Combinatorial Optimization and Machine Learning – Part I

Thibaut Vidal[1], Thiago Serra[2]

[1] Departamento de Informática, PUC-Rio
vidalt@inf.puc-rio.br
[2] Freeman College of Management, Bucknell University
thiago.serra@bucknell.edu

Image Credit: Hitchhikers Guide to the Galaxy

Content

Combinatorial Optimization in Machine Learning

- Combinatorial problems occur in many situations in the machine learning domain. For example, when
  - Choosing subsets of training samples or dimensions (e.g., dimension reduction, outliers detection, sparse models)
  - Training non-linear models (e.g., ReLU neural networks)
  - Aggregating elements (e.g., clustering or community detection)
  - Searching in high-dimensional spaces (e.g., adversarial examples or meta-parameter tuning)...

Combinatorial Optimization in Machine Learning

- Though, ML and OR application contexts are quite different:
  - Large data set $\Rightarrow$ Large combinatorial problems
  - Training data is only a glimpse of the true distribution, but performance should generalize
  - No consensus on an ideal objective or model for many tasks of interest
  - Computational requirements can be constraining (e.g., limited time or processor capacity in on-line applications or embarked systems)
  - Data may change over time
- Many ML algorithms are **heuristics** for some mathematical optimization problem.

Content

Disciplined Evaluation of ML algorithms

- Many algorithms used in ML are in fact **heuristics** for some mathematical optimization problem, but their *optimization performance* (quality of the local minima in the objective space) is not always investigated.
- This represents a challenge for scientific evaluation of methods, as it becomes difficult to discern two main sources of errors:
  A) inadequate solution algorithm for the model at hand,
  B) inadequate choice of model for the task at hand.
- Experimental analyses relying on task-based performance metrics (e.g., accuracy, F1 score, NMI) only measure an aggregate performance which includes both error sources.

Disciplined Evaluation of ML algorithms

- Separating these errors requires:
  - A) The development of state-of-the-art optimization algorithms (ideally, exact methods) with performance evaluations in the objective space.
  - B) The use of state-of-the-art optimization algorithms with a good performance record to assess model suitability for a given task.

Content

Content

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 $\mathbb{R}^d$.
- For optimal solution algorithms, see [1]

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

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

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

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

Two important properties

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.

Content

Methodology

- Combination of genetic algorithm (GA) with local search [10] with additional strategies:
  - 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 operated by running the K-means algorithm, taking the candidate solution generated by the crossover as a starting point.

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

Methodology

Crossover

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

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

Methodology

Crossover

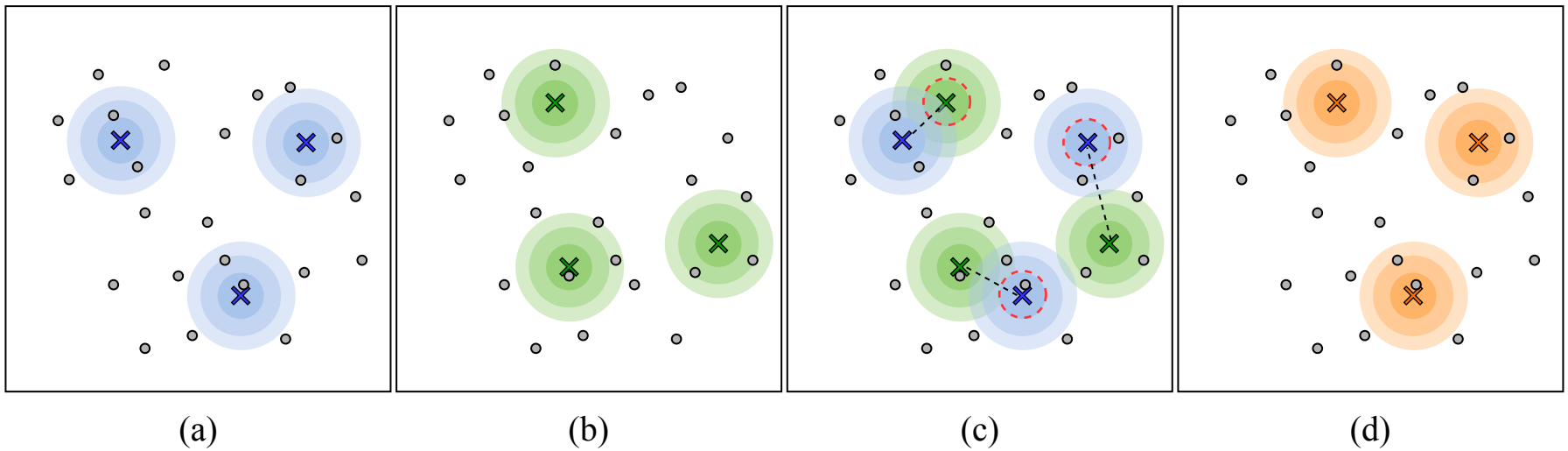


Figure 1: Crossover based on centroids matching: (a) Parent $P1$; (b)Parent $P2$; (c) The assignment between centroids of P_1 and P_2, and random selection (d) The resulting offspring

Methodology

Mutation

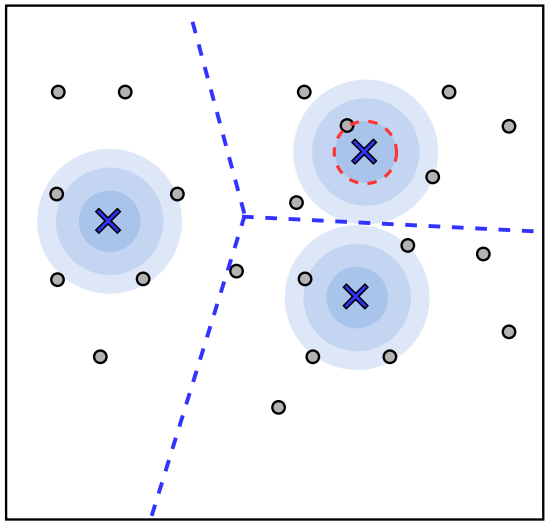
- Randomly select a centroid c^* and remove it from the solution.
- Re-assign the samples to their closest center.
- Randomly select a data point x_u and re-insert c^* in the position of x_u. The probability to select x_j as the new centroid is

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

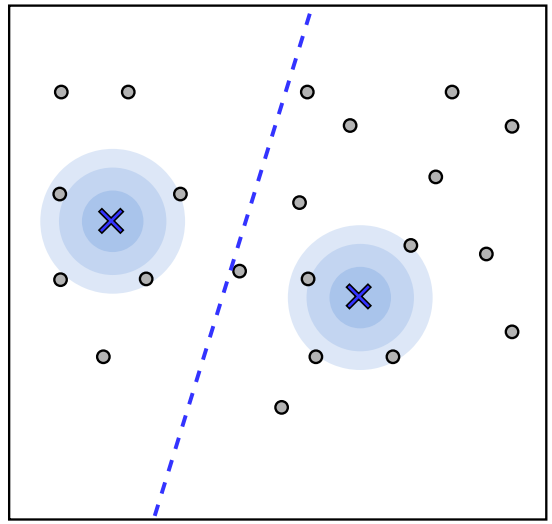
where $\alpha_{C'}$ is the mutation parameter to control the impact of outliers. This parameter evolves along with the genetic material of the solutions through dedicated mutation and crossover operations.

Methodology

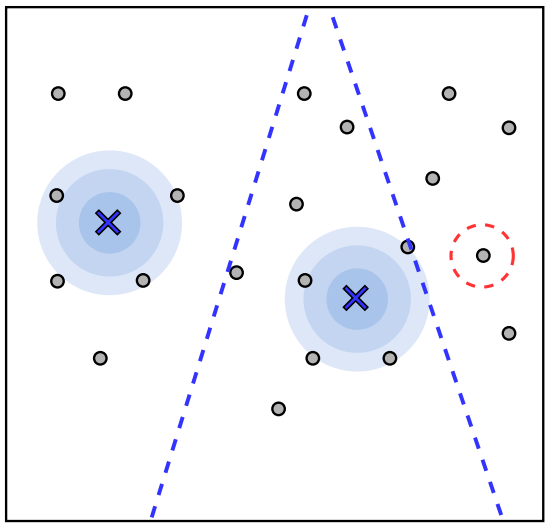
(a) Removal of a random center



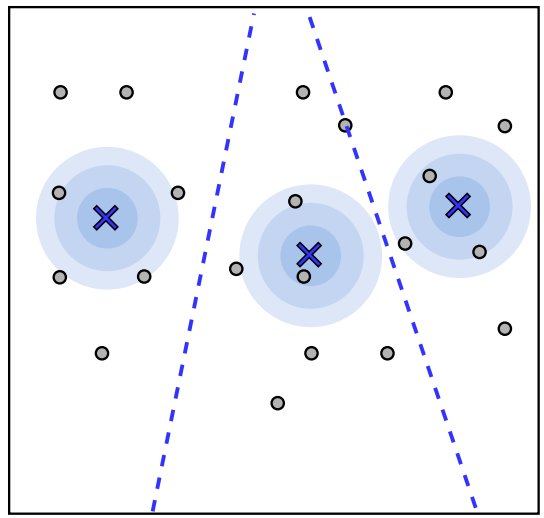
(b) Re-assignment of the samples to their nearest center



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



(d) Final solution (after local search)



Methodology

Local improvement

One run of the K-means algorithm. 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.

$$cluster(x_i) = \min_j d(x_i, c_j), \quad j = 1, ..., m \tag{5}$$

2. **Update step**. Locate the new centroids c_j at the location of the barycenters of the clusters.

$$c_j = \frac{\sum_{x \in S_j} x}{|S_j|}, \quad j = 1, ..., m \tag{6}$$

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

Content

Computational Experiments and Analysis

Experiments focused around three main goals:

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

Computational Experiments and Analysis

In the 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;

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

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

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

Performance on the MSSC Optimization Problem

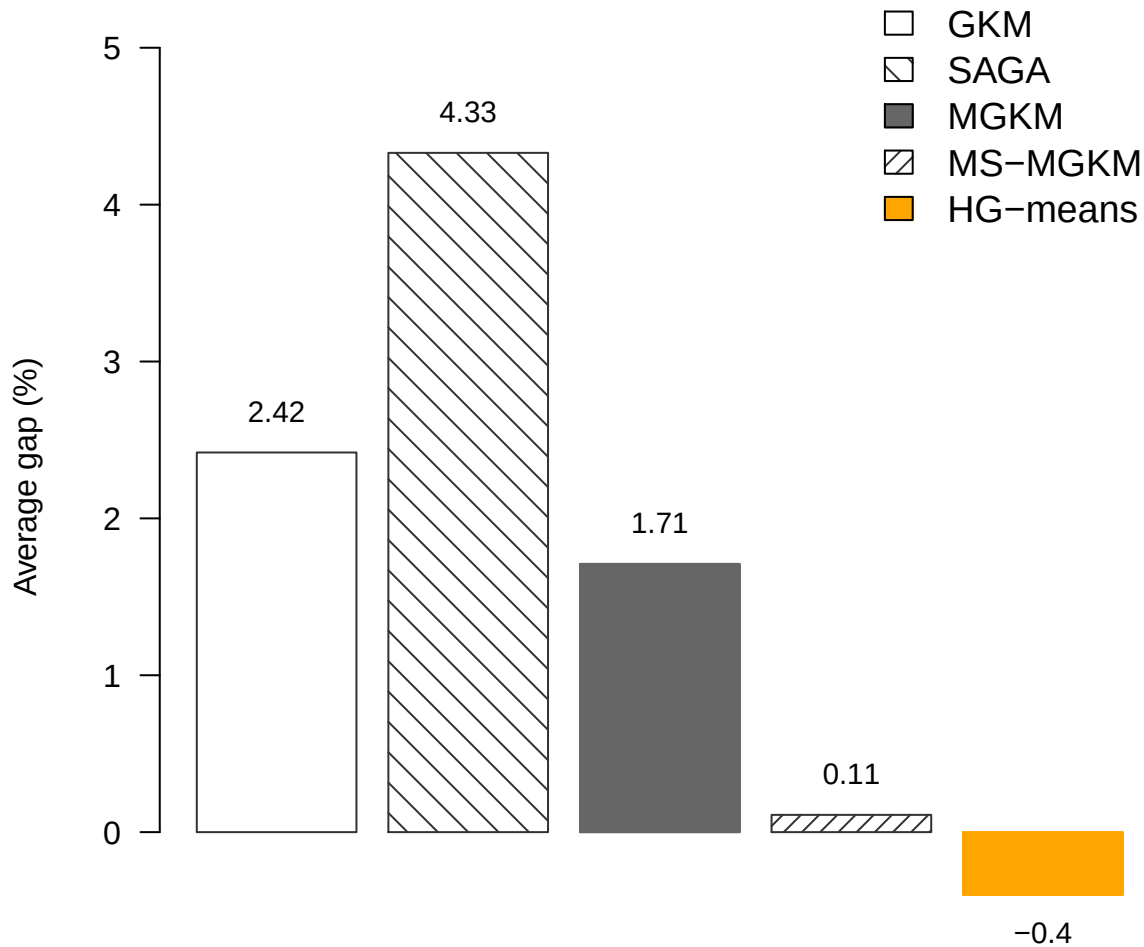


Figure 2: Average gap from the best known solution for UCI Small to Medium datasets

Performance on the MSSC Optimization Problem

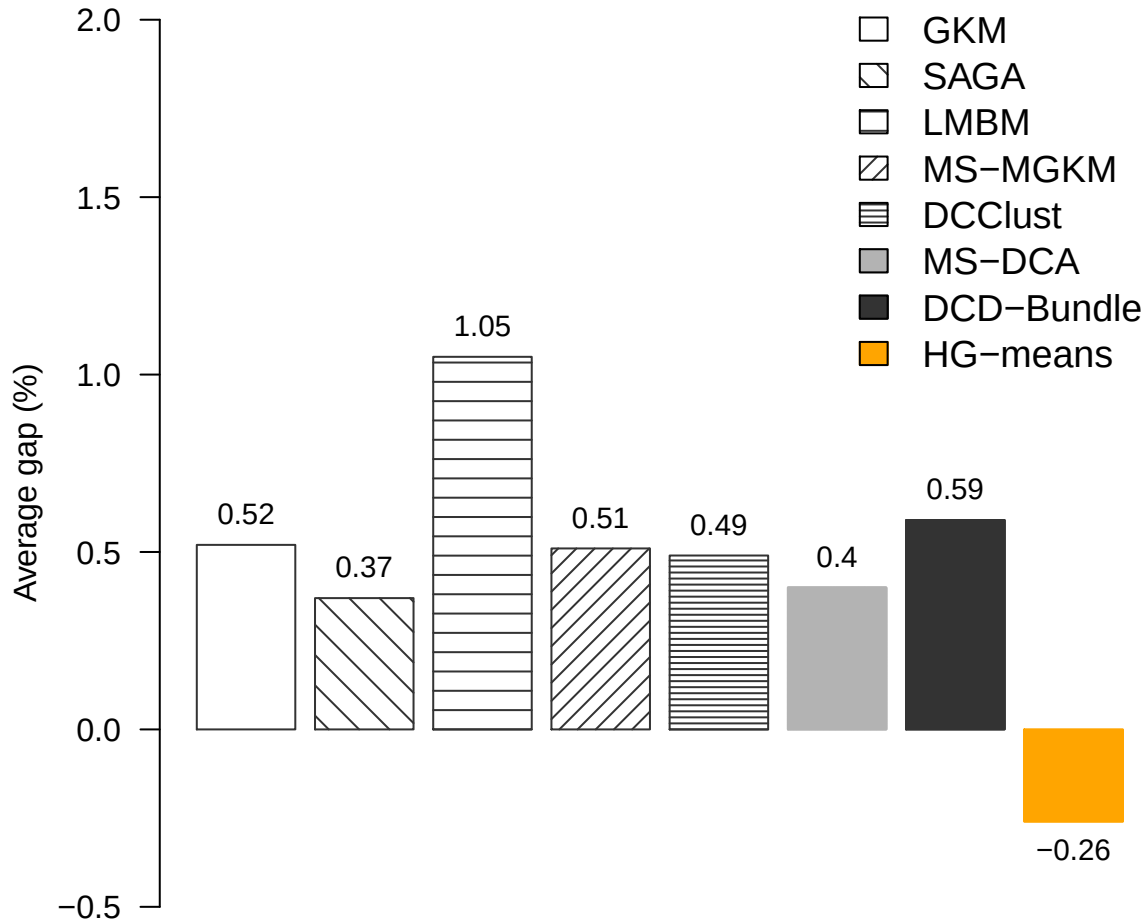


Figure 3: Average gap from the best known solution for UCI Large datasets

Computational time on largest datasets

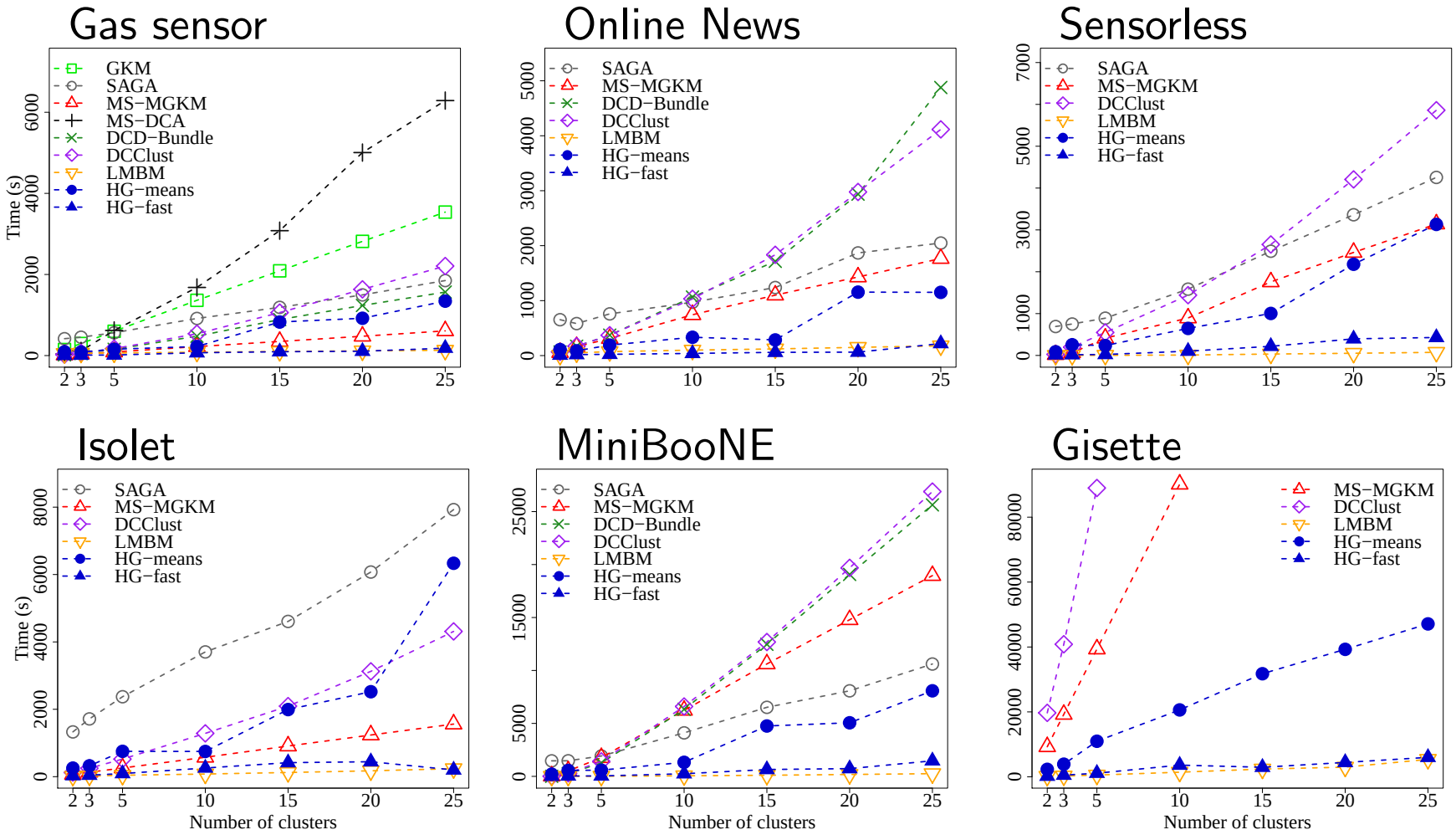


Figure 4: CPU time of state-of-the-art algorithms on UCI large-scale datasets

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

Discussions and Perspectives

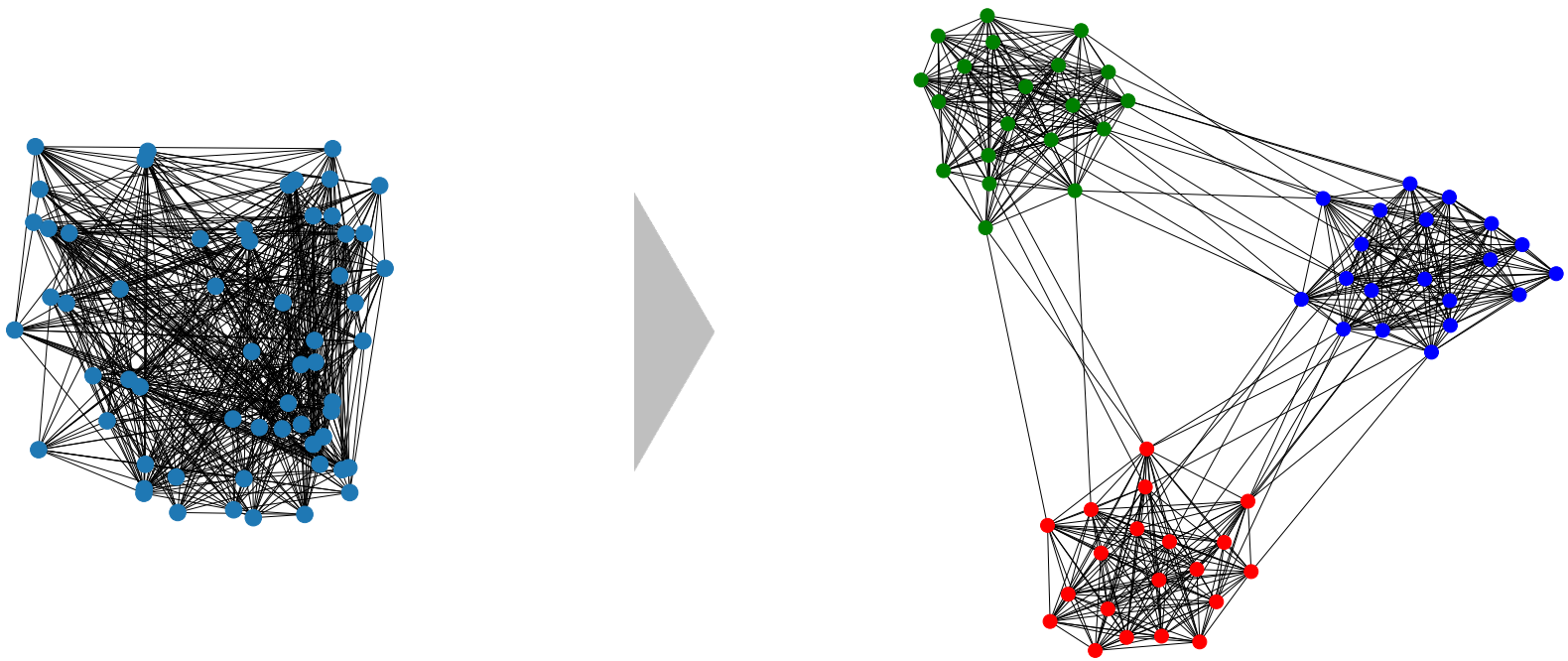
- Possible to design efficient and scalable algorithms for MSSC which outperform by far the existing ones.
- **Optimization performance matters**: directly influences classification performance, especially for **high-dimensional** datasets.

Content

Content

SBMs for community detection

- Optimal solutions of Community Detection in the Stochastic Block Model using mixed integer programming



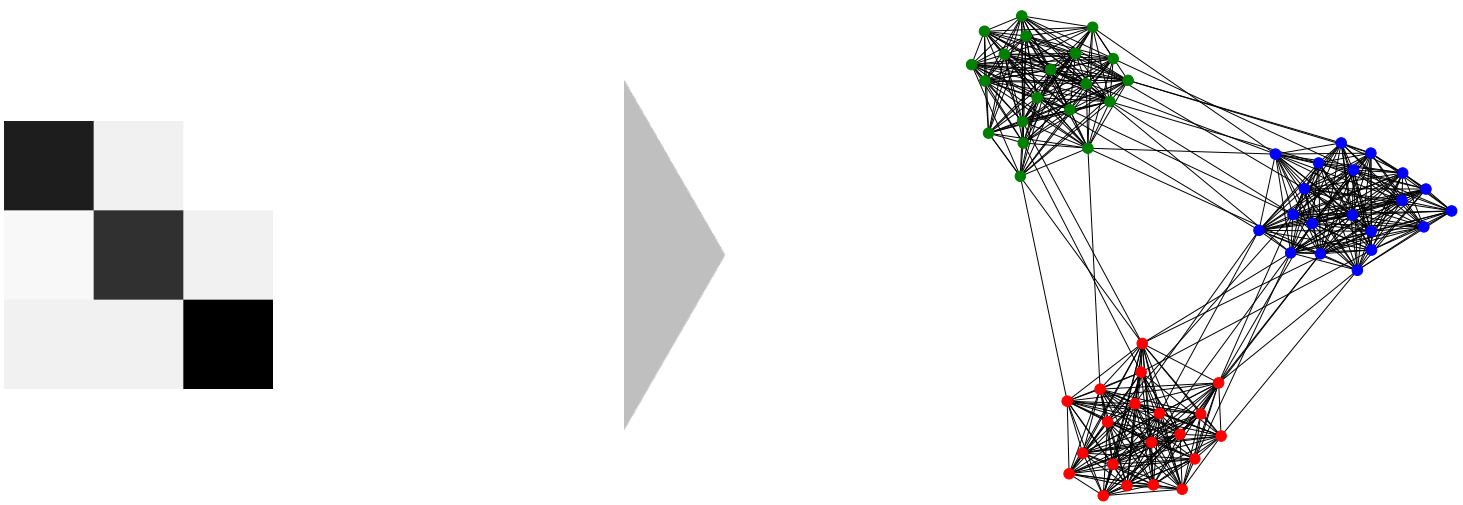
Stochastic Block Model (SBM)

A simple model for generating random graphs with community structure

$$G \sim SBM(\boldsymbol{g}, \boldsymbol{\Omega})$$

$\boldsymbol{g} \in \{1, \ldots, K\}^n$: group membership vector

$\boldsymbol{\Omega} \in \mathbb{R}^{K \times K}$: connectivity matrix



$A_{ij} \sim \text{Bernoulli}(\omega_{g_i g_j})$ for simple graphs

$A_{ij} \sim \text{Pois}(\omega_{g_i g_j})$ for multi-graphs

Different types of community structure

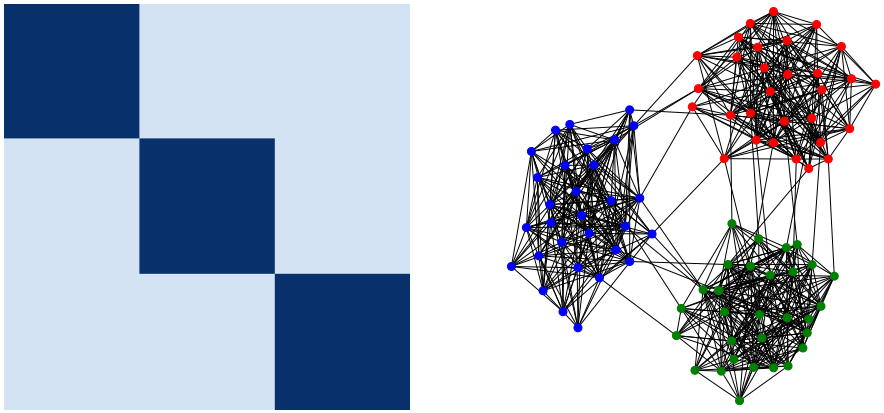


Figure 5: Assortative structure

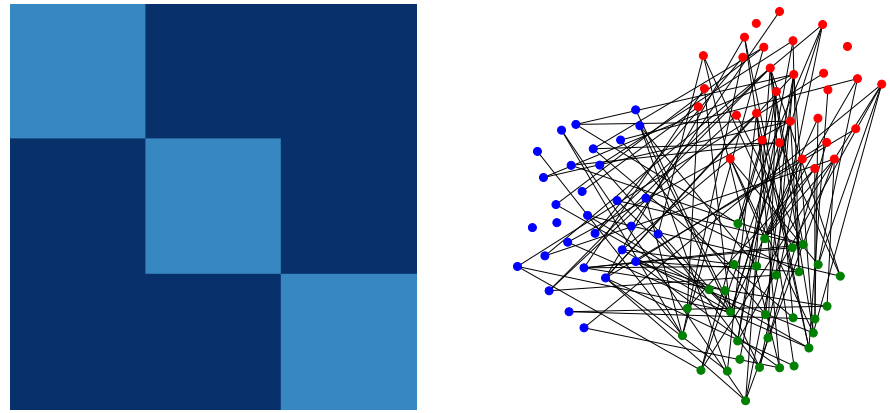


Figure 6: Disassortative structure

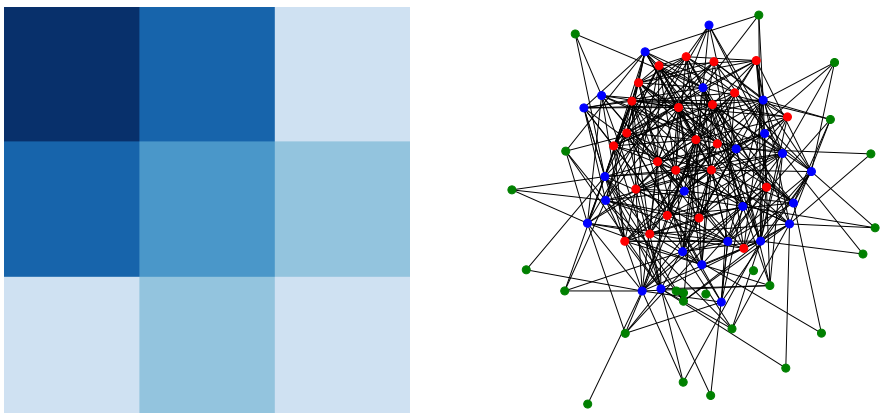


Figure 7: Core-periphery structure

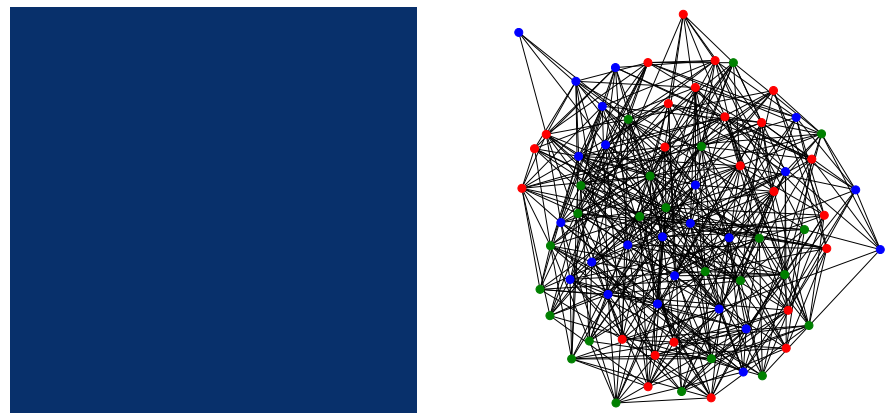


Figure 8: Random graph

Community detection with the DCSBM

The DCSBM is an extension of the SBM that allows the generation of networks with arbitrary degree distributions, by introducing parameters θ_i that control the expected degree of vertex i: $G \sim SBM(\boldsymbol{g}, \boldsymbol{\Omega}, \boldsymbol{\theta})$.

When used for community detection, the goal is to find the parameter values that best fit the observed graph G, by Maximum Likelihood Estimation (MLE):

$$P(G|\boldsymbol{\theta}, \boldsymbol{\Omega}, \boldsymbol{g}) = \prod_{i<j} \frac{(\theta_i \theta_j \omega_{g_i g_j})^{A_{ij}}}{A_{ij}!} \exp\left(-\theta_i \theta_j \omega_{g_i g_j}\right) \times \prod_i \frac{\left(\frac{1}{2}\theta_i^2 \omega_{g_i g_i}\right)^{A_{ii}/2}}{\left(\frac{1}{2}A_{ii}\right)!} \exp\left(-\tfrac{1}{2}\theta_i^2 \omega_{g_i g_i}\right)$$

We consider the special case where $\theta_i \theta_j = \frac{k_i k_j}{2m}$. After removing constant terms, the log-likelihood objective simplifies to:

$$\log P(G|\boldsymbol{\Omega}, \boldsymbol{g}) = \frac{1}{2} \sum_{i,j} \left(A_{ij} \log \omega_{g_i g_j} - \frac{k_i k_j}{2m} \omega_{g_i g_j} \right)$$

Descriptive formulation (MINLP)

- $\mathcal{C} = \{1, \ldots, K\}$ is the set of possible communities
- $z_{ir} = 1$ iff vertex $i \in V$ is assigned to group $r \in \mathcal{C}$ and 0 otherwise
- $\omega_{rs} \in \mathbb{R}^+$ represents the expected number of edges between two nodes belonging to communities r and s

$$
\begin{aligned}
\underset{\mathbf{Z},\boldsymbol{\Omega}}{\text{minimize}} \quad & \frac{1}{2} \sum_{i,j}^{n} \sum_{r,s}^{K} f_{ij}(\omega_{rs})\, z_{ir} z_{js} \\
\text{subject to} \quad & \sum_{r=1}^{K} z_{ir} = 1, && \forall i \in V \\
& z_{ir} \in \{0,1\}, && \forall i \in V, r \in \mathcal{C} \\
& \omega_{rs} \in \mathbb{R}^+, && \forall r, s \in \mathcal{C} \\
\text{where} \quad & f_{ij}(\omega_{rs}) = -A_{ij} \log \omega_{rs} + \tfrac{k_i k_j}{2m} \omega_{rs}
\end{aligned}
$$

The model can be solved by MINLP solvers such as `Couenne`.

Mixed Integer Linear Programming (MILP) formulation

This problem can be reformulated as a MILP by piecewise outer-approximation and linearization [6]:

$$
\begin{aligned}
\underset{\mathbf{Z},\boldsymbol{\Omega},\mathbf{Y},\mathbf{X}}{\text{minimize}} \quad & \frac{1}{2}\sum_{i,j}^{n}\sum_{r,s}^{K} x_{ijrs} \\
\text{subject to} \quad & x_{ijrs} \geq a_{ij}\omega_{rs} + b_{ij} - \overline{M_{ijrs}}(1 - y_{ijrs}), && \forall i,j \in V, \forall r,s \in \mathcal{C}, \forall \widetilde{\omega} \in \mathbb{R}^{+} \\
& x_{ijrs} \leq \overline{M_{ijrs}}\, y_{ijrs} \\
& x_{ijrs} \geq \underline{M_{ijrs}}\, y_{ijrs} \\
& z_{ir} - y_{ijrs} \geq 0 && \forall i,j \in V, \forall r,s \in \mathcal{C} \\
& z_{js} - y_{ijrs} \geq 0 \\
& 1 - z_{ir} - z_{js} + y_{ijrs} \geq 0 \\
& \sum_{r=1}^{K} z_{ir} = 1 && \forall i \in V \\
& z_{ir}, y_{ijrs} \in \{0,1\} && \forall i,j \in V, \forall r,s \in \mathcal{C} \\
& \omega_{rs}, x_{ijrs} \in \mathbb{R}^{+} && \forall i,j \in V, \forall r,s \in \mathcal{C}
\end{aligned}
$$

Solving the MILP

- **Lazy constraints:** The MILP has an infinite number of constraints due to the outer-approximation process. We introduce violated constraints every time an integer feasible solution is found (*lazy constraints callback* in `CPLEX`).
- **Bounds tightening:** Formulations with big-M constants tend to suffer from a weak continuous relaxation, especially if the values for the lower and upper bounds, $\underline{M}$ and $\overline{M}$, are not tight enough. To circumvent this issue we identified tighter bounds.
- **Symmetry-breaking constraints:** Any permutation of the clusters indices gives an equivalent solution $\Rightarrow$ we use additional symmetry-breaking constraints.

Performance of the exact methods

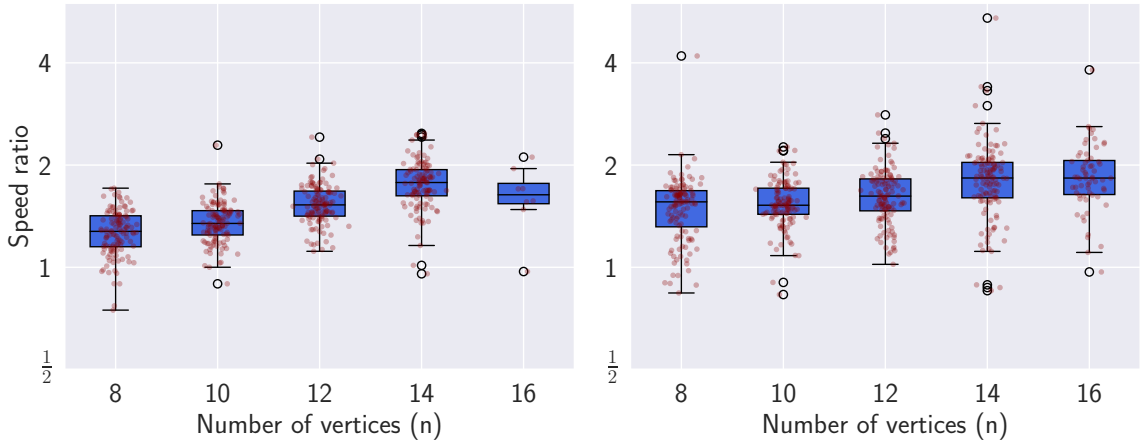


Figure 9: Impact of symmetry breaking constraints, for the MINLP (left) and MILP (right)

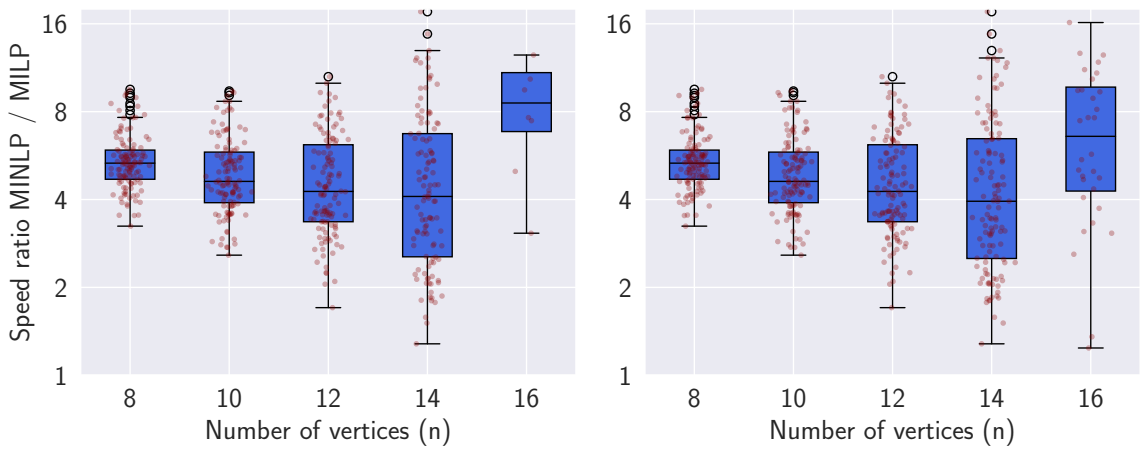


Figure 10: MILP vs. MINLP model, without (left) and with (right) symmetry breaking constraints.

Expectation-Maximization (EM) algorithms

As seen in our experiments, the optimal solution approaches do not scale to very large problems. Therefore, we aim to assess the performance of classical heuristics for this problem based on expectation-maximization (EM). These heuristics work in two steps:

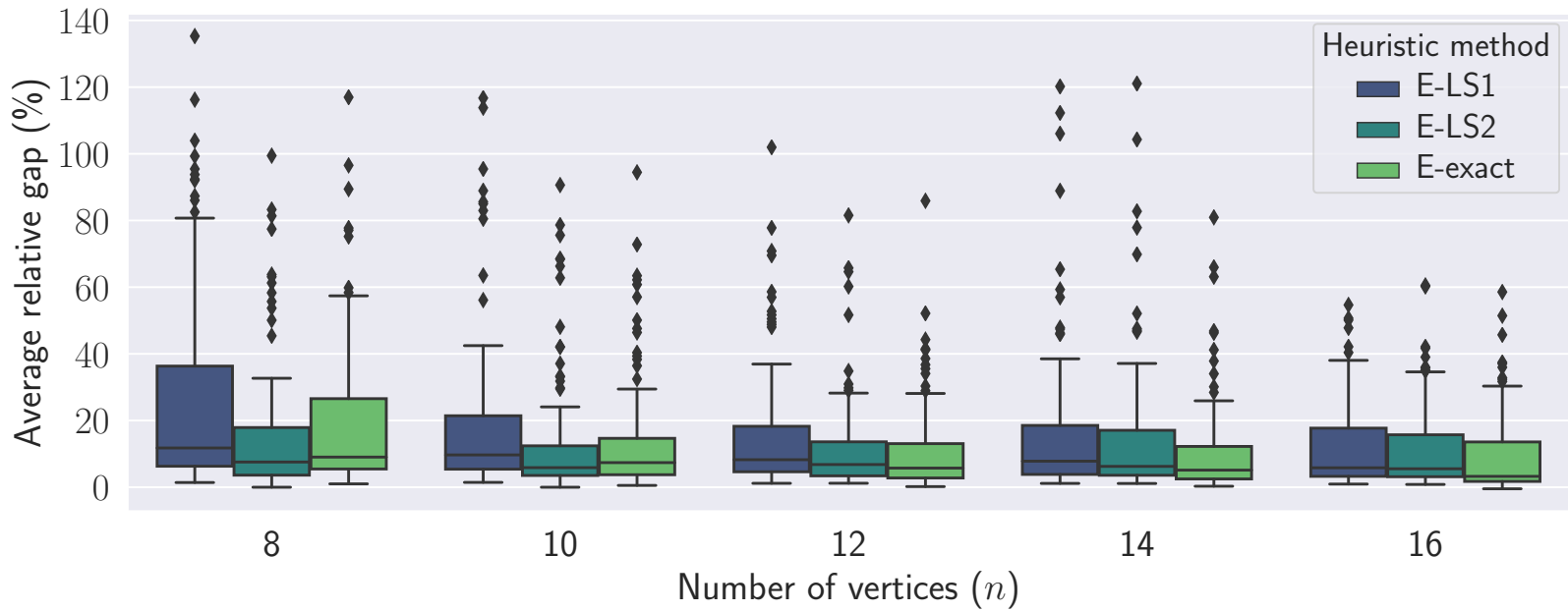
- Maximization step (M-step): Fix the assignments $\mathbf{Z}$ and find the optimal value of $\mathbf{\Omega}$. Solved by differentiation.

$$\omega^*_{rs} = \frac{\sum_{i,j} A_{ij} z_{ir} z_{js}}{\sum_{i,j} \frac{k_i k_j}{2m} z_{ir} z_{js}} = \frac{2m \cdot m_{rs}}{\kappa_r \kappa_s}$$

- Expectation step (E-step): Fix $\mathbf{\Omega}$ and search for the optimal community assignments $\mathbf{Z}$. Three variants:
  - Local search on the community assignment variables (`E-LS1`)
  - Local search integrated with M-step (`E-LS2`)
  - Exact community assignments (`E-exact`)

Performance of the heuristic methods

- For these data sets, the solution value (in the objective space) of the EM algorithms can be quite volatile, with average gaps around 10%.
- Multiple runs need to performance for a more robust performance
- Further investigation is needed on larger-scale data sets, though this will require significant methodological progress on mathematical programming algorithms for community detection.



Content

Support Vector Machines (SVM)

- Identifying a hyperplane that separates two classes of data points with maximal separation.
- When data is not linearly separable (as common in practical applications), SVMs rely on loss functions to penalize data points within the margin or on the wrong side of the hyperplane.

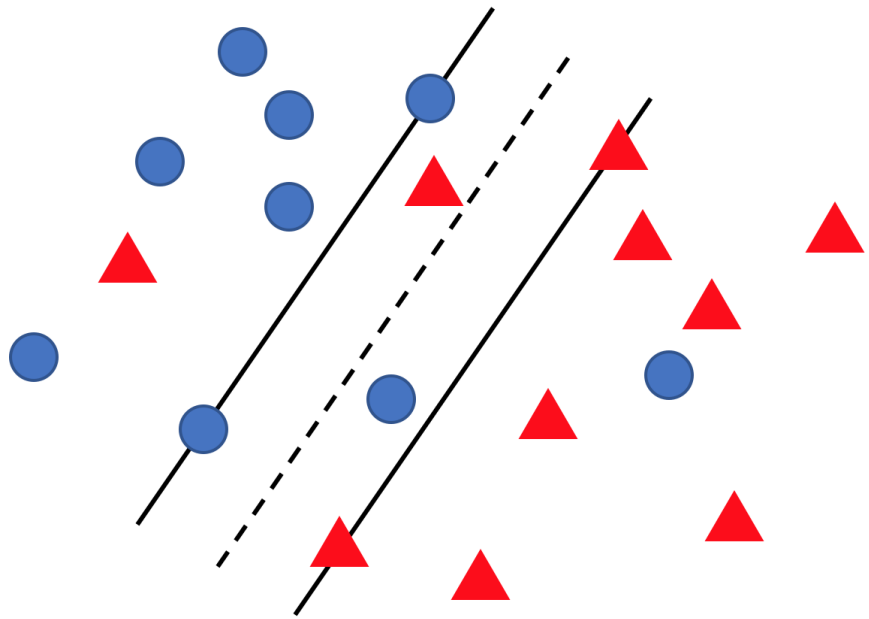


Figure 11: A hyperplane separating two classes

Support Vector Machines (SVM)

Classical SVM with hinge-loss penalization:

$$\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \tag{7}$$

$$\text{s.t.} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \tag{8}$$

$$\xi_i \geq 0 \tag{9}$$

- ξ_i stands for the continuous error for an observation i proportionally to the distance from the separating hyperplane, and C is the trade-off in maximizing the margin versus minimizing the error and
- The hinge-loss function is known to be very sensitive to outliers and lack robustness since this function is unbounded [13].

Non-convex support vector machines

- In recent years, there has been renewed interest on interpretable and robust machine learning models trained through combinatorial optimization algorithms.
- The classical SVM has a main weakness: outliers have an unbounded influence on the objective. Therefore, non-convex variations of this model have been proposed to mitigate the impact of these outliers [5].
- Due to non-convexity, requires a MILP formulation.

Non-convex support vector machines

SVM with hard-margin loss:

$$\min_{\mathbf{w},b,\xi,z} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} z_i \tag{10}$$

$$\text{s.t.} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, \text{ if } z_i = 0 \tag{11}$$

$$z_i \in \{0,1\} \tag{12}$$

SVM with ramp loss:

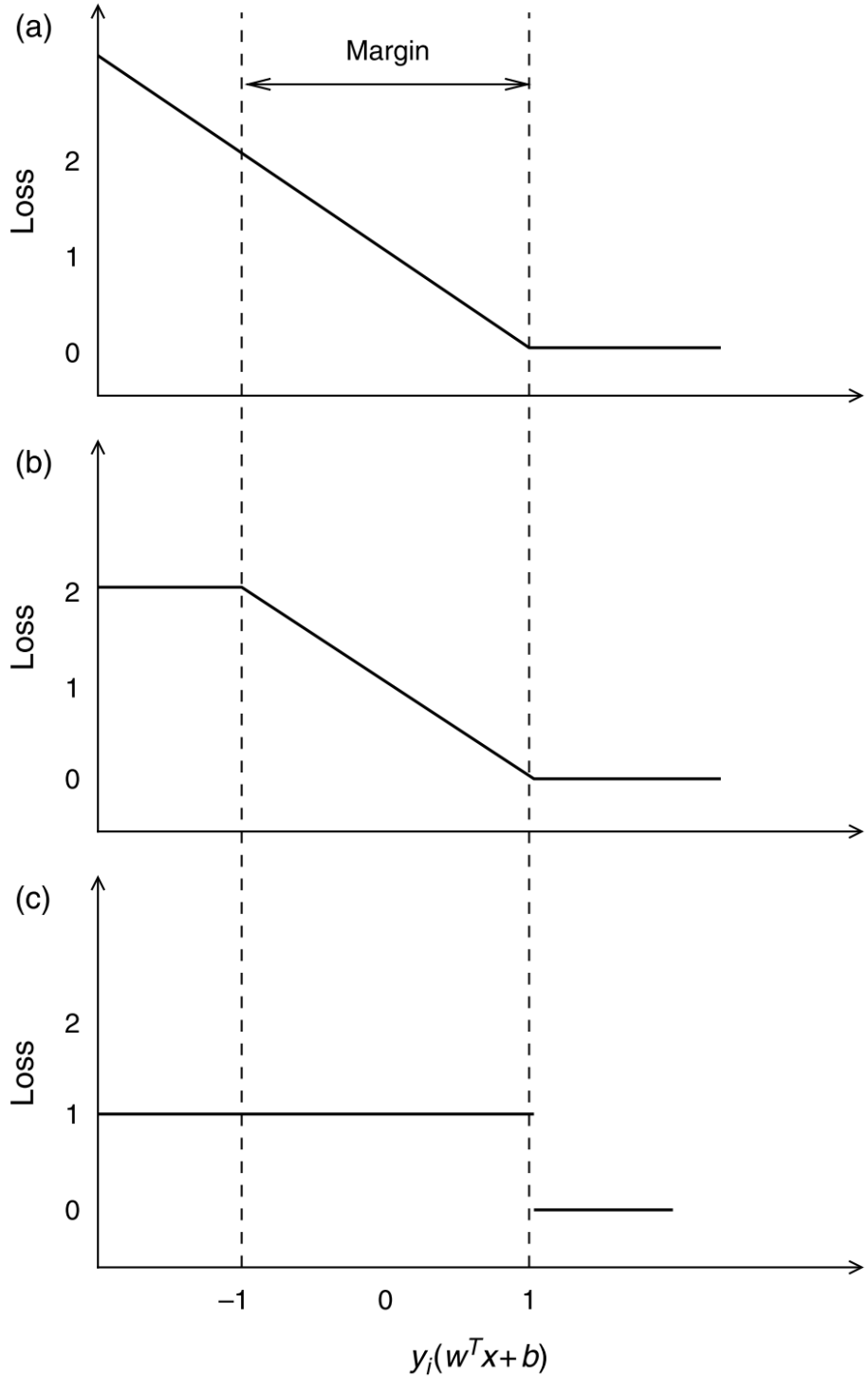
$$\min_{\mathbf{w},b,\xi,z} \quad \frac{1}{2}||\mathbf{w}||^2 + C\left(\sum_{i=1}^{n}\xi_i + 2\sum_{i=1}^{n} z_i\right) \tag{13}$$

$$\text{s.t.} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \text{ if } z_i = 0 \tag{14}$$

$$z_i \in \{0,1\} \tag{15}$$

$$0 \leq \xi_i \leq 2 \tag{16}$$

Constraints (11) and (14) are indicator constraints, i.e., constraints that either hold or are relaxed depending on the value of a binary variable. These constraints can be linearized using a big "M", such that (11) becomes $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - Mz_i$.



Misclassified samples are on the left side.
(a) traditional hinge loss,
(b) ramp loss, and
(c) hard margin loss.

An observation whose left-hand side falls between -1 and 1 lies in the margin.

Source of the figure: [5].

Non-convex support vector machines

- However, non-convex SVM models are much harder to solve. Instances of a few hundreds (up to 500) training samples are still out of reach of current MILP approaches:

Size	# Opts	Gap (%)	Size	# Opts	Gap (%)
60	75	0.00	60	69	1.49
100	44	13.17	100	31	25.34
200	5	54.32	200	0	66.69
500	0	90.28	500	0	92.64

Table 6: Results on SVM with hard-margin loss: set A (left) and set B (right). Data sets from [5]

Non-convex support vector machines

- However, non-convex SVM models are much harder to solve. Instances of a few hundreds (up to 500) training samples are still out of reach of current MILP approaches:

Size	# Opts	Gap (%)	Size	# Opts	Gap (%)
60	30	16.89	60	16	34.21
100	0	46.16	100	0	67.07
200	0	78.88	200	0	88.79
500	0	94.38	500	0	96.07

Table 7: Results on SVM with ramp loss: set A (left) and set B (right). Data sets from [5]

Non-convex support vector machines

- More effective MIP approaches are needed (possibly exploiting sparsity and other decomposition approaches?).
- A disciplined evaluation of heuristics for these formulations would also be a critical asset for practical cases.

Content

Other combinatorial optimization models in ML

- Many other ML models are currently being re-evaluated under the lenses of modern MIP approaches:
  - Optimal decision trees [3, 4, 7, 11]
  - Optimal training of sparse ML models [2, 9, 12]
  - Combinational optimization for adversary generation and model validation [8]
  - Optimization for interpretable ML... (see the 2^{nd} part of this course)
- The opportunities and research perspectives along these lines are numerous.

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*, 88, 569–583"

"de Araujo, B.S. A MIP approach for Community Detection in the Stochastic Block Model. Master Thesis, PUC-Rio, 2020"

Source codes available at:
`https://github.com/danielgribel`
`https://github.com/vidalt`

Bibliography I

[1] Aloise, Daniel, Pierre Hansen, Leo Liberti. 2012. An improved column generation algorithm for minimum sum-of-squares clustering. *Mathematical Programming* .

[2] Bertsimas, D., B. van Parys. 2020. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *Annals of Statistics* **48**(1) 300–323.

[3] Bertsimas, Dimitris, Jack Dunn. 2017. Optimal classification trees. *Machine Learning* **106**(7) 1039–1082.

[4] Blanquero, R., E. Carrizosa, C. Molero-Río, D. Romero Morales. 2020. Sparsity in optimal randomized classification trees. *European Journal of Operational Research* **284** 255–272.

[5] Brooks, J.P. 2011. Support vector machines with the ramp loss and the hard margin loss. *Operations Research* **59**(2) 467–479.

[6] de Araujo, B.S. 2020. A MIP approach for Community Detection in the Stochastic Block Model. Ph.D. thesis, PUC-Rio.

[7] Firat, Murat, Guillaume Crognier, Adriana F. Gabor, C. A.J. Hurkens, Yingqian Zhang. 2020. Column generation based heuristic for learning classification trees. *Computers and Operations Research* **116** 104866.

[8] Fischetti, Matteo, Jason Jo. 2018. Deep neural networks and mixed integer linear optimization. *Constraints* **23**(3) 296–309.

[9] Gambella, C., B. Ghaddar, J. Naoum-Sawaya. 2020. Optimization problems for machine learning: A survey. *European Journal of Operational Research, In Press* .

Bibliography II

[10] Gribel, D., T. Vidal. 2019. HG-means: A scalable hybrid metaheuristic for minimum sum-of-squares clustering. *Pattern Recognition* **88** 569–583.

[11] Günlük, Oktay, Jayant Kalagnanam, Minhan Li, Matt Menickelly, Katya Scheinberg. 2019. Optimal Decision Trees for Categorical Data via Integer Programming. Tech. rep.

[12] Labbé, M., L.I. Martínez-Merino, A.M. Rodríguez-Chía. 2019. Mixed integer linear programming for feature selection in support vector machine. *Discrete Applied Mathematics* **261** 276–304.

[13] Wu, Yichao, Yufeng Liu. 2007. Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association* **102**(479) 974–983.