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

Daniel Gribel and **Thibaut Vidal**

Department of Computer Science,
Pontifical Catholic University of Rio de Janeiro, Brazil

December 20^{th}, 2018

Data Clustering

- Most classical clustering algorithms are *ad-hoc* methods, based on iterative aggregations, splits, or simple search steps (e.g., K-means, DBSCAN, BIRCH etc...).
  - Many algorithms can be interpreted as local searches for some mathematical optimization problems, but their *optimization performance* (quality of the local minima for the model at hand) is rarely discussed.
  - Performance of clustering algorithms usually measured relatively to some cluster validity indices (e.g., CRand or NMI) based on a ground truth that we *wish* to recover

  $\Rightarrow$ Difficult to separate the two main sources of errors:
  1) inadequate choice of clustering model for the task at hand,
  2) inadequate solution algorithm for the model at hand.

Data Clustering

- Little incentive to go *beyond* the classical algorithms for machine learning for practitioners. Here are some reasons:
  1. Data size and computational time restrictions
  2. Code simplicity and availability
  3. The belief that better optimization solutions have only minor impact on classification performance
- Breaking through this standstill:
  1. New *efficient*, *scalable* and *open-source* optimization algorithms
  2. Highlighting the correlation between optimization and classification performance

Our first focus – Minimum sum-of-squares clustering

- MSSC: minimization of the squared Euclidean distances of objects to their cluster means (minimization of within-group sum-of-squares).
- Given a set $P = \{p_1, \ldots, p_n\}$ of n samples in $\mathbb{R}^d$.
- Return a set of centers $\{y_1, y_2, ..., y_k\}$ in $\mathbb{R}^d$.

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

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

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

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

Two important properties of the problem

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 assigned to it.

Property (2)

In any optimal MSSC solution, each sample p_i is assigned to its closest center.

Proposed Methodology

- Combination of genetic algorithm (GA) with local search, and a few problem-specific tricks:
  - population-diversity management
  - elimination of clones
  - specialized crossover based on a bipartite-matching procedure
  - adaptive mutation to avoid excessive attraction towards outliers
- Local search is simply operated by running the K-means algorithm, taking the candidate solution generated by the crossover as a starting point.

Proposed Methodology

Algorithm 1 HG-MEANS – general structure

1: Initialize population with Π_{MAX} individuals/solutions
2: **while** (number of iterations without improvement $< N_1$) $\wedge$ (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: **end if**
11: **end while**
12: Return best solution

Proposed Methodology

Crossover

Crossover procedure applied to two parent solutions P_1 and P_2 in order to produce a (child) solution:

1. **Centroids matching.** Solve bipartite matching problem based on the centroids of P_1 and P_2.
2. **Selection**. For each pair of centroids, inherit one randomly into the offspring.
3. **Assignment**. Re-assign data points to the closest offspring centroid.

Proposed Methodology

Crossover

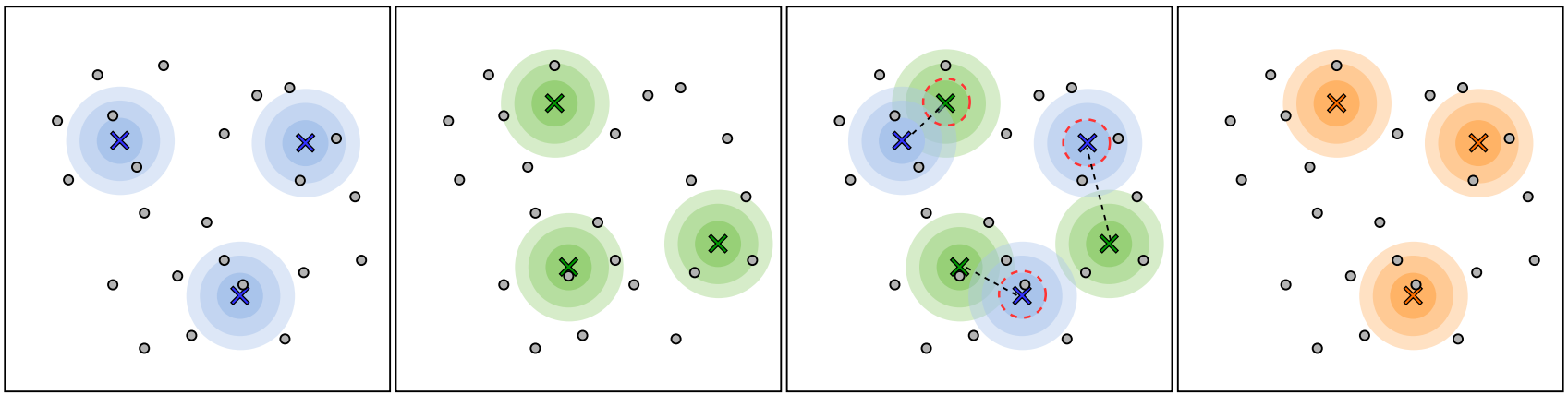


Figure 1: Crossover based on centroids matching: (a) Parent $P1$; (b)Parent $P2$; (c) The assignment between centroids of P_1 and P_2, and random selection (d) The resulting offspring

Proposed Methodology

Mutation

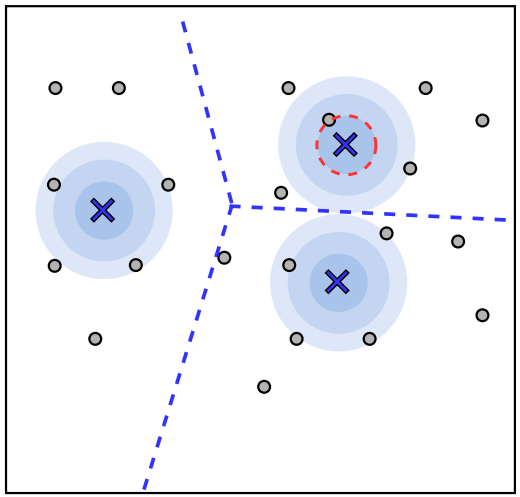
1. Randomly select a centroid c^* and remove it from the solution.
2. Re-assign the samples to their closest center.
3. Randomly select a data point x_u and re-insert c^* in the position of x_u. The probability to select x_j as the new centroid is

$$P(x_j) = \left((1 - \alpha_{C'}) \times \frac{1}{n} \right) + \left(\alpha_{C'} \times \frac{d(x_j, C(x_j))}{\sum_{i=1}^{n} d(x_i, C(x_i))} \right),$$

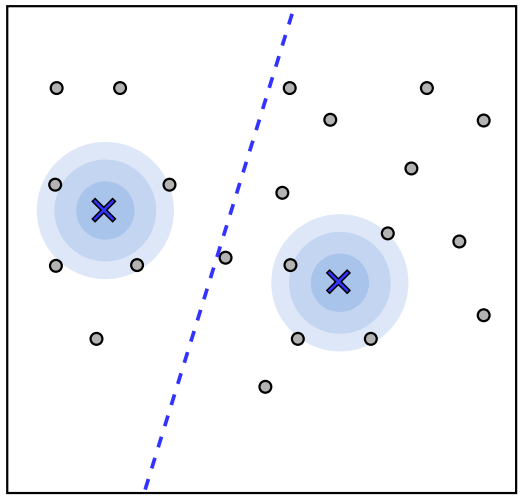
where $\alpha_{C'}$ is the mutation parameter to control the impact of outliers. This parameter evolves along with the genetic material of the solutions through dedicated mutation and crossover operations.

Proposed Methodology

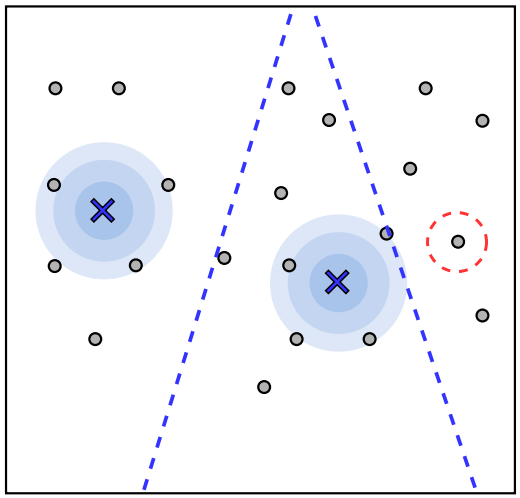
(a) Removal of a random center



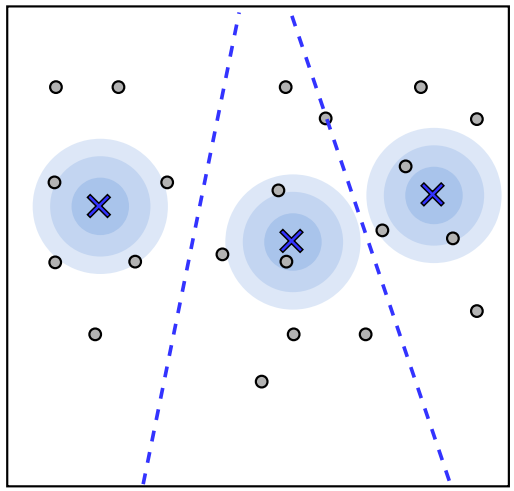
(b) Re-assignment of the samples to their nearest center



(c) Randomized reinsertion of center, biased by sample-to-center distances



(d) Final solution (after local search)



Proposed Methodology

Survivors selection

- Selects the best individuals to propagate when the maximum population size Π_{max} is reached, determining the Π_{min} individuals that will go on to the next generation, by discarding λ individuals ($\lambda = \Pi_{max} - \Pi_{min}$)
- Individuals selected for removal:
  - Clones (identical to any other solution)
  - Bad solution quality

Computational Experiments and Analysis

Experiments focused around three main goals:

- Performance on the MSSC Optimization Problem
- Computational time and Scalability
- **Correlation between Optimization Performance and Classification Performance**

Computational Experiments and Analysis

In the result tables,

- n is the number of samples;
- m is the number of clusters;
- d is the number of features (data dimensionality);
- Gap is the error from the best known solution, calculated as:

$$GAP = \frac{f - f_{best}}{f_{best}} \times 100$$

where f is the value of the MSSC objective found by any previous algorithm and f_{best} is the best known value;

Computational Experiments and Analysis

Instances

Group	Dataset	n	d	$n \times d$	Clusters
A1	German Towns	59	2	118	$m \in \{2, 3, 4...$
	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	
	Heart Disease	297	13	4k	
	Breast Cancer	683	9	6k	$m \in \{2, 5, 10, 15...$
	Pima Indians Diabetes	768	8	6k	$20, 25, 30, 40, 50\}$
	Congressional Voting	435	16	7k	
	Ionosphere	351	34	12k	
B	TSPLib1060	1,060	2	2k	
	TSPLib3038	3,038	2	6k	
	Image Segmentation	2,310	19	44k	$m \in \{2, 10, 20, 30...$
	Page Blocks	5,473	10	55k	$40, 50, 60, 80, 100\}$
	Pendigit	10,992	16	176k	
	Letters	20,000	16	320k	

Table 1: Small to Medium datasets used for performance comparisons on the MSSC optimization problem

Computational Experiments and Analysis

Instances

Group	Dataset	n	d	$n \times d$	Clusters
C	D15112	15,112	2	30k	$m \in \{2, 3, 5, 10 \ldots 15, 20, 25\}$
	Pla85900	85,900	2	172k	
	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	

Table 2: Large datasets used for performance comparisons on the MSSC optimization problem

Computational Experiments and Analysis

Parameters

- Π_{min}: Population size
- Π_{max}: Maximum size of population
- I_{max}: Maximum number of iterations

Configuration	Π_{min}	Π_{max}	I_{max}	$Time(s)$	Gap
Standard	10	20	5000	1060.48	-0.35
Fast	5	10	500	127.48	0.16

Table 3: Fast and Standard configurations of HG-means

Computational Experiments and Analysis

Performance on the MSSC Optimization Problem

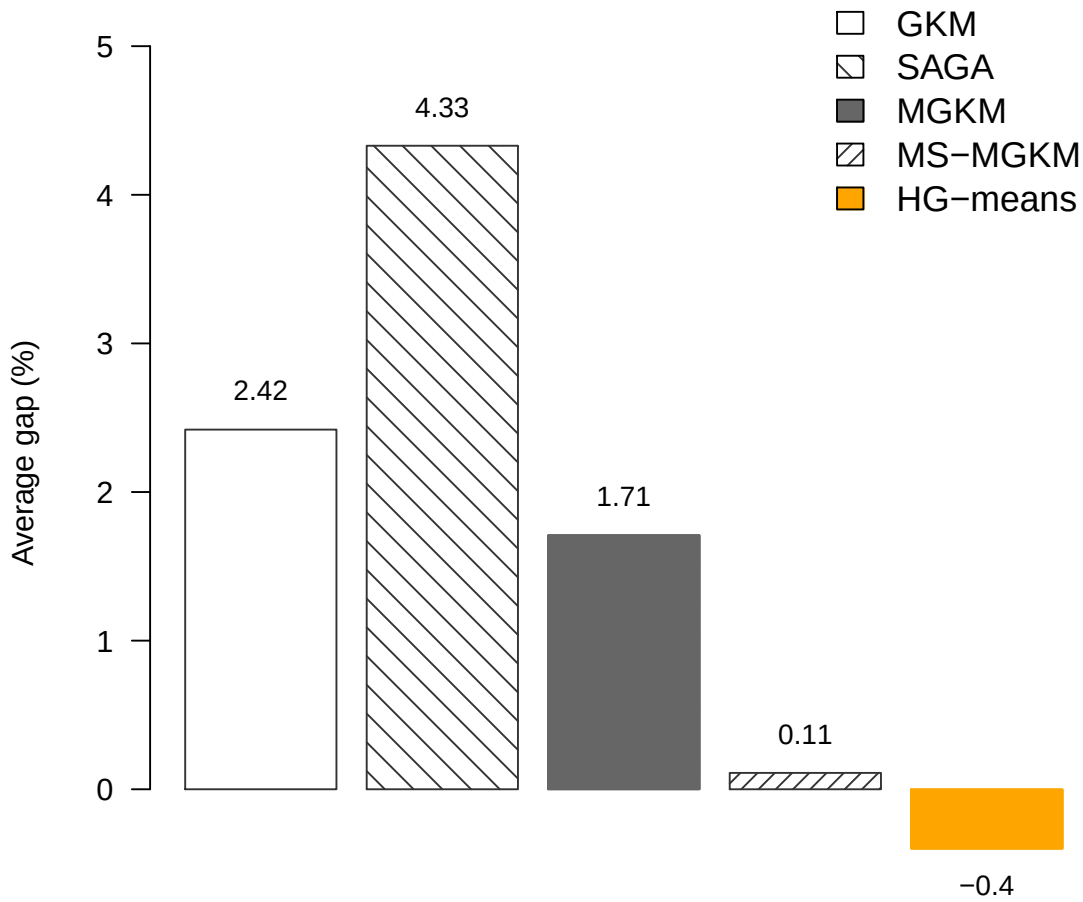


Figure 2: Average gap from the best known solution for UCI Small to Medium datasets

Computational Experiments and Analysis

Performance on the MSSC Optimization Problem

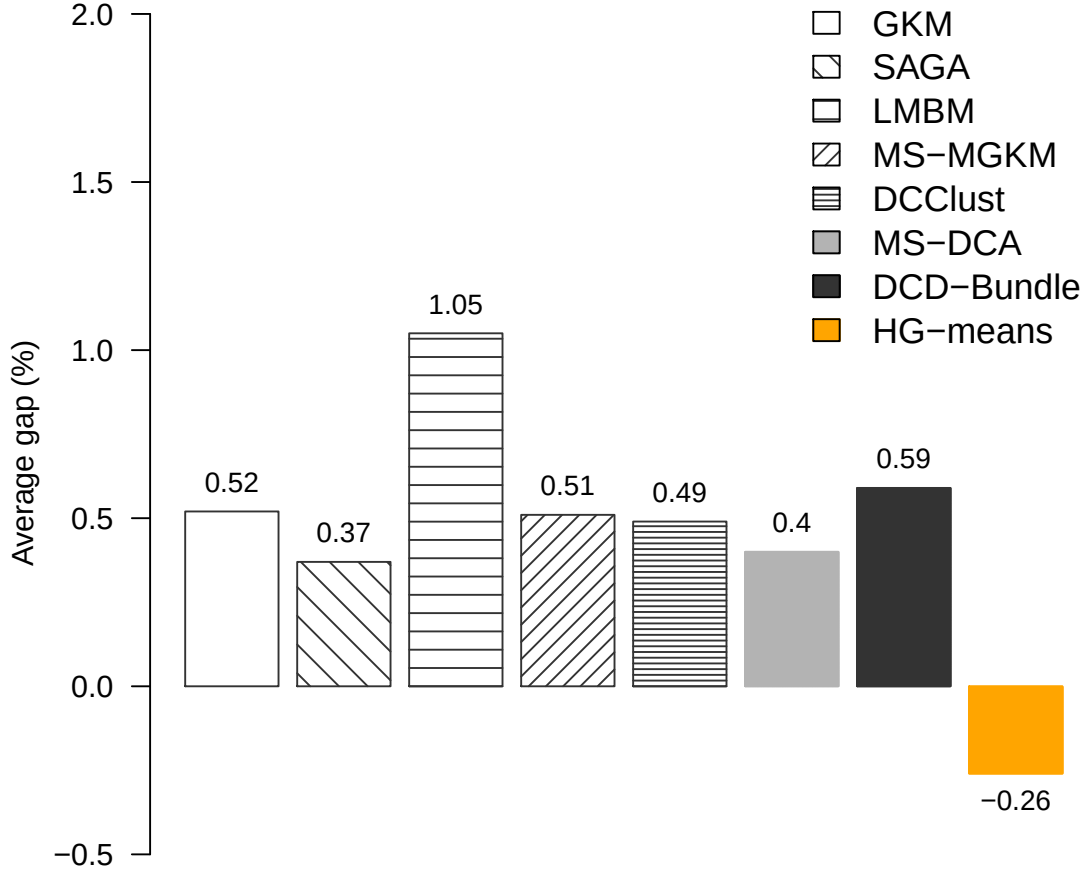


Figure 3: Average gap from the best known solution for UCI Large datasets

Computational Experiments and Analysis

Computational time on largest datasets

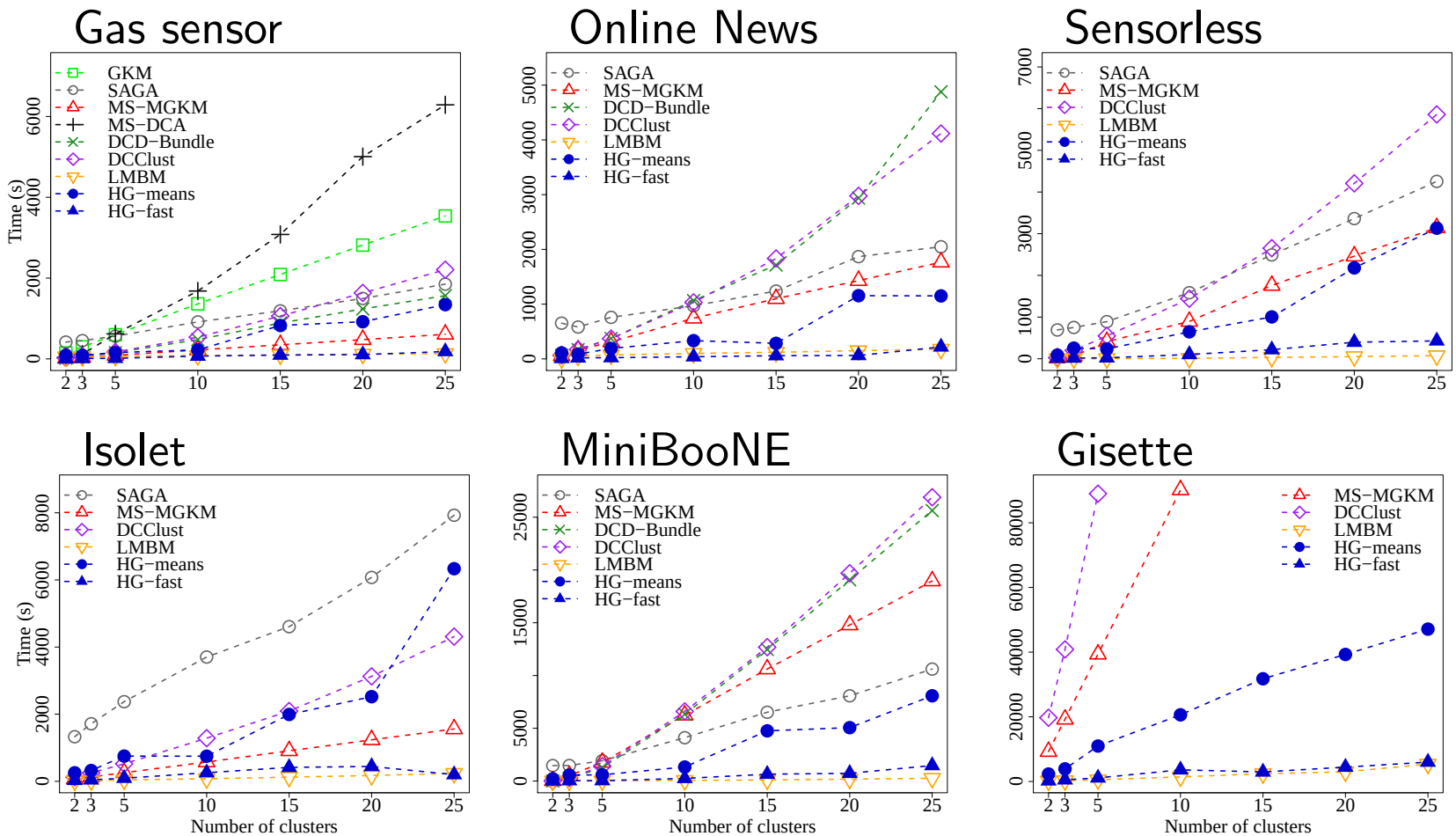


Figure 4: CPU time of state-of-the-art algorithms on UCI large-scale datasets

Computational Experiments and Analysis

Solution Quality and Classification Performance

- Experimental setting to measure the ability of HG-MEANS, K-MEANS and K-MEANS++ to classify 50,000 samples issued from a mixture of spherical Gaussian distributions:

$$X \sim {}^1\!/_m \sum_{i=1}^{m} \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \text{ with } \boldsymbol{\Sigma}_i = \sigma_i^2 \mathbf{I}$$

.

- For each $i \in \{1, \ldots, m\}$, μ_i and σ_i^2 are uniformly selected in $[0, 5]$ and $[1, 10]$, respectively.
- Generated to be hardly separable.
- Fundamental setting: no hidden structure, a lot of independent information.

Computational Experiments and Analysis

m	d	BKS Objective Value	Gap (%) K-means 1 Run	Gap (%) K-means 500 Runs	Gap (%) K-means++ 1 Run	Gap (%) K-means++ 500 Runs	Gap (%) HG-means	Time (s) K-means 1 Run	Time (s) K-means 500 Runs	Time (s) K-means++ 1 Run	Time (s) K-means++ 500 Runs	Time (s) HG-means
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 4: Mixture of spherical Gaussian distributions – Solution quality

Computational Experiments and Analysis

		CRand					NMI					CI				
		K-means		K-means++		HG-means	K-means		K-means++		HG-means	K-means		K-means++		HG-means
m	d	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**

Table 5: Mixture of spherical Gaussian distributions – Clustering performance

Conclusions and Future work

- Possible to design efficient and scalable algorithms for MSSC which outperform by far the existing ones.
- **Optimization performance matters**, and directly influences classification performance, especially for **high-dimensional** datasets.
- Need to pursue a **disciplined study** of machine learning algorithms, to be able to track error causes: **model inadequacy for the data at hand** or **low-quality local minima** for the model at hand.

Thank you

THANK YOU FOR YOUR ATTENTION !

Further reading:
"Gribel, D., & Vidal, T. (2019). HG-means: A scalable hybrid metaheuristic for minimum sum-of-squares clustering. *Pattern Recognition, Articles in Advance.*
`https://arxiv.org/pdf/1804.09813.pdf`

Source code in C++ (and soon Python) available at:
`https://github.com/danielgribel/hg-means`
`https://w1.cirrelt.ca/~vidalt/`