Limitations of Using Constraint Set Utility in Semi-Supervised Clustering

Toon Van Craenendonck and Hendrik Blockeel

Department of Computer Science
KU Leuven

Abstract. Semi-supervised clustering algorithms allow the user to incorporate background knowledge into the clustering process. Often, this background knowledge is specified in the form of must-link (ML) and cannot-link (CL) constraints, indicating whether certain pairs of elements should be in the same cluster or not. Several traditional clustering algorithms have been adapted to operate in this setting. We compare some of these algorithms experimentally, and observe that their performances vary significantly, depending on the data set and constraints. We use two previously introduced constraint set utility measures, consistency and coherence, to help explain these differences. Motivated by the correlation between consistency and clustering performance, we also examine its use in algorithm selection. We find this consistency-based approach to be unsuccessful, and explain this result by observing that the previously found correlation between utility measures and clustering performance is only present when we look at results of different data sets jointly. This limits the use of these constraint set utility measures, as often we are interested in using them in the context of a particular data set.

Keywords: semi-supervised clustering, constraint set utility, algorithm selection

1 Introduction

Clustering is the task of grouping data into clusters, or groups of similar objects. Traditional unsupervised clustering algorithms only rely on information intrinsic to the data. In contrast, in semi-supervised clustering [2, 19, 20] the user can provide background knowledge to guide the algorithm towards better clusterings. Often, such background knowledge is given in the form of pairwise constraints, stating whether elements should be in the same cluster (must-link) or not (cannot-link). Semi-supervised extensions have been developed for most of the traditional clustering algorithms, such as K-means [19], DBSCAN [10, 14] and spectral clustering [12]. A user who wants to cluster a data set, and influence this clustering with pairwise constraints, has to select one of these algorithms. In addition, appropriate values have to be chosen for the algorithm hyperparameters. While these problems have received significant attention in the context of

supervised learning [3,16], little work has been done for clustering, both unsupervised and semi-supervised. In this paper we focus on semi-supervised clustering, which is closer to the well-studied supervised setting. The contributions of this paper are (a) a comparison of a diverse set of semi-supervised clustering algorithms on several UCI data sets and (b) the exploration of the semi-supervised clustering algorithm selection strategy based on constraint-set utility measures suggested in [18].

2 Semi-supervised clustering algorithms

Semi-supervised clustering algorithms can be broadly divided into three categories: methods that use the constraints to adapt their similarity measure, methods that adapt the actual clustering procedure to satisfy the constraints, and hybrid algorithms that combine these two approaches. In the remainder of this section, we briefly discuss these three approaches and the algorithms that we use in our experiments. We consider these algorithms in combination with hyperparameter selection methods, as ultimately we are interested in mappings of the following form:

$$\Gamma(\mathcal{X}, \mathcal{M}, \mathcal{C}) = y \tag{1}$$

with $\mathcal{X} = \{x_i\}_{i=1}^n$ the data set, $\mathcal{M} = \{(x_i, x_j)\}$ a set of must-link constraints, $\mathcal{C} = \{(x_i, x_j)\}$ a set of cannot-link constraints and $y = \{c_1, c_2, \ldots, c_K\}$ s.t. $\cup_i c_i = \mathcal{X}$ (we only consider partitional clusterings). Γ encapsulates the clustering method as well as the hyperparameter selection procedure.

2.1 Methods that adapt the clustering algorithm directly

The first category consists of methods that alter the clustering procedure to satisfy constraints. One such algorithm is COP-KMeans [19], an adaptation of the traditional K-Means algorithm in which points are only assigned to clusters if the assignment does not result in a constraint violation. Since the introduction of COP-K-Means, several other variants of the original K-means algorithm have been developed. Semi-supervised extensions have also been developed for other types of clustering algorithms, including density-based methods [10, 14] and spectral clustering algorithms [9, 12]. In the remainder of this section we discuss two such methods that are used in the experiments.

FOSC-OpticsDend
In [5] Campello et al. introduce FOSC, a "Framework for Optimal Selection of Clusters" from clustering hierarchies. Given a local unsupervised clustering quality measure (one that can be computed for each cluster individually) and a set of constraints, FOSC determines a local cut of a given hierarchy that is optimal with respect to the quality measure and the constraint set. The clustering hierarchy on which FOSC operates can be provided by any hierachical

clustering algorithm. In our experiments, these hierarchies will be provided by OPTICS, a density-based clustering algorithm (we use the implementation provided in the ELKI environment [1]). It produces a reachability plot, from which a dendrogram is constructed using the algorithm by Sander et al. [15]. Campello et al. also experiment with this combination and find that this approach, which they call FOSC-OpticsDend, outperforms SSDBSCAN [10], a semi-supervised extension of DBSCAN. OPTICS requires setting *minPts*, but as this parameter is non-critical it is common to fix its value for all runs [5, 10]. As in [5], we set it to $minPts = 4$. Often several cuts of the dendrogram yield a partitional clustering that respects all constraints. In this case it is the unsupervised quality measure that will determine the chosen cut. As in [5], we use cluster lifetime for this purpose, which can be seen as the length along the dendrogram for which a cluster exists.

Constrainted 1-Spectral Clustering
Constrained 1-Spectral Clustering (COSC) [12] is an extension of spectral clustering to the semi-supervised setting.[1] Spectral clustering methods aim to partition a similarity graph such that edges within clusters have high weights and edges between clusters have low weights [17]. Several types of similarity graphs can be constructed from a set of data points. In our experiments we use a symmetric K-NN graph with local scaling as in [4], which avoids the need to select a scaling parameter [21]. Parameter K, indicating the number of neighbors, is not critical for the clustering result, and we set it to $K = 10$, as in [4]. The resulting graph can be represented by an affinity matrix consisting of pairwise similarities. Some semi-supervised spectral clustering algorithms incorporate must-link and cannot-link constraints by modifiying this matrix directly. Others, such as COSC, adapt the optimization objective of spectral clustering to incorporate constraints and propose alternative optimization procedures. COSC requires setting the number of clusters, k. In our experiments we run COSC for $k \in [2, 10]$, and select the clustering that violates the lowest number of constraints. If multiple solutions score equally on this measure, the clustering with the smallest number of clusters is chosen.

2.2 Methods based on metric-learning

The second type of methods does not alter the clustering algorithm directly, but modifies the underlying similarity measure. One of the first such methods was proposed by Xing et al. [20], who introduce an algorithm to learn a Mahalanobis distance measure that minimizes the distance between pairs involved in must-link constraints, while keeping pairs involved in cannot-link constraints far apart. Since the work of Xing et al., many others have focused on learning Mahalanobis metrics, which can be defined as

$$d_A(x, y) = \sqrt{(x - y)^T A (x - y)} \tag{2}$$

[1] Code is available at http://www.ml.uni-saarland.de/code/cosc/cosc.htm

The formula simplifies to the Euclidean distance if A is the identity matrix. If a diagonal matrix A is learned, this corresponds to feature weighting. Using a full matrix corresponds to feature generation, with the newly generated features being linear combinations of existing ones [2]. Using a Mahalanobis metric defined by A is equivalent to using the Euclidean distance in the transformed space obtained by muliplying by $A^{1/2}$, i.e. $X_{\text{transformed}} = A^{1/2}X$. Note that adapting an algorithm's similarity metric can only modify the bias of a clustering algorithm to some extent. For example, with a Mahalanobis distance K-means can find parallel ellipsoidal clusters instead of only spherical ones, but still no non-parallel ellipsoidal or non-convex clusters.

Information-Theoretic Metric Learning
In our experiments, we use the Information-Theoretic Metric Learning (ITML) algorithm [7][2], which has been shown to outperform the earlier algorithm by Xing et al. [7,8]. ITML requires setting one hyperparameter, γ, which determines the importance of satisfying the constraints. For each data set and constraint collection, we learn Mahalanobis matrices for $\gamma \in \{.01, .1, 1, 10\}$ (the values suggested in [7]), and select the best one as the one that results in the clustering that violates the lowest number of constraints. If multiple solutions score equally, we select the one that corresponds to the lowest value of γ. We learn a full metric matrix A, transform the data using this matrix, and construct clusterings using one of the following unsupervised algorithms:

- **K-Means**: We run the traditional K-Means algorithm for $k \in [2, 10]$, and experiment with two ways of selecting the "best" solution from the generated candidates: (a) the one violating the lowest number of constraints, as before, or (b) the one with the highest silhouette index, an unsupervised measure [13]. In the latter case the silhouette index is calculated in the transformed space.
- **Self-Tuning Spectral Clustering**: We also apply Self-Tuning Spectral Clustering [21][3] to the transformed data, which does not require any parameters to be set. The affinity matrix is constructed using local scaling (as with the COSC experiments), and the number of clusters is determined by examining the structure of the eigenvectors of the Laplacian.

2.3 Hybrid methods

The last group consists of hybrid methods, which combine metric learning with adapting the clustering procedure. A common representative of this type of algorithms is MPCK-Means (Metric Pairwise Constrained K-Means, [2][4]). Briefly, MPCK-Means iterates between 1) assigning elements to clusters, in this step the within-cluster sum of squares is minimized but also constraint satisfaction

[2] Code is available at http://www.cs.utexas.edu/~pjain/itml/
[3] Code is available at http://www.vision.caltech.edu/lihi/Demos/SelfTuningClustering.html
[4] Code is available at http://www.cs.utexas.edu/users/ml/risc/code/

is incorporated 2) updating the means, just like in the traditional K-Means algorithm and 3) re-estimating the metric, as to minimize the objective function. Bilenko et al. [2] define several variants of this scheme. For example, one can learn a separate metric for each cluster, or a global one. These metrics can either be defined by full matrices, which corresponds to doing feature generation, or diagonal ones, which corresponds to feature weighting. In our experiments we use MPCK-Means with a single diagonal metric matrix. As before, we vary the number of clusters $k \in [2, 10]$ for each problem instance, and select the best solution as either the one that violates the lowest number of constraints, or the one that has the highest silhouette score.

3 Experiments

In this section we compare the performance of the previously discussed algorithms on several UCI data sets.

3.1 Overview of algorithms and experimental methodology

In total, we compare 7 clusterers:

- **FOSC-OpticsDend**
- **COSC**: Constrained 1-Spectral Clustering, selecting k based on the constraints
- **K-Means-ITML-NumSat**: K-Means on ITML transformed data, selecting k and γ based on the constraints
- **K-Means-ITML-Silhouette**: K-Means on ITML transformed data, selecting k and γ based on the silhouette index
- **Self-Tuning-Spectral-ITML-NumSat**: Spectral clustering, selecting γ based on the constraints
- **MPCK-Means-NumSat**: MPCK-Means, selecting k based on the constraints
- **MPCK-Means-Silhouette**: MPCK-Means, selecting k based on the silhouette index

For each data set, we also show the results of four unsupervised variants of the algorithms:

- **K-Means-Unsup**: the traditional K-Means algorithm, selecting k based on the silhouette index
- **Self-Tuning-Spectral-Unsup**: Self-Tuning-Spectral clustering on the original data
- **FOSC-OpticsDend-Unsup**: extraction of a partitional clustering from the OPTICS dendrogram based on the unsupervised cluster lifetime measure
- **MPCK-Means-Unsup**: running MPCK-Means without constraints, which is different from the traditional K-Means algorithm, as MPCK-Means also performs unsupervised metric learning

For each constraint set size in $\{25, 100, 400, 1600\}$, we generate 10 constraint sets in the following way:

1. We select 70% of the data set randomly
2. From this subset, pairs of elements are randomly selected and a must-link or cannot-link constraint is added depending on whether the selected elements belong to the same class or not.

Clusterings are evaluated using the adjusted Rand Index (ARI) [11], which is calculated using only the elements of the 30% of the data that were not selected in step one. The first step ensures that there will be enough elements (i.e. at least 30% of the data) to evaluate the clusterings on. For larger numbers of constraints and relatively small data sets, it might otherwise occur that all elements are involved in constraints, leaving none for evaluation.

3.2 Results and discussion

Figures 1 and 2 show the results of our experiments for the *iris* and *wine* data sets. Similar figures for the *dermatology, column, glass* and *ecoli* data sets are added in the appendix. For the *iris* data set, three algorithms show consistent average improvement when constraints are added: FOSC-OpticsDend, K-Means-ITML-NumSat and MPCK-Means-NumSat. In general, for *iris* the performance seems to improve with constraints, or does at least not decline drastically (e.g. for K-Means-ITML-Silhouette the performance remains largely constant).

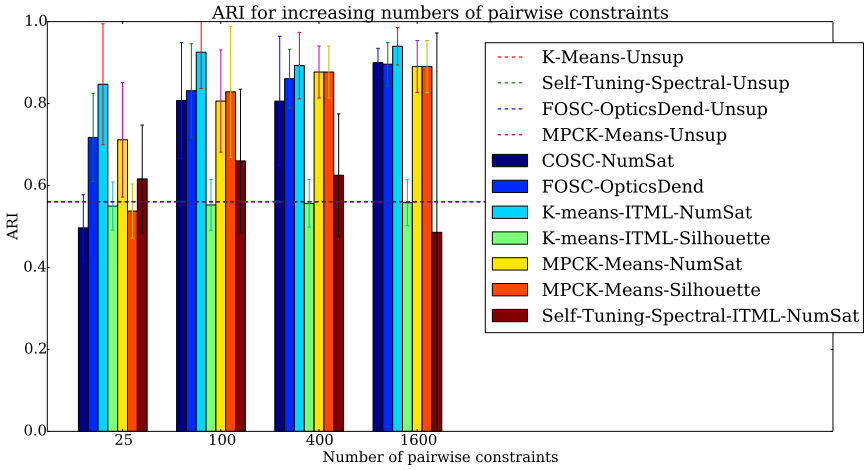

Fig. 1. Scores for the *iris* data set (150 instances, 4 features, 3 classes). All unsupervised algorithms produced the same clustering for this data set (only the MPCK-Means line is visible in the plot, as they all overlap).

Figure 2 shows that the results are quite different for the *wine* data set. Most clustering algorithms already attain a high ARI score without any constraints, and in many cases adding constraints leads to an average decrease in performance. For this data set, MCPK-Means-Silhouette and Self-Tuning-Spectral-ITML seem to be the most robust in this aspect.

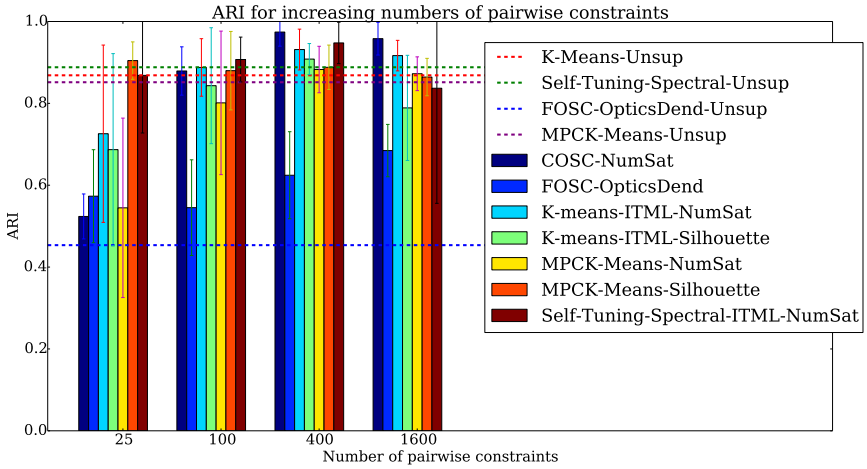

Fig. 2. Scores for the *wine* data set (178 instances, 13 features, 3 classes)

In general, from the results on these two data sets and the four others that are added in the appendix, it is clear that no single semi-supervised clustering algorithm outperforms all others in all scenarios. The preferable option for a certain task depends on the data set, the size of the constraint set, and even the specific constraint set under consideration. Often, the preferrable option even seems to be to not make use of the constraints at all. This is quite counter-intuitive, as one would expect constraints to "point the algorithm in the right direction". Davidson et al. [6, 18] also observe this potentially negative effect of adding constraints. They point out that, while performance improves *on average* when more constraints are provided, individual constraint sets can have a detrimental effect. In our experiments this happened frequently: clustering performance decreased when constraints were added for 45% of the considered runs (with a run we indicate a data set, particular constraint set and clusterer combination).

Constraint set consistency and coherence
The observation that adding constraints can result in decreased performance is of course crucial, as we are mostly interested in the performance gain that we can obtain with one particular constraint set. To provide insight into this behaviour, Davidson et al. propose two measures that characterize the utility of constraint sets [6]:

- **Consistency** is defined by the number of constraints that are satisfied by the clustering produced without any constraints. It is a property of both the constraint set and the algorithm producing the clustering.
- **Coherence** measures the amount of agreement between constraints. In [18], Wagstaff et al. define two variants of this measure: distance coherence and direction coherence. We use the former one in our experiments, which is defined as the fraction of constraint pairs (one ML and one CL constraint) for which the distance between the points in the ML constraint is greater than the distance between the points in the CL constraint [18]. This property only depends on the data set and the constraints.

Wagstaff et al. show that consistency and coherence are strongly correlated with clustering performance: if a consistent and coherent constraint set is provided, performance is likely to increase, whereas inconsistent and incoherent constraint sets have an adverse effect. They study these properties in the context of MPCK-Means and variants thereof. Here, we verify whether this correlation also holds for the diverse set of clustering algorithms used in our experiments. We perform an analysis similar to the one in [18]. If the property holds, it may provide insight into whether a particular constraint set should be used or not in combination with a clustering algorithm.

Figure 3 shows the relation between the constraint utility measures and ARI, illustrating that also in our experiments these are strongly correlated (Pearson coefficient of 0.66 for consistency, Pearson coefficient of 0.75 for coherence). Table 1 shows the correlation coefficients for the separate algorithms, providing insight into the sensitivities of the different algorithms to constraint set inconsistency and incoherence. For example, the correlation between consistency and performance is larger for the K-Means-ITML algorithms than for the MPCK-Means algorithms, meaning that consistency might be a better predictor for performance for the former ones. The correlation between consistency and performance is lowest for COSC, but this can be explained by the fact that we used the outcome of the Self-Tuning spectral clustering algorithm as an unsupervised baseline for COSC, as otherwise we would not be able to select the number of clusters k, which COSC requires.

Consistency-based algorithm selection
Given the correlation between consistency and clustering performance, Wagstaff et al. [18] suggest the simple algorithm selection strategy of choosing the one with the highest consistency, given a data set and constraints. We explore the

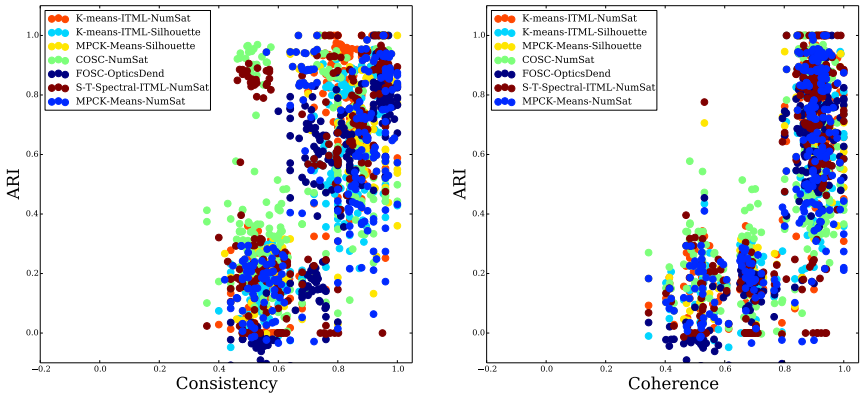

Fig. 3. (left) Consistency vs. ARI, Pearson correlation coefficient = 0.66 , (right) Coherence vs. ARI, Pearson correlation coefficient = 0.75

Algorithm	**Consistency vs. ARI**	**Coherence vs. ARI**
FOSC-OpticsDend	0.81	0.88
COSC-NumSat	0.45	0.62
K-Means-ITML-NumSat	0.83	0.82
K-Means-ITML-Silhouette	0.82	0.76
Self-Tuning Spectral-ITML-NumSat	0.56	0.69
MPCK-Means-NumSat	0.63	0.72
MPCK-Means-Silhouette	0.67	0.80

Table 1. Pearson correlation coefficients of consistency and coherence vs. ARI for each algorithm

effectiveness of this strategy by applying it to our clustering experiments. For each problem instance, which consists of a data set and constraints, we compute the relative score for each algorithm as its ARI for this instance divided by the largest ARI obtained for that problem instance by all algorithms. We compute the average relative score for each algorithm over all 1680 problem instances (6 data sets, 7 algorithms, 4 constraint set sizes and 10 constraint sets per size). These averages are shown in Table 2. The table also shows the average relative score that is obtained by using the consistency-based algorithm selection strategy. It is clear that, despite the observed correlation between consistency and clustering performance, the algorithm selection strategy does not perform well, as we would be better of by simply picking the (on average) best algorithm for each problem instance, which is MPCK-Means-Silhouette.

Algorithm	Avg. rel. score
MPCK-Means-Sil	0.814
COSC-NumSat	0.798
K-Means-ITML-NumSat	0.770
ST-Spectral-ITML-NumSat	0.725
Consistency-based AS	**0.713**
MPCK-Means-NumSat	0.712
K-Means-ITML-Sil	0.695
FOSC-OpticsDend	0.638

Table 2. Average relative scores for each algorithm, and also for the consistency-based algorithm selection strategy (AS)

Data set	Consistency vs. ARI	Coherence vs. ARI
iris	0.20	-0.14
ecoli	-0.06	0.04
column_2C	-0.17	-0.19
glass	-0.12	-0.10
dermatology	0.16	-0.07
wine	0.04	-0.07

Table 3. Pearson correlation coefficients of consistency and coherence vs. ARI for each data set

Why consistency-based algorithm selection does not work

In this section we explain the unsatisfactory results of the consistency-based algorithm selection strategy, and in doing so identify an important property of the consistency and coherence measures. We observe that, while these measures correlate strongly with performance if we look at all problem instances combined, this correlation disappears when we consider the data sets separately. This can be seen in Figure 4, which is similar to Figure 3 but colored by data set instead of algorithm. This visual observation is confirmed by looking at the correlation coefficients for each data set separately, shown in Table 3. These results indicate that, while consistency and coherence can be indicative of the difficulty of clustering a particular data set given constraints, these measures cannot be used to decide between clustering algorithms for a given data set.

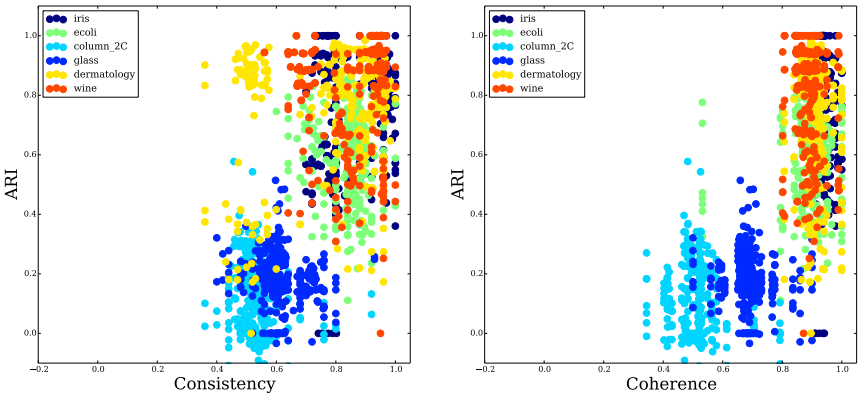

Fig. 4. (left) Consistency vs. ARI (right) Coherence vs. ARI (same as Figure 3, but colored by data set)

Furthermore, for a consistency-based algorithm selection strategy to be succesful, the different clusterers should produce significantly different clusterings when no constraints are given. We verify whether this is actually the case for the algorithms and data sets that are considered in the experiments, by comparing the unsupervised clusterings using the adjusted Rand index (ARI). While ARI is mostly used to compare a produced clustering to a ground truth one, it can more generally be used to measure the similarity between any two clusterings. When no constraints are given, all algorithms produce the same solution for the *iris* data set (pairwise ARI scores of 1.0). Also for the *wine* and *ecoli* data sets all clusterings are quite similar (ARIs > 0.9), except for the ones generated by FOSC-OpticsDend (ARIs < 0.5). The clusterings generated for the *dermatology*, *glass* and *column* data sets are more diverse (with an average pairwise ARI of 0.49, not taking into account the similarities between the K-Means and MPCK-Means solutions, which are more similar with an average pairwise ARI of 0.95). These observations complement the lack of correlation between consistency and ARI in explaining the failure of the consistency-based algorithm selection strategy for the *iris*, *ecoli* and *wine* data sets.

4 Conclusions

Semi-supervised clustering is a popular research topic, and its usefulness has been demonstrated in several practical applications. Most major clustering algorithms have been extended to incorporate domain knowledge, often in the form of must-link and cannot-link constraints. It has been frequently demonstrated that, on average, performance increases when constraints are added. However, it is known that individual constraint sets can harm perfomance. We have experimented with a diverse set of semi-supervised clustering algorithms, and have observed that this is indeed often the case. Given data to cluster and a set of constraints, a user then has to determine which semi-supervised clustering algorithm to use, if any. Previous work proposed to use constraint set consistency and coherence for this purpose, two constraint set utility measures that were shown to correlate strongly with clustering performance. We have experimented with such consistency-based algorithm selection, but found it to be unsuccessful. For some data sets, these results can be explained by the similarities between the clusterings that are produced when no constraints are given. If these similarities are high, comparing the corresponding consistency values is not informative. More importantly, we also explain the unsatisfactory results of the selection strategy for data sets for which the clusterings produced without constraints are more diverse. We do this by showing that the utility measures only correlate strongly with clustering performance if we look at problem instances from several data sets combined, and that this correlation disappears when we consider individual data sets. These results severely restrict the use of these measures in semi-supervised clustering, as practitioners are mainly interested in applying them in the context of a particular data set.

Acknowledgements

Toon Van Craenendonck is supported by the Agency for Innovation by Science and Technology in Flanders (IWT).

References

[1] Elke Achtert, Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek. Interactive data mining with 3d-parallel-coordinate-trees. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 1009–1012, 2013.

[2] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 11–, New York, NY, USA, 2004. ACM.

[3] Pavel B. Brazdil, Carlos Soares, and Joaquim Pinto da Costa. Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277, 2003.

[4] Thomas Bühler and Matthias Hein. Spectral clustering based on the graph p-laplacian. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 81–88, New York, NY, USA, 2009. ACM.

[5] 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.

[6] Ian Davidson, Kiri L. Wagstaff, and Sugato Basu. Measuring constraint-set utility for partitional clustering algorithms. In *Knowledge Discovery in Databases: PKDD 2006*, volume 4213 of *Lecture Notes in Computer Science*, pages 115–126. Springer Berlin Heidelberg, 2006.

[7] 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.

[8] Amir Globerson and Sam T. Roweis. Metric learning by collapsing classes. In Y. Weiss, B. Schölkopf, and J.C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 451–458. MIT Press, 2006.

[9] Sepandar D. Kamvar, Dan Klein, and Christopher D. Manning. Spectral learning. In *In IJCAI*, pages 561–566, 2003.

[10] Levi Lelis and Jörg Sander. Semi-supervised density-based clustering. In *ICDM '09. Ninth IEEE International Conference on Data Mining, 2009*, pages 842–847, Dec 2009.

[11] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[12] Syama S. Rangapuram and Matthias Hein. Constrained 1-spectral clustering. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, volume 22, pages 1143–1151, 2012.

[13] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[14] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. C-dbscan: Density-based clustering with constraints. In *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, volume 4482 of *Lecture Notes in Computer Science*, pages 216–223. Springer Berlin Heidelberg, 2007.

[15] Jörg Sander, Xuejie Qin, Zhiyong Lu, Nan Niu, and Alex Kovarsky. Automatic extraction of clusters from hierarchical clustering representations. In *Proceedings of the 7th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '03, pages 75–87, Berlin, Heidelberg, 2003. Springer-Verlag.

[16] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, page 847–855, 2013.

[17] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[18] Kiri Wagstaff, Sugato Basu, and Ian Davidson. When is constrained clustering beneficial, and why? In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, 2006.

[19] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[20] Eric P. Xing, Michael I. Jordan, Stuart Russell, and Andrew Y. Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002.

[21] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. *Advances in Neural Information Processing Systems 17*, 2:1601–1608, 2004.

A Performance comparison for UCI data sets

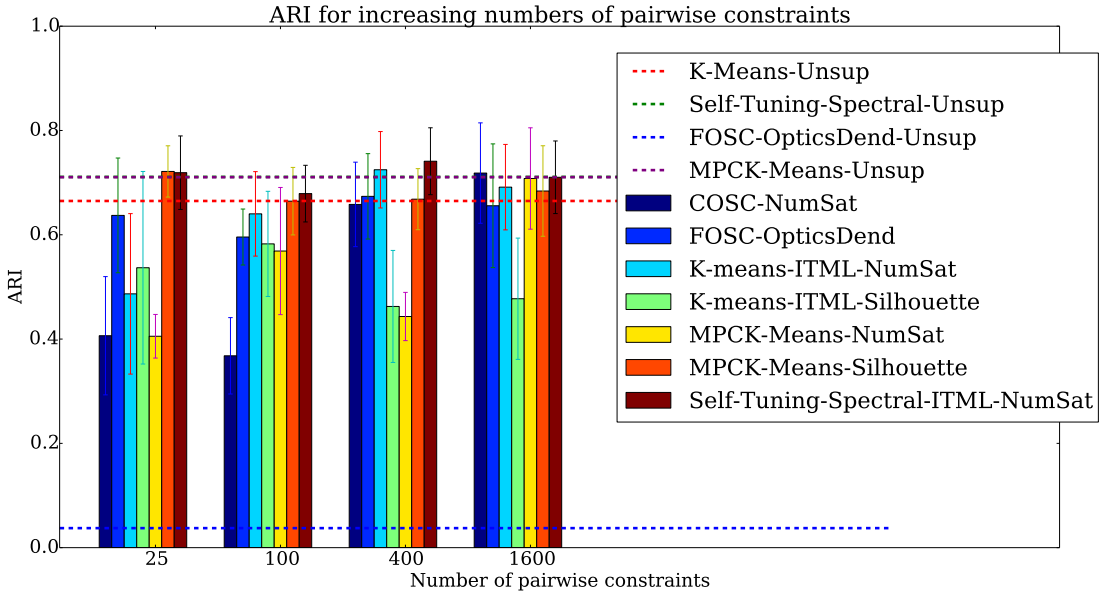

Fig. 5. Scores for the *ecoli* data set (336 instances, 7 features, 8 classes)

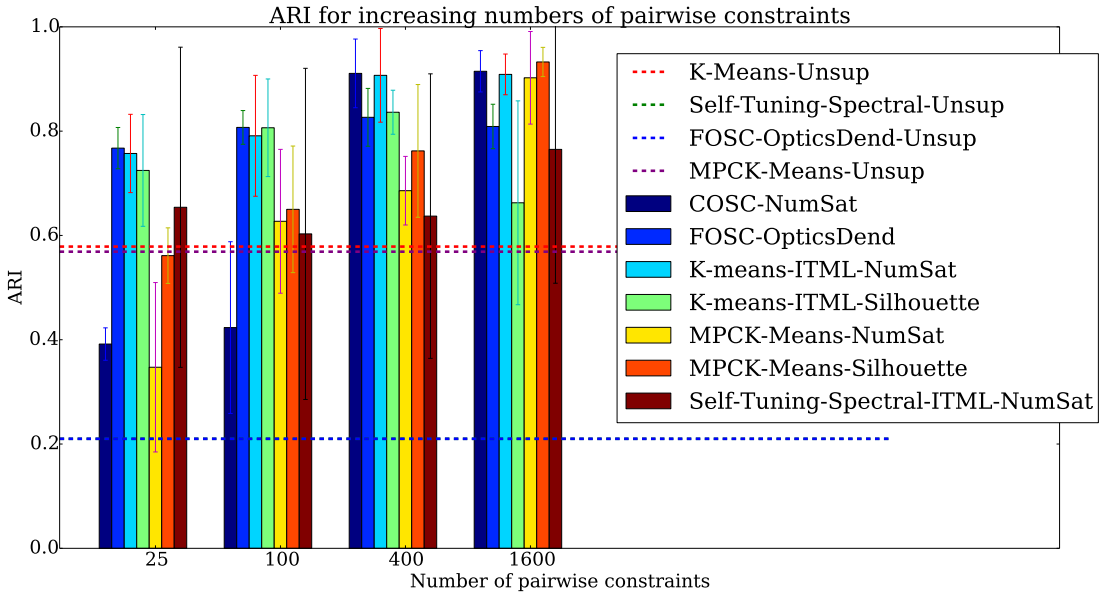

Fig. 6. Scores for the *dermatology* data set (366 instances, 34 attributes, 6 classes)

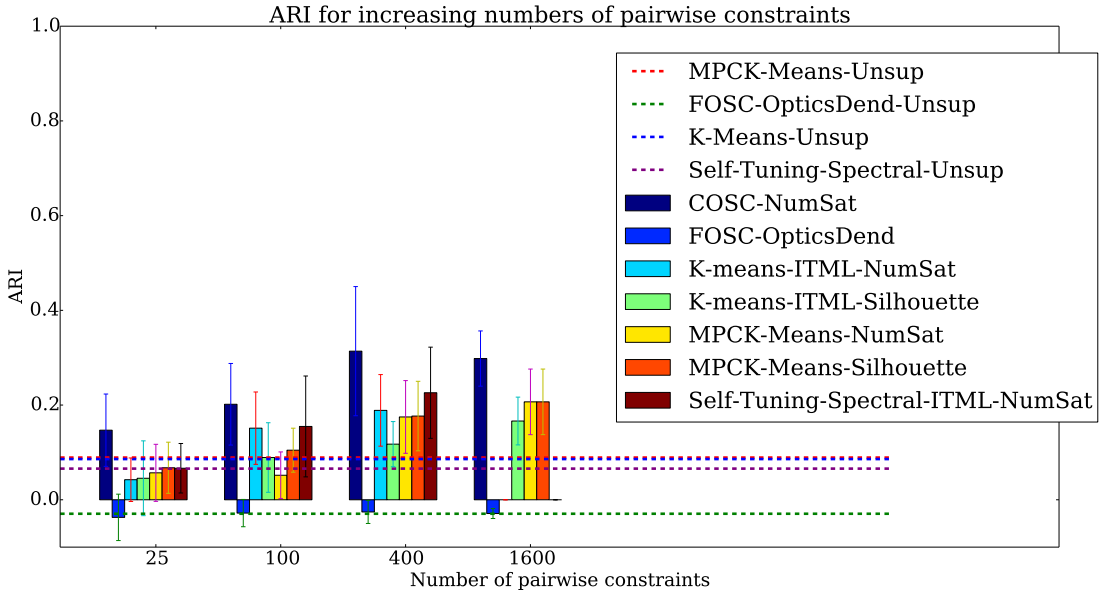

Fig. 7. Scores for the *column* data set (310 instances, 6 attributes, 2 classes)

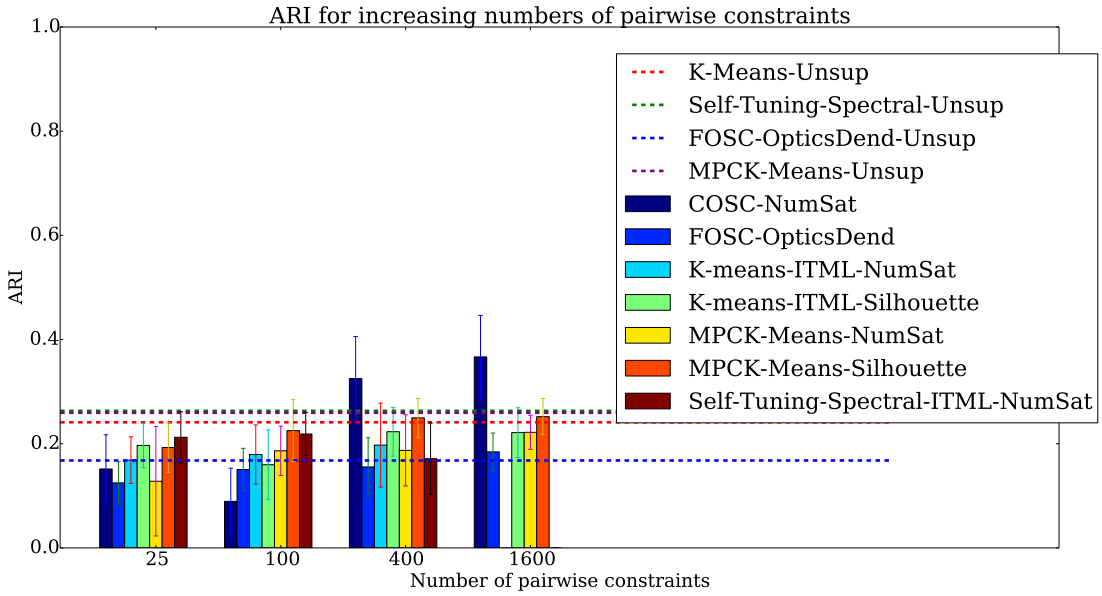

Fig. 8. Scores for the *glass* data set (214 instances, 9 attributes, 6 classes)