−0.63	0.17	88.43	256.11	236.20	646.31	1004.29	2179.93	3132.41
Isolet	0.00	0.00	0.00	0.00	−0.15	−0.39	−0.96	255.04	322.27	751.86	748.94	1992.24	2521.89	6334.68
Miniboone	0.00	0.00	0.00	0.00	0.00	−0.12	−0.57	209.08	565.88	585.91	1329.57	4758.04	5061.36	8080.78
Gisette	0.00	0.00	−0.02	0.00	−0.51	−1.85	−1.28	2304.41	3896.54	10964.95	20617.57	31767.12	39283.79	47121.92

algorithm. This can be partly explained by the fact that the solutions generated by the exact matching crossover require less time to converge via K-MEANS than initial sample points selected according to some probability distributions. Moreover, a careful exploitation of the search history, represented by the genetic material of high-quality parent solutions, makes the method more efficient and accurate.

Finally, we measured the sensitivity of HG-MEANS to changes in its parameters: (Π_{MIN}, Π_{MAX}) defining the population-size limits, and (N_1, N_2) defining the termination criterion. In Table 4, we fix the termination criterion to $(N_1, N_2) = (500, 5000)$ and consider a range of population-size parameters, reporting the average gap and median time over all datasets for each configuration. The choice of Π_{MIN} and Π_{MAX} appears to have only a limited impact on solution quality and CPU time: regardless of the parameter setting, HG-MEANS returns better average solutions than all individual best known solutions collected from the literature. Some differences can still be noted between configurations: as highlighted by pairwise Wilcoxon tests, every configuration underlined in the table performs better than every non-underlined one (with p-values $\leq$ 0.018). Letting the population rise to the double of the minimum population size ($\Pi_{MAX} \approx 2 \times \Pi_{MIN}$) before survivors selection is generally a good policy. Moreover, we observe that smaller populations trigger a faster convergence but at the risk of reducing diversity, whereas excessively large populations (i.e., $\Pi_{MAX} = 200$) unnecessarily spread the search effort, with an adverse impact on solution quality.

In Table 5, we retain two of the best population-size configurations $(\Pi_{MIN}, \Pi_{MAX}) \in \{(5, 10), (10, 20)\}$ and vary the termination criterion. Naturally, the quality of the solutions improves with longer runs, but even a short termination criterion such as $(N_1, N_2) = (100, 1000)$ already gives good solutions, with an average gap of −0.07%. Finally, reducing the population size to $(\Pi_{MIN}, \Pi_{MAX}) = (5, 10)$ for short runs allows us to better exploit a limited number of iterations, whereas the baseline setting performs slightly better for longer runs.

4.4. Scalability

The solution quality and computational efficiency of most clustering algorithms is known to deteriorate as the number of clusters m grows, since this leads to more complex combinatorial problems with numerous local minima. To evaluate how HG-MEANS behaves in these circumstances, we conduct additional experiments focused on the large datasets of class C. Table 6 reports the solution quality and CPU time of HG-MEANS for each dataset as a function of m. Moreover, to explore the case where the CPU time is more restricted, Table 7 reports the same information for a *fast* HG-MEANS configuration where $(\Pi_{MIN}, \Pi_{MAX}) = (5, 10)$ and $(N_1, N_2) = (50, 500)$.

As observed in Table 6, HG-MEANS retrieves or improves the BKS for all datasets and values of m. Significant improvements are more frequently observed for larger values of m. A likely explanation is that the global minimum has already been found for most datasets with a limited number of clusters, whereas previous methods did not succeed in finding the global optimum for larger values of m. In terms of computational effort, there is a visible correlation between the number of clusters m and the CPU time. Power law regressions of the form $f(m) = \alpha m^{\beta}$ indicate that the computational effort of HG-MEANS grows as $\Theta(m^{2.09})$ for Eye

Table 7
Performance of a fast configuration of HG-means as a function of the number of clusters.

	Gap (%)							Time (s)						
	$m = 2$	3	5	10	15	20	25	2	3	5	10	15	20	25
D15112	0.00	0.00	0.00	0.00	0.00	0.00	−0.03	0.29	0.95	0.54	1.77	1.93	4.53	13.84
Pla85900	0.00	0.00	0.00	0.00	0.00	−0.02	−0.13	3.21	3.86	5.22	11.84	29.58	42.44	53.23
Eye	0.00	0.00	0.00	−0.01	0.00	0.00	−0.16	0.68	1.08	1.10	12.28	43.89	31.88	92.41
Shuttle	0.00	0.00	0.00	0.46	0.02	−3.55	−2.68	2.03	2.36	6.50	12.40	35.60	57.68	67.04
Skin	0.00	0.00	0.00	0.00	−1.63	−0.89	−0.19	5.99	9.24	14.37	14.45	35.76	25.49	96.52
Kegg	0.00	0.00	0.00	0.00	−1.29	−1.25	−0.93	4.05	6.17	5.91	25.21	42.94	115.85	73.93
3Droad	0.00	0.00	0.00	0.00	0.00	0.00	0.00	18.37	30.66	38.11	394.55	166.58	288.66	619.88
Gas	0.00	0.00	0.00	−0.18	−0.94	−0.21	−0.18	4.98	11.07	13.74	69.15	94.68	106.94	182.29
Online	0.00	0.00	0.00	0.00	0.00	−0.01	−1.20	8.61	18.12	23.07	39.39	59.10	62.54	218.82
Sensorless	0.00	0.00	0.00	−2.42	0.00	−0.63	0.00	13.92	16.43	22.92	102.44	217.94	397.28	429.3
Isolet	0.00	0.00	0.00	0.00	−0.08	−0.39	−0.79	29.37	43.21	91.75	256.56	415.56	443.05	199.26
Miniboone	0.00	0.00	0.00	0.00	0.00	−0.12	−0.57	17.46	37.82	39.95	246.58	647.88	731.73	1469.62
Gisette	0.00	0.00	−0.02	0.53	−0.16	−1.62	−1.14	232.87	1067.71	1252.21	3190.44	5671.95	13542.21	12568.59

State, $\Theta(m^{1.38})$ for Miniboone, and $\Theta(m^{\beta})$ for $\beta \leq 1.29$ in all other cases. Similarly, for $m = 10$, fitting the CPU time of the method as a power law of the form $g(n, d) = \alpha n^{\beta} d^{\gamma}$ indicates that the measured CPU time of HG-means grows as $\mathcal{O}(n^{1.08} d^{0.88})$, i.e., linearly with the number of samples and the dimension of the feature space.

We observe a significant reduction in CPU time when comparing the results of the *fast* HG-means in Table 7 with those of the standard version in Table 6. Considering the speed ratio between methods for each dataset, the fast configuration is in average seven times faster than the standard HG-means and over 10 times faster than SAGA, MS-MGKM, DCClust, DCD-Bundle and MS-DCA. Fig. 4 also displays the CPU time of HG-means, its *fast* configuration, and the other algorithms listed in Section 4.3 as a function of m. Surprisingly, the solution quality did not deteriorate much by reducing the termination criterion for these large datasets: with a percentage gap of −0.25%, the solutions found by the fast HG-means are close to those of the standard version (gap of −0.26%) and still much better than all solutions found in previous studies. Therefore, researchers interested in using HG-means can easily adapt the termination criterion of the method, so as to obtain significant gains of solution quality within their own computational budget.

4.5. Solution quality and clustering performance

The previous section has established that HG-means finds better MSSC local minima than other state-of-the-art algorithms and exemplified the known fact that K-means or K-means++ solutions can be arbitrarily far from the global minimum. In most situations, using the most accurate method for a problem should be the best option. However, HG-means is slower than a few runs of K-means or K-means++. To determine whether it is worth investing this additional effort, we must determine whether a better solution quality for the MSSC problem effectively translates into better clustering performance. There have been similar investigations in other machine learning subfields, e.g., to choose the amount of effort dedicated to training deep neural networks (see, e.g. [17,25]).

To explore this, we conduct an experiment in which we compare the ability of HG-means, K-means and K-means++ to classify 50,000 samples issued from a mixture of spherical Gaussian distributions: $X \sim \frac{1}{m}\sum_{i=1}^{m} \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with $\boldsymbol{\Sigma}_i = \sigma_i^2 \mathbf{I}$. For each $i \in \{1, \ldots, m\}$, μ_i and σ_i^2 are uniformly selected in [0,5] and [1,10], respectively. This is a fundamental setting, without any hidden structure, in which we expect the MSSC model and the associated K-means variants to be a good choice since these methods promote spherical and balanced clusters. To increase the challenge, we consider a medium to large number of Gaussians, with $m \in \{20, 50, 100, 200\}$, in feature spaces of medium to high dimensions $d \in \{20, 50, 100, 200, 500\}$. For each combination of m and d, we repeat the generation process until we obtain a mixture that is not 1-separated and in which at least 99% of the pairs of Gaussians are $\frac{1}{2}$-separated [10]. Such a mixture corresponds to Gaussians that significantly overlap. These datasets can be accessed at https://w1.cirrelt.ca/~vidalt/en/research-data.html.

Tables 8 and 9 compare the results of HG-means with those of K-means and K-means++ over a single run or 500 repeated runs, in terms of MSSC solution quality (as represented by the percentage gap) and cluster validity in relation to the ground truth. We use three external measures of cluster validity: the adjusted Rand index (CRand – Hubert and Arabie [20]), the normalized mutual information (NMI – Kvalseth [29]), and the centroid index (CI – Fränti et al. [12]). CRand and NMI take continuous values in $[-1, 1]$ and [0,1], respectively. They converge toward 1 as the clustering solution becomes closer to the ground truth. CI takes integer values and measures the number of fundamental cluster differences between solutions, with a value of 0 indicating that the solution has the same cluster-level structure as the ground truth.

As in the previous experiments, a single run of K-means or K-means++ visibly leads to shallow local minima that can be improved with multiple runs from different starting points. However, even 500 repetitions of these algorithms are insufficient to reach the solution quality of HG-means, despite the similar CPU time. K-means performs better than K-means++ for these datasets, most likely because it is more robust to outliers when selecting initial center locations. A pairwise Wilcoxon test highlights significant differences between HG-means and all other methods (p-values ≤ 0.0002). With a Pearson coefficient $r \geq 0.8$, the dimension d of the feature space is correlated to the inaccuracy (percentage gap) of the repeated K-means and K-means++ algorithms, which appear to be more easily trapped in low-quality local minima for feature spaces of larger dimension.

Comparing the external clustering performance of these methods (via CRand, NMI, and CI) leads to additional insights. For all three metrics, pairwise Wilcoxon tests highlight significant performance differences between HG-means and repeated K-means variants (with p-values $\leq 3.1 \times 10^{-5}$). We also observe a correlation between the solution quality (percentage gap) and the three external measures. Although the differences in the MSSC objective function values appear small at first glance (e.g., average gap of 4.87% for repeated K-means), these inaccuracies have a large effect on cluster validity, especially for datasets with feature spaces of higher dimension. When $d = 500$, HG-means is able to exploit the increased amount of available independent information

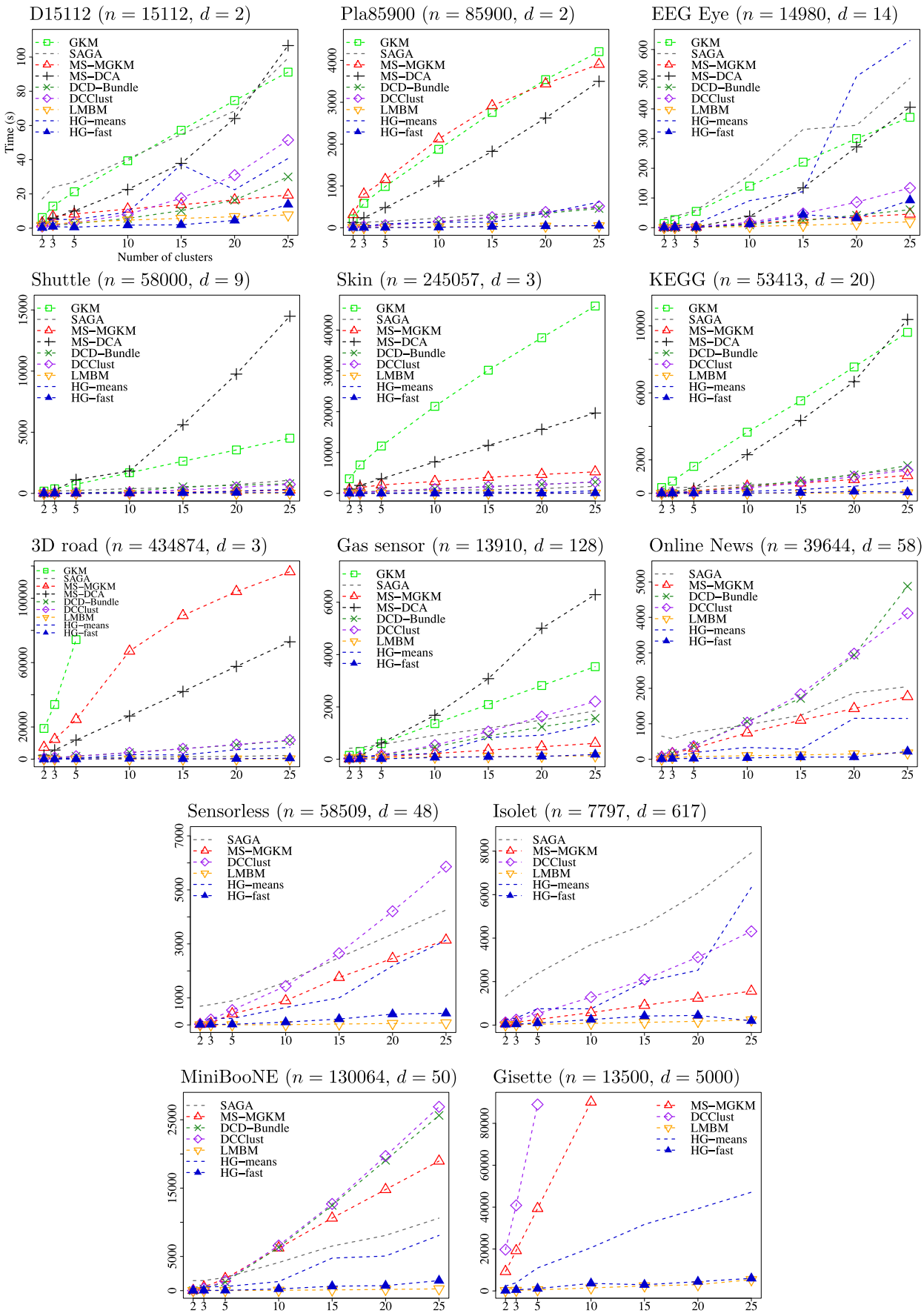

Fig. 4. CPU time of state-of-the-art algorithms on class C datasets, as a function of the number of clusters.

Table 8
Mixture of spherical Gaussian distributions – Solution quality and CPU time.

m	d	BKS	Gap (%)					Time (s)				
		Objective	K-means		K-means^{++}		HG-means	K-means		K-means^{++}		HG-means
		Value	1 run	500 runs	1 run	500 runs		1 run	500 runs	1 run	500 runs	
20	20	5432601.91	0.73	**0.0**	1.15	**0.0**	**0.0**	2.40	668.93	3.00	764.76	1085.40
20	50	12815114.52	6.19	**0.0**	3.75	1.15	**0.0**	2.86	860.95	3.17	1171.09	1308.96
20	100	24266784.28	14.84	**0.0**	5.01	4.83	**0.0**	5.38	1243.53	4.75	1958.25	553.25
20	200	59340268.17	17.70	2.57	11.79	7.00	**0.0**	14.90	2677.57	12.43	3938.29	1505.16
20	500	125359202.26	16.53	8.06	25.35	8.00	**0.0**	30.13	6118.59	25.17	8389.50	2563.73
50	20	5305274.24	0.47	**0.0**	0.43	**0.0**	**0.0**	5.03	2599.11	4.84	2755.17	3189.56
50	50	13864882.54	2.10	**0.0**	3.22	0.72	**0.0**	7.28	2695.69	8.23	3258.11	4307.12
50	100	25645070.92	8.86	3.70	12.04	5.76	**0.0**	10.78	4226.64	14.33	5871.70	2934.41
50	200	52561077.57	14.62	7.76	19.90	9.92	**0.0**	22.98	7837.70	37.60	11063.60	9629.09
50	500	143469250.17	16.92	9.79	20.0	11.11	**0.0**	38.89	14778.04	58.13	19077.48	18360.24
100	20	5027688.54	0.34	0.12	0.54	0.04	**0.0**	19.79	7281.83	18.89	8435.48	13529.09
100	50	12897680.57	3.07	1.17	4.81	2.25	**0.0**	12.07	6612.89	15.27	7962.07	10344.57
100	100	27284752.32	6.30	4.67	10.58	6.89	**0.0**	24.43	11864.87	30.54	14991.49	6728.71
100	200	51552765.51	14.03	7.97	15.78	11.13	**0.0**	34.63	14537.27	52.73	20128.89	20499.22
100	500	130903680.95	18.90	15.61	22.71	15.69	**0.0**	61.45	25313.95	86.04	34062.29	38217.57
200	20	4774890.45	0.72	0.45	1.24	0.53	**0.0**	42.85	18861.45	38.91	19896.36	38126.21
200	50	13490838.00	1.97	1.16	2.88	1.86	**0.0**	34.49	18792.14	39.88	21036.63	28513.22
200	100	27337380.17	8.08	5.29	9.68	7.56	**0.0**	70.30	30880.39	82.03	36219.66	39980.98
200	200	52946223.09	15.77	11.70	19.67	14.45	**0.0**	74.33	37459.76	139.62	46365.94	67745.79
200	500	135201463.76	20.97	17.32	23.83	19.28	**0.0**	142.85	63202.41	210.16	92765.62	93444.51

Table 9
Mixture of spherical Gaussian distributions – External cluster validity.

m	d	CRand					NMI					CI				
		K-means		K-means^{++}		HG-means	K-means		K-means^{++}		HG-means	K-means		K-means^{++}		HG-means
		1 run	500 runs	1 run	500 runs		1 run	500 runs	1 run	500 runs		1 run	500 runs	1 run	500 runs	
20	20	0.69	**0.72**	0.67	**0.72**	**0.72**	0.73	**0.75**	0.73	**0.75**	**0.75**	1	**0**	1	**0**	**0**
20	50	0.76	**0.98**	0.86	0.92	**0.98**	0.91	**0.98**	0.94	0.96	**0.98**	3	**0**	2	1	**0**
20	100	0.63	**1.00**	0.89	0.89	**1.00**	0.89	**1.00**	0.97	0.97	**1.00**	5	**0**	2	2	**0**
20	200	0.47	0.94	0.61	0.83	**1.00**	0.84	0.98	0.89	0.95	**1.00**	7	1	5	3	**0**
20	500	0.55	0.81	0.32	0.81	**1.00**	0.88	0.95	0.79	0.95	**1.00**	6	2	9	3	**0**
50	20	0.58	**0.59**	0.57	**0.59**	**0.59**	0.67	**0.68**	0.67	**0.68**	**0.68**	1	**0**	2	**0**	**0**
50	50	0.87	**0.94**	0.82	0.92	**0.94**	0.93	**0.95**	0.92	0.94	**0.95**	3	**0**	5	1	**0**
50	100	0.76	0.90	0.59	0.85	**1.00**	0.95	0.98	0.92	0.96	**1.00**	9	4	12	6	**0**
50	200	0.52	0.80	0.34	0.72	**1.00**	0.90	0.96	0.85	0.94	**1.00**	14	8	19	10	**0**
50	500	0.41	0.69	0.24	0.39	**1.00**	0.88	0.94	0.83	0.91	**1.00**	16	9	16	10	**0**
100	20	0.48	0.48	0.47	**0.49**	**0.49**	0.62	**0.63**	0.62	**0.63**	**0.63**	4	2	5	1	**0**
100	50	0.80	0.86	0.78	0.84	**0.91**	0.91	0.93	0.90	0.92	**0.94**	9	4	13	6	**0**
100	100	0.80	0.86	0.68	0.74	**0.99**	0.96	0.97	0.93	0.94	**1.00**	15	11	23	16	**1**
100	200	0.63	0.79	0.53	0.74	**0.99**	0.93	0.96	0.92	0.95	**1.00**	27	16	30	20	**1**
100	500	0.40	0.60	0.23	0.35	**0.98**	0.89	0.93	0.84	0.90	**1.00**	33	27	37	29	**2**
200	20	0.39	0.40	0.38	0.39	**0.41**	0.59	0.59	0.58	0.59	**0.60**	22	14	25	20	**6**
200	50	0.81	0.82	0.78	0.80	**0.87**	0.91	0.90	0.90	0.89	**0.92**	12	10	18	13	**0**
200	100	0.71	0.81	0.66	0.73	**0.96**	0.94	0.95	0.94	0.94	**0.99**	38	27	49	38	**5**
200	200	0.51	0.64	0.31	0.56	**0.99**	0.92	0.94	0.87	0.93	**1.00**	61	45	71	53	**3**
200	500	0.41	0.50	0.26	0.33	**0.98**	0.90	0.92	0.85	0.89	**1.00**	65	57	74	60	**5**

to find a close approximation to the ground truth (average CRand of 0.99, NMI of 1.00, and CI of 1.75) whereas repeated K-means and K-means++ reach shallow local optima and obtain inaccurate results (average CRand and NMI below 0.65 and 0.94, respectively, and average CI above 23.75). Classical distance metrics are known to become more uniform as the feature-space dimension grows, and the number of local minima of the MSSC quickly increases, so feature-reduction or subspace-clustering techniques are often recommended for high-dimensional datasets. In these experiments, however, the inaccuracy of repeated K-means (or K-means++) appears to be a direct consequence of its inability to find good-quality local minima, rather than a consequence of the MSSC model itself, since near-optimal solutions of the MSSC translate into accurate results.

Overall, we conclude that even for simple Gaussian-based distribution mixtures, finding good local minima of the MSSC problem is essential for an accurate information retrieval. This is a major difference with studies on, for example, deep neural networks, where it is conjectured that most local minima have similar objective values, and where more intensive training (e.g., stochastic gradient descent with large batches) have adverse effects on generalization [30]. For clustering problems, it is essential to keep progressing toward faster and more accurate MSSC solvers, and to recognize when to use these high-performance methods for large and high-dimensional datasets.

5. Conclusions and perspectives

In this article, we have studied the MSSC problem, a classical clustering model of which the popular K-means algorithm constitutes a local minimizer. We have proposed a hybrid genetic algorithm, HG-means, that combines the improvement capabilities of K-means as a local search with the diversification capabilities of problem-tailored genetic operators. The algorithm uses an exact

minimum-cost matching crossover operator and an adaptive mutation procedure to generate strategic initial center positions for K-means and to promote a thorough exploration of the search space. Moreover, it uses population diversity management strategies to prevent premature convergence to shallow local minima.

We conducted extensive computational experiments to evaluate the performance of the method in terms of MSSC solution quality, computational effort and scalability, and external cluster validity. Our results indicate that HG-means attains better local minima than all recent state-of-the-art algorithms. Large solution improvements are usually observed for large datasets with a medium-to-high number of clusters m, since these characteristics lead to MSSC problems that have not been effectively solved by previous approaches. The CPU time of HG-means is directly proportional to that of the K-means local-improvement procedure and to the number of iterations allowed (the termination criterion). It appears to grow linearly with the number of samples and feature-space dimensions, and the termination criterion can be adjusted to achieve solutions in a shorter time without a large impact on solution accuracy.

Through additional tests conducted on datasets generated via Gaussian mixtures, we observed a strong correlation between MSSC solution quality and cluster validity measures. A repeated K-means algorithm, for example, obtains solution inaccuracies (percentage gap to the best known local minima) that are small at first glance but highly detrimental for the outcome of the clustering task. In particular, a gap as small as 5% in the objective space can make the difference between accurate clustering and failure. This effect was observed in all Gaussian datasets studied, and it became more prominent in feature spaces of higher dimension. In those situations, the inability of K-means or K-means++ to provide satisfactory results seems to be tied to its inability to find good-quality local minima of the MSSC model, rather than to an inadequacy of the model itself.

Overall, beyond the immediate gains in terms of clustering performance, research into efficient optimization algorithms for MSSC remains linked to important methodological stakes. Indeed, a number of studies aim to find adequate models (e.g., MSSC) for different tasks and datasets. With that goal in mind, it is essential to differentiate the limitations of the model themselves (inadequacy for a given task or data type), and those of algorithms used to solve such models (shallow local optima). While an analysis using external measures (e.g., CRand, NMI or CI) allows a general evaluation of error (due to both sources of inaccuracy), only a precise investigation of a method's performance in the objective space can help evaluating the magnitude of each imprecision, and only accurate or exact optimization methods can give meaningful conclusions regarding model suitability. In future research, we plan to continue progressing in this direction, generalizing the proposed solution method to other clustering models, possibly considering the use of kernel transformations, integrating semi-supervised information in the form of must-link or cannot-link constraints, and pursuing the development of high-performance optimization algorithms for other classes of applications.

Declarations of interest

None.

Acknowledgments

The authors thank the four anonymous referees for their detailed comments, which significantly contributed to improving this paper. This research is partially supported by CNPq [grant number 308498/2015-1] and FAPERJ [grant number E-26/203.310/2016] in Brazil. This support is gratefully acknowledged.

References

[1] K. Al-Sultan, A tabu search approach to the clustering problem, Pattern Recognit. 28 (9) (1995) 1443–1451.
[2] D. Aloise, A. Deshpande, P. Hansen, P. Popat, NP-hardness of euclidean sum-of-squares clustering, Mach. Learn. 75 (2) (2009) 245–248.
[3] D. Aloise, P. Hansen, L. Liberti, An improved column generation algorithm for minimum sum-of-squares clustering, Math. Program. 131 (1) (2012) 195–220.
[4] L.T.H. An, L.H. Minh, P.D. Tao, New and efficient DCA based algorithms for minimum sum-of-squares clustering, Pattern Recognit. 47 (1) (2014) 388–401.
[5] D. Arthur, S. Vassilvitskii, K-means++: the advantages of careful seeding, in: SODA'07. Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, New Orleans, Louisiana, USA, 2007, pp. 1027–1035.
[6] A.M. Bagirov, Modified global k-means algorithm for minimum sum-of-squares clustering problems, Pattern Recognit. 41 (10) (2008) 3192–3199.
[7] A.M. Bagirov, S. Taheri, J. Ugon., Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems, Pattern Recognit. 53 (2016) 12–24.
[8] A.M. Bagirov, J. Ugon, D. Webb, Fast modified global k-means algorithm for incremental cluster construction, Pattern Recognit. 44 (4) (2011) 866–876.
[9] C. Blum, J. Puchinger, G. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft Comput. 11 (6) (2011) 4135–4151.
[10] S. Dasgupta, Learning mixtures of gaussians, in: 40th Annual Symposium on Foundations of Computer Science, 1, 1999, pp. 634–644.
[11] P. Fränti, J. Kivijärvi, T. Kaukoranta, O. Nevalainen, Genetic algorithms for large-scale clustering problems, Comput. J. 40 (9) (1997) 547–554.
[12] P. Fränti, M. Rezaei, Q. Zhao, Centroid index: cluster level similarity measure, Pattern Recognit. 47 (9) (2014) 3034–3045.
[13] G. Hamerly, Making k-means even faster, in: SDM'10, SIAM International Conference on Data Mining, 2010, pp. 130–140.
[14] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, 3rd ed., Morgan Kaufmann., 2011.
[15] P. Hansen, N. Mladenović, J-means: a new local search heuristic for minimum sum of squares clustering, Pattern Recognit. 34 (2) (2001) 405–413.
[16] J.A. Hartigan, M.A. Wong, Algorithm AS 136: a k-means clustering algorithm, Appl. Stat. 28 (1) (1979) 100–108.
[17] E. Hoffer, I. Hubara, D. Soudry, Train longer, generalize better: closing the generalization gap in large batch training of neural networks, Adv. Neural Inf. Process. Syst. (2017) 1729–1739.
[18] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI, 1975.
[19] E.R. Hruschka, R.J.G.B. Campello, A.A. Freitas, A.C.P.L.F. de Carvalho, A survey of evolutionary algorithms for clustering, IEEE Trans. Syst. Man Cybern.Part C 39 (2) (2009) 133–155.
[20] L. Hubert, P. Arabie, Comparing partitions, J. Classification 2 (1) (1985) 193–218.
[21] H. Ismkhan, I-k-means-+: an iterative clustering algorithm based on an enhanced version of the k-means, Pattern Recognit. 79 (1) (2018) 402–413.
[22] A.K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[23] N. Karmitsa, A.M. Bagirov, S. Taheri, New diagonal bundle method for clustering problems in large data sets, Eur. J. Oper. Res. 263 (2) (2017) 367–379.
[24] N. Karmitsa, A.M. Bagirov, S. Taheri, Clustering in large data sets with the limited memory bundle method, Pattern Recognit. 83 (2018) 245–259.
[25] N.S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P.T.P. Tang, On large-batch training for deep learning: generalization gap and sharp minima, in: ICLR'17, International Conference on Learning Representations, 2017.
[26] J. Kivijärvi, P. Fränti, O. Nevalainen, Self-adaptive genetic algorithm for clustering, J. Heuristics 9 (2) (2003) 113–129.
[27] K. Krishna, M.N. Murty, Genetic k-means algorithm, IEEE Trans. Syst. Man Cybern. Part B 29 (3) (1999) 433–439.
[28] H.W. Kuhn, The hungarian method for the assignment problem, Nav. Res. Logist. 2 (1–2) (1955) 83–97.
[29] T.O. Kvalseth, Entropy and correlation: some comments, IEEE Trans. Syst. Man Cybern. 17 (3) (1987) 517–519.
[30] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[31] A. Likas, N. Vlassis, J.J. Verbeek, The global k-means clustering algorithm, Pattern Recognit. 36 (2) (2003) 451–461.
[32] S.P. Lloyd, Least squares quantization in PCM, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137.
[33] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, S.J. Brown, FGKA: a fast genetic k-means clustering algorithm, in: Proceedings of the 2004 ACM Symposium on Applied Computing, 2004, pp. 622–623.
[34] U. Maulik, S. Bandyopadhyay, Genetic algorithm-based clustering technique, Pattern Recognit. 33 (2000) 1455–1465.
[35] P. Merz, A. Zell, Clustering gene expression profiles with memetic algorithms, in: Parallel Problem Solving from Nature, Springer, 2002, pp. 811–820.
[36] B. Ordin, A.M. Bagirov, A heuristic algorithm for solving the minimum sum-of-squares clustering problems, J. Global Optim. 61 (2) (2015) 341–361.
[37] M. Sarkar, B. Yegnanarayana, D. Khemani, A clustering algorithm using an evolutionary programming-based approach, Pattern Recognit. Lett. 18 (10) (1997) 975–986.

[38] P. Scheunders, A comparison of clustering algorithms applied to color image quantization, Pattern Recognit. Lett. 18 (1997) 1379–1384.
[39] S.Z. Selim, K. Alsultan, A simulated annealing algorithm for the clustering problem, Pattern Recognit. 24 (10) (1991) 1003–1008.
[40] K. Sörensen, M. Sevaux, MAPM: Memetic algorithms with population management, Comput. Oper. Res. 33 (5) (2006) 1214–1225.
[41] D. Steinley, K-means clustering: a half-century synthesis, Br. J. Math. Stat.Psychol. 59 (1) (2006) 1–34.
[42] C.A. Sugar, G.M. James, Finding the number of clusters in a dataset, J. Am. Stat. Assoc. 98 (463) (2003) 750–763.
[43] T. Vidal, T.G. Crainic, M. Gendreau, N. Lahrichi, W. Rei, A hybrid genetic algorithm for multidepot and periodic vehicle routing problems, Oper. Res. 60 (3) (2012) 611–624.

Daniel Gribel is currently pursuing a PhD degree in the field of optimization and automated reasoning, in the Department of Computer Science at the Pontifical Catholic University of Rio de Janeiro, Brazil. He is active in a variety of academic and industrial projects. His current research focuses on global optimization techniques applied to data mining problems, such as clustering, classification, regression and detection of patterns.

Thibaut Vidal is a professor in the Department of Computer Science at the Pontifical Catholic University of Rio de Janeiro, Brazil. Previously, he was postdoctoral researcher at the Laboratory for Information and Decision Systems at MIT. His research interests include combinatorial optimization, integer and convex programming, with applications to machine learning, signal processing, resource allocation and logistics problems.