

# COBRA: A Fast and Simple Method for Active Clustering with Pairwise Constraints

Toon Van Craenendonck, Sebastijan Dumančić and Hendrik Blockeel

Department of Computer Science, KU Leuven, Belgium

{firstname.lastname}@kuleuven.be

## Abstract

Clustering is inherently ill-posed: there often exist multiple valid clusterings of a single dataset, and without any additional information a clustering system has no way of knowing which clustering it should produce. This motivates the use of constraints in clustering, as they allow users to communicate their interests to the clustering system. Active constraint-based clustering algorithms select the most useful constraints to query, aiming to produce a good clustering using as few constraints as possible. We propose COBRA, an active method that first over-clusters the data by running K-means with a  $K$  that is intended to be too large, and subsequently merges the resulting small clusters into larger ones based on pairwise constraints. In its merging step, COBRA is able to keep the number of pairwise queries low by maximally exploiting constraint transitivity and entailment. We experimentally show that COBRA outperforms the state of the art in terms of clustering quality and runtime, without requiring the number of clusters in advance.

## 1 Introduction

Clustering is inherently subjective [Caruana *et al.*, 2006; von Luxburg *et al.*, 2014]: a single dataset can often be clustered in multiple ways, and different users may prefer different clusterings. This subjectivity is one of the motivations for constraint-based (or semi-supervised) clustering [Wagstaff *et al.*, 2001; Bilenko *et al.*, 2004]. Methods in this setting exploit background knowledge to obtain clusterings that are more aligned with the user’s preferences. Often, this knowledge is given in the form of pairwise constraints that indicate whether two instances should be in the same cluster (a must-link constraint) or not (a cannot-link constraint) [Wagstaff *et al.*, 2001]. In traditional constraint-based clustering systems the set of constraints is assumed to be given a priori, and in practice, the pairs that are queried are often selected randomly. In contrast, in active clustering [Basu *et al.*, 2004a; Mallapragada *et al.*, 2008; Xiong *et al.*, 2014] it is the method itself that decides which pairs to query. Typically, active methods query pairs that are more informative than random ones, which improves clustering quality.

This work introduces an active constraint-based clustering method named *Constraint-Based Repeated Aggregation* (COBRA). It differs from existing approaches in several ways. First, it aims to maximally exploit constraint transitivity and entailment [Wagstaff *et al.*, 2001], two properties that allow deriving additional constraints from a given set of constraints. By doing this, the actual number of pairwise constraints that COBRA works with is typically much larger than the number of pairwise constraints that are queried from the user. Secondly, COBRA introduces the assumption that there exist small local regions in the data that are grouped together in all potential clusterings. To clarify this, consider the example of clustering images of people taking different poses (e.g. facing left or right). There are at least two natural clustering targets for this data: one might want to cluster based on identity or pose. In an appropriate feature space, one expects images that agree on both criteria (i.e. of a single person, taking a single pose) to be close. There is no need to consider all of these instances individually, as they will end up in the same cluster for each of the two targets that the user might be interested in. COBRA aims to group such instances into a *super-instance*, such that they can be treated jointly in the clustering process. Doing so substantially reduces the number of pairwise queries. Thirdly, COBRA is an inherently active method: the constraints are selected during the execution of the algorithm itself, as constraint selection and algorithm execution are intertwined. In contrast, existing approaches consist of a component that selects constraints and another one that uses them during clustering.

Our experiments show that COBRA outperforms state-of-the-art active clustering methods in terms of both clustering quality and runtime. Furthermore, it has the distinct advantage that it does not require knowing the number of clusters beforehand, as the competitors do. In many realistic clustering scenarios this number is not known, and running an algorithm with the wrong number of clusters often results in a significant decrease in clustering quality.

We discuss related work on (active) constraint-based clustering in section 2. In section 3 we elaborate the key ideas in COBRA and describe the method in more detail. We present our experimental evaluation in section 4, and conclude in section 5.

## 2 Background and Related Work

Most existing constraint-based methods are extensions of well-known unsupervised clustering algorithms. They use the constraints either in an adapted clustering procedure [Wagstaff *et al.*, 2001; Rangapuram and Hein, 2012; Wang *et al.*, 2014], to learn a similarity metric [Xing *et al.*, 2003; Davis *et al.*, 2007], or both [Bilenko *et al.*, 2004; Basu *et al.*, 2004b]. Constraint-based extensions have been developed for most clustering algorithms, including K-means [Wagstaff *et al.*, 2001; Bilenko *et al.*, 2004], spectral clustering [Rangapuram and Hein, 2012; Wang *et al.*, 2014], DBSCAN [Lelis and Sander, 2009; Campello *et al.*, 2013] and EM [Shental *et al.*, 2004].

Basu *et al.* [2004a] introduce a strategy to select the most informative constraints, prior to performing a single run of a constraint-based clustering algorithm. They show that active constraint selection can improve clustering performance. Several selection strategies have been proposed since [Mallapragada *et al.*, 2008; Xu *et al.*, 2005; Xiong *et al.*, 2014], most of which are based on the classic approach of uncertainty sampling. As COBRA also chooses which pairs to query, we consider it to be an active method, and in our experiments we compare to other methods in this setting. Note, however, that COBRA is quite different from existing methods in active constrained clustering and active learning in general. For COBRA, selecting which pairs to query is inherent to the clustering procedure, whereas for most other methods the selection strategy is optional and considered to be a separate component.

In its core, COBRA is related to hierarchical clustering as it follows the same procedure of sequentially trying to merge the two closest clusters. Constraints have been used in hierarchical clustering before but in different ways. Davidson *et al.* [2009], for example, present an algorithm to find a clustering hierarchy that is consistent with a *given* set of constraints. Nogueira *et al.* [2012] propose an active semi-supervised hierarchical clustering algorithm that is based on merge confidence. Also related to ours is the work of Campello *et al.* [2013], who have developed a framework to extract from a given hierarchy a flat clustering that is consistent with a given set of constraints. The key difference is that COBRA starts from super-instances, i.e. small clusters produced by K-means, and that each merging decision is settled by a pairwise constraint.

The idea of working with a small number of representatives (in our case the super-instance medoids, as will be discussed in section 3) instead of all individual instances has been used before, but for very different purposes. For example, Yan *et al.* [2009] use it to speed up unsupervised spectral clustering, whereas we use it to reduce the number of pairwise queries.

## 3 Constraint-Based Repeated Aggregation

Constraint-based clustering algorithms aim to produce a clustering of a dataset that resembles an unknown target clustering  $Y$  as close as possible. The algorithm cannot query the cluster labels in  $Y$  directly, but can query the relation between pairs of instances. A must-link constraint is obtained if the instances have the same cluster label in  $Y$ , a cannot-link constraint otherwise. The aim is to produce a clustering that is close to the target clustering  $Y$ , using as few pairwise queries as possible.

Several strategies can be used to exploit constraints in clustering. Figure 1 illustrates some of them. The most naive strategy is to query all pairwise relations, and construct clusters as sets of instances that are connected by a must-link constraint (Figure 1a). Though this is clearly not a good strategy in any scenario, it allows us to formulate a baseline for further improvements. It always results in a perfect clustering, but at a very high cost: for a dataset of  $N$  instances,  $\binom{N}{2}$  questions are asked.

The previous strategy can be improved by exploiting *constraint transitivity* and *entailment*, two well known properties in constraint-based clustering [Wagstaff *et al.*, 2001; Bilenko *et al.*, 2004]. Must-link constraints are known to be transitive:

$$\text{must-link}(A, B) \wedge \text{must-link}(B, C) \Rightarrow \text{must-link}(A, C),$$

whereas cannot-link constraints have an entailment property:

$$\text{must-link}(A, B) \wedge \text{cannot-link}(B, C) \Rightarrow \text{cannot-link}(A, C).$$

Thus, every time a constraint is queried and added to the set of constraints, transitivity and entailment can be applied to expand the set. This strategy is illustrated in Figure 1b. Exploiting transitivity and entailment significantly reduces the number of pairwise queries needed to obtain a clustering, without a loss in clustering quality.

The order in which constraints are queried strongly influences the number of constraints that can be derived. In general, it is better to obtain must-link constraints early on. That way, any future query involving one of the instances connected by a must-link also applies to all others. This suggests querying the closest pairs first, as they are more likely to belong to the same cluster and hence be connected by a must-link constraint. This is strategy is illustrated in Figure 1c.

The previous strategies all obtain a perfect clustering, but require a high number of queries which makes them inapplicable for reasonably sized datasets. To further reduce the number of queries, COBRA groups similar instances into *super-instances* and only clusters their representatives, i.e. medoids. It assumes that all instances within a super-instance are connected by a must-link constraint. While clustering the medoids, COBRA uses both previously discussed strategies of querying the closest pairs and exploiting transitivity and entailment. This strategy, illustrated in Figure 1d, results in a substantial reduction of the number of queries. It does not always result in a perfect clustering as it is possible that the instances within a particular super-instance should not be grouped together w.r.t. the target clustering.

Table 1 illustrates to what extent each of the improvements described above reduces the number of queries. We perform an extensive evaluation of the quality of the clusterings that COBRA produces in section 4.

### 3.1 Algorithmic Description

After presenting the main motivations for each step of COBRA, we now give a more detailed description in Algorithm 1. Let  $\mathcal{X} = \{x_i\}_{i=1}^N$ ,  $x_i \in \mathbb{R}^m$  the instances to be clustered. The set of instances  $\mathcal{X}$  is first *over-clustered* into  $N_S$  disjoint subsets, namely super-instances  $\{\mathcal{S}_i\}_{i=1}^{N_S}$ , such that  $\bigcup_i \mathcal{S}_i = \mathcal{X}$ .

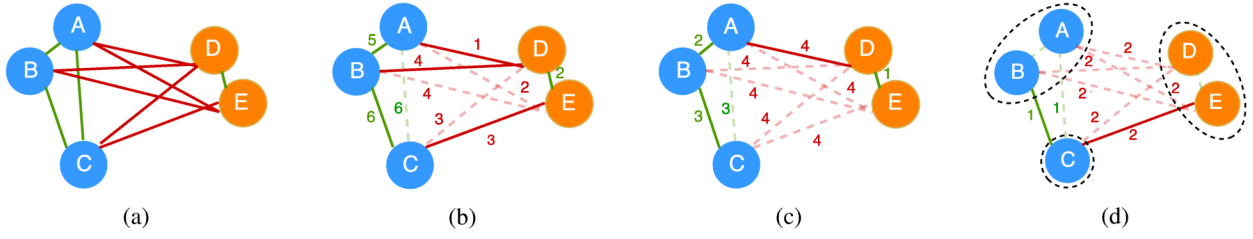


Figure 1: Illustration of different querying strategies. Colors indicate the desired clustering. Solid green lines indicate *must-link*, red ones *cannot-link* constraints. Dashed lines indicate derived constraints with the same color code. The number next to the solid line indicates the ordering of queried constraints, whereas the number next to the dashed line indicates the constraint number from which the constraint was derived. (a) Querying all 10 constraints (b) Exploiting entailment and transitivity results in querying 6 constraints. (c) Querying the closest pairs first results in 4 constraints. (d) Introducing *super-instances* (dashed ellipses) results in only 2 queries.

dataset	# instances	total # pairs		# queries		# queries		# queries
iris	147	10731		409		155		34
wine	178	15753		457		187		35
dermatology	358	63903	transitivity	1660	closest pairs	379	super	42
hepatitis	112	6216	entailment	173	first	140	instances	39
ecoli	336	56280		1343		440		51
opdigits389	1151	661825		3380		1164		54

Table 1: This table shows the total number of pairwise relations in several datasets (column 3), as well the number of pairwise queries that is required when exploiting transitivity and entailment (column 4, these numbers are averages of runs for 5 random orderings of pairwise queries), and additionally querying the closest pairs first (column 5). The last column shows the number of pairwise queries (averaged over 5 runs) when COBRA is run with 25 super-instances.

This over-clustering is obtained by running K-means with a  $K$  that may be significantly larger than the actual number of clusters. Each super-instance  $S_i$  is represented by its medoid  $s_i$ , forming a set of super-instance representatives  $S = \{s_1, \dots, s_{N_S}\}$ . All pairwise queries that are performed are between these super-instance representatives. The goal of COBRA is now to cluster these representatives into disjoint subsets  $\{C_1, \dots, C_{N_C}\}$  of  $S$  ( $\bigcup_i C_i = S$ ).

Each cluster  $C_i$  is a set of super-instance representatives, but conceptually contains all the points in the corresponding super-instances. The number of clusters  $N_C$  is unknown a priori and will be determined during the clustering procedure. Initially, there are  $N_S$  clusters, each containing a single super-instance representative, as shown on line 2 in Algorithm 1. These clusters are merged (if necessary) in the subsequent while loop.

In each iteration, COBRA first sorts all pairs of clusters between which there is no *cannot-link* constraint (line 6). The distance between clusters, by which the pairs are sorted, is defined as follows (as in single-linkage clustering):

$$d(C_1, C_2) = \min_{s_1 \in C_1, s_2 \in C_2} \|s_1 - s_2\|_2 \quad (1)$$

Next, COBRA loops over the pairs of clusters and checks whether they should be merged. It starts by selecting the closest pair of super-instance representatives of the clusters (line 9), and asks whether they should be in the same cluster (line 10). If this is the case, the clusters are merged (lines 11 and 12), and the while-loop is restarted (as the set of clusters is changed). If that is not the case, the pair of representatives

is added to the set of *cannot-link* constraints (line 16), and the inner loop continues by inspecting the next pair of clusters. The execution stops when all clusters are complete and no merge is to be done anymore.

### Number of Super-instances and Number of Queries

The exact number of queries COBRA needs depends on the extent to which querying the closest pairs first leads to *must-link* constraints and on the actual number of clusters. It is thus difficult to determine it before the execution. However, an estimate in terms of a lower and upper bound can be posed:

$$N_S - N_C + \binom{N_C}{2} \lesssim \# \text{ queries} \leq \binom{N_S}{2}$$

with  $N_S$  the number of super-instances, and  $N_C$  the number of clusters in the target clustering. In the worst case, COBRA will need to query all pairwise relations between super-instances, which requires  $\binom{N_S}{2}$  queries. This happens if there is a *cannot-link* between each pair of super-instances. In practice, COBRA typically needs much less than  $\binom{N_S}{2}$  queries. If (i) super-instances are perfectly homogeneous w.r.t. the target clustering and (ii) the distances between *must-link* pairs are smaller than the distances between *cannot-link* pairs, COBRA needs exactly  $N_S - N_C + \binom{N_C}{2}$  queries.  $N_S - N_C$  *must-link* constraints are needed to merge the  $N_S$  super-instances into the  $N_C$  clusters, followed by an additional  $\binom{N_C}{2}$  queries to ensure nothing can be merged anymore. This formula is inapplicable in practice as the number of clusters is not known beforehand, and thus serves as a means to understand the number of queries that might be needed. Figure 2 compares this lower bound to the actual number of queries that was needed

---

**Algorithm 1** COBRA
 

---

**Require:**  $\mathcal{X}$ : a dataset,  
 $N_S$ : the number of super-instances,  
**Ensure:**  $\mathbf{C}$ , a clustering of  $D$

```

1:  $\{\mathcal{S}_i\}_{i=1}^{N_S} = \text{K-means}(\mathcal{X}, N_S)$ 
2:  $\forall i = 1, \dots, N_S : \mathcal{C}_i = \{\mathbf{s}_i\}$ 
3:  $\mathcal{CL} = \emptyset$ 
4:  $\text{mergeHappened} = \text{True}$ 
5: while  $\text{mergeHappened}$  do
6:    $\mathcal{P} = \{\mathcal{C}_1, \mathcal{C}_2 : \nexists \mathbf{x} \in \mathcal{C}_1, \mathbf{y} \in \mathcal{C}_2 : (\mathbf{x}, \mathbf{y}) \in \mathcal{CL}\}$ ,  

   ordered by  $d(\mathcal{C}_1, \mathcal{C}_2)$ 
7:    $\text{mergeHappened} = \text{False}$ 
8:   for  $\mathcal{C}_1, \mathcal{C}_2$  in  $\mathcal{P}$  do
9:      $\mathbf{s}_a, \mathbf{s}_b = \arg \min_{\mathbf{s}_1 \in \mathcal{C}_1, \mathbf{s}_2 \in \mathcal{C}_2} \|\mathbf{s}_1 - \mathbf{s}_2\|_2$ 
10:    if  $\text{must-link}(\mathbf{s}_a, \mathbf{s}_b)$  then
11:       $\mathcal{C}_1 = \mathcal{C}_1 \cup \mathcal{C}_2$ 
12:       $\mathbf{C} = \mathbf{C} \setminus \{\mathcal{C}_2\}$ 
13:       $\text{mergeHappened} = \text{True}$ 
14:      break
15:    else
16:       $\mathcal{CL} = \mathcal{CL} \cup \{(\mathbf{s}_a, \mathbf{s}_b)\}$ 
17:    end if
18:  end for
19: end while
20: return  $\mathbf{C}$ 
```

---

by COBRA for 21 clustering tasks (these will be described in more detail in section 4). For most tasks, the actual number of pairwise queries is relatively close to the lower bound. It is possible to get a smaller number of queries than that suggested by the lower bound: this happens if a single super-instance contains the instances of two actual clusters, rendering the  $\binom{N_C}{2}$  factor inaccurate.

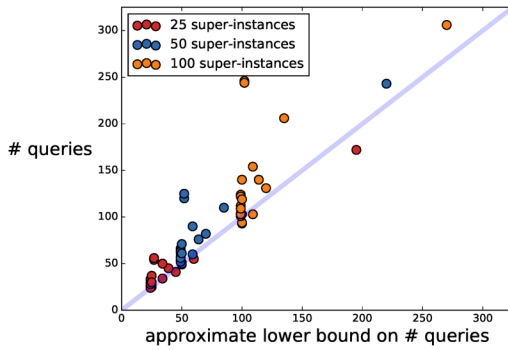


Figure 2: The estimated lower bound on the number of queries vs. the number of queries needed by COBRA (averaged over 5 CV folds). Each dot corresponds to one of the 21 clustering tasks.

## 4 Experimental Evaluation

In this section, we discuss the experimental evaluation of COBRA.

### Existing Constraint-based Algorithms

We compare COBRA to the following state-of-the-art constraint-based clustering algorithms:

- MPCKMeans [Bilenko *et al.*, 2004] is a hybrid constraint-based extension of K-means: it uses metric learning and an adapted clustering procedure combining the within-cluster sum of squares with the cost of violating constraints in the objective. We use the implementation that is available in the WekaUT package<sup>1</sup>.
- Constrained Spectral Clustering (COSC) [Rangapuram and Hein, 2012] is based on spectral clustering, but optimizes for a modified objective that also takes constraint violation into account. We use the code provided by the authors on their web page<sup>2</sup>.

It is important to note that, in contrast to COBRA, COSC and MPCKMeans require the number of clusters as an input parameter. In our experiments, the true number of clusters is provided to these algorithms. In many clustering applications, however, this number is typically not known beforehand. Thus, COSC and MPCKMeans are at an advantage.

### Active Selection Strategies

Each of these algorithms is combined with the following two active selection strategies:

- MinMax [Mallapragada *et al.*, 2008] starts with an *exploration* phase in which  $K$  (the number of clusters, assumed to be known in advance) neighborhoods with cannot-links between them are sought. In the subsequent *consolidation* phase these neighborhoods are expanded by selecting the most uncertain instances and determining their neighborhood membership by means of pairwise constraints. We set the width parameter of the Gaussian kernel to the 20th percentile of the distribution of pairwise Euclidean distances, as in [Mallapragada *et al.*, 2008].
- NPU [Xiong *et al.*, 2014] is also based on the concept of neighborhoods, but in contrast to MinMax it is an iterative method: the data is clustered multiple times, and each clustering is used to determine the next set of pairs to query. Xiong *et al.* [2014] show that NPU typically outperforms MinMax in terms of clustering quality, at the cost of increased runtime.

### Datasets

We experiment with 15 UCI datasets: iris, wine, dermatology, hepatitis, glass, ionosphere, optdigits389, ecoli, breast-cancer-wisconsin, segmentation, column.2C, parkinsons, spambase, sonar and yeast. Most of these datasets have been used in earlier work on constraint-based clustering [Bilenko *et al.*, 2004; Xiong *et al.*, 2014]. Optdigits389 contains digits 3, 8 and 9 of the UCI handwritten digits data [Bilenko *et al.*, 2004; Mallapragada *et al.*, 2008]. Duplicate instances were removed for all these datasets, and all data was normalized between 0 and 1. We also perform experiments on the CMU faces dataset, which contains 624 images of 20 persons taking different poses, with different expressions, with and without sunglasses.

<sup>1</sup><http://www.cs.utexas.edu/users/ml/risc/code/>

<sup>2</sup><http://www.ml.uni-saarland.de/code/cosc/cosc.htm>

Table 2: Wins and losses aggregated over all 21 clustering tasks. After each win (loss) count, we report the average margin by which COBRA wins (loses). For win counts marked with an asterisk, the differences are significant according to the Wilcoxon test with  $p < 0.05$ .

	25 super-instances		50 super-instances		100 super-instances	
	win	loss	win	loss	win	loss
COBRA vs. MPCKM-MinMax	<b>13</b> (0.14)	8 (0.12)	<b>13</b> (0.16)	8 (0.09)	<b>12</b> (0.19)	9 (0.05)
COBRA vs. MPCKM-NPU	<b>11</b> (0.16)	10 (0.11)	<b>17*</b> (0.12)	4 (0.09)	<b>12</b> (0.17)	9 (0.06)
COBRA vs. COSC-MinMax	<b>15*</b> (0.21)	6 (0.05)	<b>16*</b> (0.21)	5 (0.06)	<b>14*</b> (0.21)	7 (0.04)
COBRA vs. COSC-NPU	<b>15*</b> (0.20)	6 (0.04)	<b>14*</b> (0.23)	7 (0.04)	<b>13*</b> (0.23)	8 (0.03)

Hence, this dataset has 4 target clusterings: identity, pose, expression and sunglasses. We extract a 2048-value feature vector for each image by running it through the pre-trained Inception-V3 network [Szegedy *et al.*, 2015] and storing the output of the second last layer. Finally, we also cluster the 20 newsgroups text data. For this dataset, we consider two tasks: clustering documents from 3 newsgroups on related topics (the target clusters are comp.graphics, comp.os.ms-windows and comp.windows.x, as in [Basu *et al.*, 2004a; Mallapragada *et al.*, 2008]), and clustering documents from 3 newsgroups on very different topics (alt.atheism, rec.sport.baseball and sci.space, as in [Basu *et al.*, 2004a; Mallapragada *et al.*, 2008]). We first extract tf-idf features, and next apply latent semantic indexing (as in [Mallapragada *et al.*, 2008]) to reduce the dimensionality to 10. This brings the total to 17 datasets, for which 21 clustering tasks are defined (15 UCI datasets with a single target, CMU faces with 4 targets, and 2 subsets of the 20 newsgroups data).

### Experimental Methodology

We use a cross-validation procedure that is highly similar to the ones used in e.g. [Basu *et al.*, 2004a] and [Mallapragada *et al.*, 2008]. In each of 5 folds, 20% of the instances are set aside as the test set. The clustering algorithm is then run on the entire dataset, but can only query pairwise constraints for which both instances are in the training set. To evaluate the quality of the resulting clustering, we compute the Adjusted Rand index (ARI, [Hubert and Arabie, 1985]) only on the instances in the test set. The ARI measures the similarity between two clusterings, in this case between the one produced by the constraint-based clustering algorithm and the one indicated by the class labels. An ARI of 0 means that the clustering is not better than random, 1 indicates a perfect clustering. The final score for an algorithm for a particular dataset is computed as the average ARI over the 5 folds.

The exact number of pairwise queries is not known beforehand for COBRA, but more super-instances generally results in more queries. To evaluate COBRA with varying amounts of user input, we run it with 25, 50 and 100 super-instances. For each fold, we execute the following steps:

- Run COBRA and count how many constraints it needs.
- Run the competitors with the same number of constraints.
- Evaluate the resulting clusterings by computing the ARI on the test set.

To make sure that COBRA only queries pairs of which both instances are in the training set, the medoid of a super-instance is calculated based on only the training instances in that super-instance (and as such, test instances are never queried during

clustering). In the rare event that a super-instance contains only test instances, it is merged with the nearest super-instance that does contain training instances. For the MinMax and NPU selection strategies, pairs involving an instance from the test set are simply excluded from selection.

### Results

The results over all 21 clustering tasks are summarized in Tables 2 and 3. Table 2 reports wins and losses against each of the 4 competitors. It shows that COBRA tends to produce better clusterings than its competitors. The difference with COSC is significant according to the Wilcoxon test with  $p < 0.05$ , whereas the difference with MPCKMeans is not. Table 3 shows the average ranks for COBRA and its competitors. The Friedman aligned rank test [Hodges and Lehmann, 1962], which has more power than the Friedman test when the number of algorithms under comparison is low [Garca *et al.*, 2010], indicates that for 50 and 100 super-instances, the differences in rank between COBRA and all competitors are significant, using a posthoc Holm test with  $p < 0.05$ .

 Table 3: For each dataset, all algorithms are ranked from 1 (best) to 5 (worst). This table shows the average ranks for 25, 50 and 100 super-instances. Algorithms for which the difference with COBRA is significant according to the Friedman aligned rank test and a post-hoc Holm test with  $p < 0.05$  are marked with an asterisk.

25 super-instances		50 super-instances		100 super-instances	
COBRA	2.43	COBRA	2.14	COBRA	2.52
MPCK-NPU	3.00	MPCK-MM*	3.00	COSC-NPU*	2.98
MPCK-MM	3.07	COSC-NPU*	3.02	MPCK-NPU*	3.00
COSC-MM*	3.12	COSC-MM*	3.26	MPCK-MM*	3.19
COSC-NPU*	3.40	MPCK-NPU*	3.57	COSC-MM*	3.31

### Running Competitors with Different Numbers of Queries

In the previous experiments, the competitors are run with the same number of queries that COBRA required, as for COBRA this cannot be fixed beforehand. One might wonder whether this constitutes an advantage for COBRA, and whether the above conclusions also hold when competitors can be run with different numbers of constraints. To answer this question, we run COBRA with a wider range of super-instances, and its competitors with more numbers of constraints. Figure 3 shows the results for 4 datasets, but the conclusions that are drawn here also hold for the others. A first conclusion is that for the datasets for which COBRA outperforms its competitors in the experiments discussed above, it also does so for larger numbers of constraints (e.g. in Figures 3a and 3b). As such, the results discussed in the previous section are representative. Secondly,



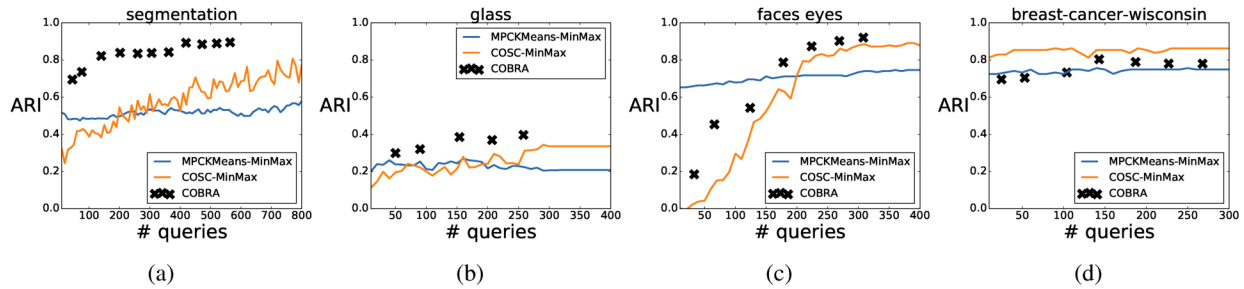


Figure 3: Comparing the clustering qualities for COBRA and its competitors for a wider range of numbers of constraints. For COBRA, each black marker shows the average number of questions that COBRA required for a particular number of super-instances, and its average ARI (over 5-fold CV). We only show the results for the MinMax selection strategy, but the conclusions that are drawn also hold for NPU. The number of super-instances for COBRA is 25, 50, 100 (as in the experiments before), 150, 200, 250, ...

clustering quality quickly plateaus for many datasets (e.g. in Figure 3d). This is especially true for MPCKMeans, which can be explained by its strong spherical bias. In contrast, for several datasets both COBRA and COSC produce increasingly better clusterings as more constraints are given (e.g. in Figures 3a and 3c).

### Selecting the Right Number of Clusters

COBRA does not require specifying the number of clusters  $K$  beforehand. Most often, it produces the correct  $K$  (i.e. the one indicated by the class labels). When it does not, the  $K$  it finds is very close to the correct one. In contrast, COSC and MPCKMeans do require specifying  $K$ . In the experiments discussed above, both of them were given the correct  $K$ . We have found experimentally that, in the majority of cases, running them with a different  $K$  reduces clustering quality, and often by a significant amount. Occasionally a different  $K$  improves results, but when this was the case it was typically only by a small margin. These results are omitted from the paper due to lack of space.

### Runtime

Figure 4 compares the runtimes of COBRA to those of the competitors. COBRA consists of two steps: constructing super-instances, and grouping them together to form clusters. Both are fast, as K-means is used for the first step and the second step is only applied to the small set of super-instances. As can be seen in Figure 4, the runtimes of MPCKMeans-MinMax are comparable to those of COBRA, which is not surprising as it is built on K-means. In contrast, COSC-MinMax is significantly more expensive, as it is built on spectral clustering. When used with the NPU selection strategy, both MPCKMeans and COSC become much slower, as NPU requires several runs of the clustering algorithm.

## 5 Conclusion

We have introduced COBRA, an active constraint-based clustering method. Unlike other methods, it is not built as an extension of an existing unsupervised algorithm. Instead, COBRA is inherently constraint-based. With its selection strategy, it aims to maximally exploit constraint transitivity and entailment. Our experiments show that COBRA outperforms the state of the art in terms of clustering quality and runtime, even

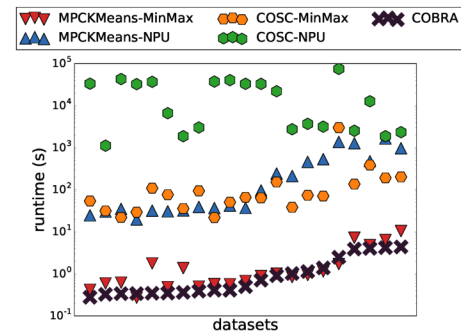


Figure 4: Each dot shows the runtime (averaged over 5 CV folds) of a method on one of the datasets. This figure shows the results for COBRA with 50 super-instances, the figures for 25 and 100 super-instances are comparable. COBRA and MPCKMeans-MinMax are highly scalable and always finish in under 10 seconds.

when the other methods have the advantage of being given the right number of clusters. In future work, we will investigate whether an appropriate number of super-instances can be determined automatically, e.g. by incrementally refining them when necessary.

## Acknowledgements

Toon Van Craenendonck is supported by the Agency for Innovation by Science and Technology in Flanders (IWT). Sebastijan Dumančić is supported by Research Fund KU Leuven (GOA/13/010).

## References

- [Basu *et al.*, 2004a] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM International Conference on Data Mining (SDM-04)*, April 2004.
- [Basu *et al.*, 2004b] Sugato Basu, Misha Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004)*, page 5968, January 2004.

- [Bilenko *et al.*, 2004] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proc. of 21st International Conference on Machine Learning*, pages 81–88, July 2004.
- [Campello *et al.*, 2013] Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies. *Data Mining and Knowledge Discovery*, 27(3):344–371, 2013.
- [Caruana *et al.*, 2006] Rich Caruana, Mohamed Elhawary, and Nam Nguyen. Meta clustering. In *Proc. of the International Conference on Data Mining*, 2006.
- [Davidson and Ravi, 2009] I Davidson and SS Ravi. Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. *Data mining and knowledge discovery*, pages 1–30, 2009.
- [Davis *et al.*, 2007] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning, ICML '07*, pages 209–216, New York, NY, USA, 2007. ACM.
- [Garca *et al.*, 2010] Salvador Garca, Alberto Fernandez, Julin Luengo, and Francisco Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044 – 2064, 2010. Special Issue on Intelligent Distributed Information Systems.
- [Hodges and Lehmann, 1962] J. L. Hodges and E. L. Lehmann. Rank methods for combination of independent experiments in analysis of variance. *The Annals of Mathematical Statistics*, 33(2):482–497, 1962.
- [Hubert and Arabie, 1985] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.
- [Lelis and Sander, 2009] Levi Lelis and Jörg Sander. Semi-supervised density-based clustering. In *2009 Ninth IEEE International Conference on Data Mining*, pages 842–847, Dec 2009.
- [Mallapragada *et al.*, 2008] Pavan K. Mallapragada, Rong Jin, and Anil K. Jain. Active query selection for semi-supervised clustering. In *Proc. of the 19th International Conference on Pattern Recognition*, 2008.
- [Nogueira *et al.*, 2012] Bruno M Nogueira, M Jorge, and Solange O Rezende. HCAC : Semi-supervised Hierarchical Clustering Using Confidence-Based Active Learning. (1):139–153, 2012.
- [Rangapuram and Hein, 2012] Syama S. Rangapuram and Matthias Hein. Constrained 1-spectral clustering. In *Proc. of the 15th International Conference on Artificial Intelligence and Statistics*, 2012.
- [Shental *et al.*, 2004] Noam Shental, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Computing Gaussian mixture models with EM using equivalence constraints. In *Advances in Neural Information Processing Systems 16*, 2004.
- [Szegedy *et al.*, 2015] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.
- [von Luxburg *et al.*, 2014] Ulrike von Luxburg, Robert C. Williamson, and Isabelle Guyon. Clustering: Science or Art? In *Workshop on Unsupervised Learning and Transfer Learning, JMLR Workshop and Conference Proceedings 27*, 2014.
- [Wagstaff *et al.*, 2001] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained K-means Clustering with Background Knowledge. In *Proc. of the Eighteenth International Conference on Machine Learning*, pages 577–584, 2001.
- [Wang *et al.*, 2014] Xiang Wang, Buyue Qian, and Ian Davidson. On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery*, 28(1):1–30, 2014.
- [Xing *et al.*, 2003] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512, 2003.
- [Xiong *et al.*, 2014] Sicheng Xiong, Javad Azimi, and Xiaoli Z. Fern. Active learning of constraints for semi-supervised clustering. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):43–54, 2014.
- [Xu *et al.*, 2005] Qianjun Xu, Marie desJardins, and Kiri L. Wagstaff. *Active Constrained Clustering by Examining Spectral Eigenvectors*, pages 294–307. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [Yan *et al.*, 2009] Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09*, pages 907–916, New York, NY, USA, 2009. ACM.