A Constraint-based Approach for Multispace Clustering

Ruggero G. Pensa and Mirco Nanni

Pisa KDD Laboratory, ISTI-CNR, I-56124 Pisa, Italy
{ruggero.pensa,mirco.nanni}@isti.cnr.it

Abstract. In many applications, a set of objects can be represented by different points of view (universes). Beside numeric, ordinal and nominal features, objects may be represented using spatio-temporal information, sequences, and more complex structures (e.g., graphs). Learning from all these different spaces is challenging, since often different algorithms and metrics are needed. In the case of data clustering, a partitional, hierarchical or density-based algorithm is often well suited for a specific type of data, but not for other ones. In this work we present a preliminary study on a framework that tries to link different clustering results by exploiting pairwise similarity constraints. We propose two algorithmic settings, and we present an application to a real-world dataset of trajectories.

1 Introduction

Exploratory data analysis processes are often based on clustering methods to get insights about global patterns that hold in the data. A clustering task provides a partitioning of objects and/or of attributes such that a grouping quality measure is optimized. Depending on the type of data one wants to analyze, different kinds of metrics and clustering approaches have been proposed to provide accurate clustering. However, in many applications, a set of objects can be represented by different points of view. For instance, in a traditional basket analysis setting, a customer can be characterized by the items he/she purchases, or by census data (age, sex, income). In bioinformatics, proteins are represented as sequences of amino acids, or by their 2D (or 3D) structure. Each point of view defines a space, and each space has its own metrics and statistics. Defining a metric which takes into account different spaces is challenging. Discretization is a straightforward way to combine numeric, ordinal and nominal features, but it does not allow to integrate different kinds of information, such as spatio-temporal information, sequential events related to the object, structural information (e.g., graphs). Given a clustering algorithm, it is possible to use different kinds of features by combining the (weighted) distances computed on different spaces. This solution has two main drawbacks. First, it is not easy to determine a good weight for each distance component. Secondly, it is hard to use the same clustering algorithm for all the feature spaces. Some algorithms could be good for a feature space, but their results could be weak for other feature spaces, or, at worst, some algorithms might be not applicable at all. To better motivate this claim, we will consider three typical KDD applications.

Document clustering. Documents can be represented in different ways. One of the most commonly used representations is the classic *documents* × *terms* scheme. Each document is represented in the term vector space, where each term is a feature, and the value of each feature is the (normalized) number of occurrences of the corresponding term in the current document. A popular task in document clustering consists in grouping similar documents w.r.t. the terms they contain. Another way to represent documents is growing in popularity. Thanks to the availability of a huge number of digitalized documents, it is now rather simple to obtain the document structure as a graph (and, more precisely, in form of a tree). Graph mining and clustering is a rising topic in document KDD, and a lot of works exist that try to define suitable distance metrics (e.g., graph edit distance) for standard data mining tasks. Another way to look at a document is to consider some other features which are context-free (such as length, type, number of authors) or context-dependent (e.g., topics). Each of these representations is a space, and might need different types of algorithms and distance metrics.

Transcriptomic/Proteomic analysis. A typical bioinformatics research issue concern the analysis of gene expression data. Gene expression data are in form of matrices where the expression level of a set of genes (rows) is measured in a set of biological samples or conditions (columns). A typical gene expression analysis task consists in clustering genes according to their expression profiles, and/or conditions. Some typical distance metrics are the Euclidean distance, the Pearson correlation, or the sum-squared residues. However, genes can also be viewed as sequences of the four nucleotides A, C, G and T. In sequence mining, the string edit distance is often used to find similar sequences. Other kinds of features can be associated to each genes, such as, gene functions or transcription factors activating them. Proteins, instead, can be viewed as sequences of amino acids, and then processed as strings (edit distance). Or one may consider the 2-dimensional structure, and this leads to the particular case of graph clustering. Also protein microarrays are often used to identify protein-to-protein interactions. Each one of these views over the same set of objects can be represented in a distinct feature space.

Trajectory clustering. Trajectories are often represented as sequences of triplets identifying a time instant and the (x, y) coordinates in the 2-dimensional space. Typically, the distance between two trajectories is computed by fixing two time instants and considering points within this interval. Clustering this kind of data, obviously tends to produce clusters containing geographically close trajectories. We can also look at trajectories as sequences (strings) of events (e.g., "turn right", "u-turn", "straight on", "prolonged stop"). In this case, trajectories which are clustered together are geometrically similar, even if they are not located on the same region of the space. A third point of view of a trajectory database is related to the regions of interest covered by each single trajectory. Even in this case trajectories are viewed as sequences of events, but the size of the alphabet could be larger than in the previous example. Finally, each trajectory can be characterized by different kinds of features, such as information about the driver, information about the type of vehicle (e.g., truck, car).

In [18] the problem of clustering in *Parallel Universes* is introduced, and a clustering algorithm is proposed that operates on different feature spaces. The algorithm uses a fuzzy *c-Means* strategy, where each object has a different cluster membership value for each space. Then, some objects can be ignored in some feature spaces. In [4] the problem of multi-view clustering is presented. The authors show that considering two views of the same set of objects leads to better results than considering one view at a time. In this paper we present a framework that extends these ideas to the context of heterogeneous representation spaces, each of them possibly requiring a different clustering algorithm. Each space is clustered separately, but it exploits the local results of other clustering algorithms on the other representation spaces to guide the grouping task thanks to a constrained-based approach. In this preliminary work we introduce two possible strategies for clustering data represented in different spaces, and we present an instance of our framework in the case of trajectory cluster analysis.

The paper is organized as follows: Section 2 introduces our framework. Section 3 analyzes a practical application of our clustering framework for trajectories. Section 4 discusses some open problems and perspectives of our framework.

2 Problem setting

We consider a set of N objects $X = \{x_1, \dots, x_N\}$ and its U representations $\mathcal{X}_1 \dots \mathcal{X}_U$ (called spaces or universes). Each space is defined individually, depending on its type. For instance, given an object x_i and a representational space u:

- $x_{i,u}$ can be represented by a vector $\boldsymbol{x}_{i,u} =< x_{i,u,1}, \dots, x_{i,u,M_u} >$ where M_u is the dimensionality of the vector space u and $x_{i,u,j} \in \mathbb{R}$;
- $x_{i,u}$ can be represented by a graph $G_{i,u}(V_{i,u}, E_{i,u})$, where $V_{i,u} = \{v_{i,u,1}, \dots, v_{i,u,NV_{i,u}}\}$ is a set of vertices, and $E_{i,u} = \{e_{i,u,1}, \dots, e_{i,u,NE_{i,u}}\}$ is a set of edges ($NV_{i,u}$ and $NE_{i,u}$ being the number of vertices and edges respectively);
- $x_{i,u}$ can be represented by a sequence $S_{i,u} =< s_{i,u,1} s_{i,u,2} \dots s_{i,u,M_{i,u}} >$ where $s_{i,u,j} \subseteq I_u$, I_u is a set of items, and $M_{i,u}$ is the length of the $i-th$ sequence.
- $x_{i,u}$ is a trajectory $T_{i,u} =< s_{i,u,1} s_{i,u,2} \dots s_{i,u,M_{i,u}} >$ where $s_{i,u,j}$ is a triplet $(x_{i,u,j}, y_{i,u,j}, t_{i,u,j})$ s.t. $(x_{i,u,j}, y_{i,u,j})$ is a point in $\mathbb{R}^2$, $t_{i,u,j}$ is a timestamp and $M_{i,u}$ is the length of the $i-th$ trajectory.

Other possible representations are geographic regions, images, item sets, strings, and so on.

The goal of our approach is to partition X into U sets Π_u $(u = 1 \dots U)$ of K clusters $(\Pi_u = \{\pi_{1,u}, \dots, \pi_{K,u}\})$. The value of K is not specified a priori.

Before describing our framework for multispace clustering we introduce the semi-supervised clustering approach which is at the basis of our technique. The goal of semi-supervised clustering is to use some a priori knowledge on the data to guide the clustering process and get better partitions w.r.t. the available knowledge. The easiest way to represent knowledge on data is by defining similarities and dissimilarities between pair of objects. If we know that objects x_i and x_j are similar, or that they belong to the same class, we can enforce x_i and x_j to be in the same cluster. Analogously, if they are dissimilar, or they belong to two distinct classes, we can enforce them to be in two distinct cluster at the end of the clustering process. Similarities and dissimilarities are usually expressed by the following pairwise constraints.

Definition 1 (must-link/cannot-link). *If two objects x_i and x_j are involved in a* ***must-link*** *constraint, denoted $c_=(x_i, x_j)$, they must be in the same cluster. If two objects x_i, x_j are involved in a* ***cannot-link*** *constraint, denoted $c_{\neq}(x_i, x_j)$, they cannot be in the same cluster.*

The ways a semi-supervised clustering algorithm processes these constraints are essentially two. In the so-called metric-based approaches, the distance function is trained to fit the set of must-link and cannot-link constraints, and then is used during the clustering process. In the so-called constraint-based approach, the distance function takes into account constraint violation as a penalty addendum, or, alternatively, objects are assigned to cluster centroids avoiding constraint violation.

Our key idea is that, given two representations $\mathcal{X}_1$ and $\mathcal{X}_2$ of the same set of objects X, we can supervise the clustering process on the space $\mathcal{X}_1$ through the results of the clustering process on $\mathcal{X}_2$ (or viceversa). In particular, given the partition $\Pi_1 = \{\pi_{1,1}, \dots, \pi_{K,1}\}$ resulting from the clustering process of X on the representation $\mathcal{X}_1$, we pick P pairs of objects and, for each pair, we generate a must-link constraint if both objects are in the same cluster of Π_1, otherwise we generate a cannot-link constraint. We call $\mathcal{C}_= = \{c_=(x_i, x_j)\}$ the set of must-link constraints and $\mathcal{C}_{\neq} = \{c_{\neq}(x_i, x_j)\}$ the set of cannot-link constraints ($|\mathcal{C}_=| + |\mathcal{C}_{\neq}| = P$).

```
Algorithm 1: MS3Clust($X$, $\mathcal{X}_1, \dots, \mathcal{X}_U$, $P_1, \dots, P_{U-1}$, $K_1, \dots, K_U$)
  Input: Set of objects $X$, representations $\mathcal{X}_1, \dots, \mathcal{X}_U$, integers $P_1, \dots, P_{U-1}$,
         integers $K_1, \dots, K_U$
  Output: Partitions $\Pi_u$
  $\mathcal{C}_{=,0} = \{\emptyset\}$;
  $\mathcal{C}_{\neq,0} = \{\emptyset\}$;
  for $u = 1 \dots U$ do
      if $u > 1$ then
          GenerateConstraints($X$, $\Pi_{u-1}$, $P_{u-1}$, $\mathcal{C}_{=,u}$, $\mathcal{C}_{\neq,u}$);
      end
      $\Pi_u$=ConsClustering($X$, $\mathcal{X}_u$, $K_u$, $\mathcal{C}_{=,u}$, $\mathcal{C}_{\neq,u}$);
  end
```

Algorithm 1 sketches the generic approach for a set of U representations (universes) of X. It starts with an unconstrained clustering on the first space, and then, at each iteration u, it first generates a set of must-link constraints $\mathcal{C}_{=,u}$ and a set of cannot-link constraints $\mathcal{C}_{\neq,u}$ using the partitions obtained at the $(u-1)$-th iteration, then it executes a semi-supervised clustering algorithm which is specific for the representation X_u. Function GenerateConstraints generates a specified number of pairwise constraints by choosing the pairs randomly and determining the constraint type looking if the cluster labels of the two objects are the same or not. The number of pairwise constraints generated at each iteration is trigged through a user-defined parameter P_u, while K_u is the desired number of clusters (possibly ignored by clustering algorithms that do not need such a parameter). Notice that K_u parameters may contain distinct values, since the supervised part of the clustering algorithm takes into account the pairs of similar/dissimilar objects, rather than the cluster labels. Notice also that the results depend on the values of parameters P_u, and on the order in which the spaces are processed. Notice also that, in most cases, the value of a parameter K_u should be at least equal to the value of the parameter K_{u-1}, otherwise the set of cannot-link constraints might not be satisfied.

Algorithm 2 presents a solution for reducing the influence of the particular chosen representation order. It is an iterative version of Algorithm 1 where the representations are processed in a different (random) order at each iteration.

```
Algorithm 2: IMS3Clust($X$, $\mathcal{X}_1, \ldots, \mathcal{X}_U$, $P_1, \ldots, P_{U-1}$, $K_1, \ldots, K_U$, $I$)
Input: Set of objects $X$, representations $\mathcal{X}_1, \ldots, \mathcal{X}_U$, integers $P_1, \ldots, P_{U-1}$,
       integers $K_1, \ldots, K_U$, integer $I$
Output: Partitions $\Pi_u$
$t = 0$;
$\mathcal{C}_{=,0} = \{\emptyset\}$;
$\mathcal{C}_{\neq,0} = \{\emptyset\}$;
for $i = 1 \ldots I$ do
    for $u = 1 \ldots U$ do
        Pick a representation $\mathcal{X}_t$ randomly;
        if $t > 0$ then
            GenerateConstraints($X$, $\Pi_{t-1}$, $P_{t-1}$, $\mathcal{C}_{=,t}$, $\mathcal{C}_{\neq,t}$);
        end
        $\Pi_t$=ConsClustering($X$, $\mathcal{X}_t$, $K_t$, $\mathcal{C}_{=,t}$, $\mathcal{C}_{\neq,t}$);
        $t = t + 1$;
    end
end
```

3 A case study

In this section we present an instantiation of our multispace clustering framework. We consider the problem of clustering trajectories which are represented in two distinct spaces. The first one (called $\mathcal{X}_{2D}$) is a 2-dimensional geographic space, where trajectories are represented by sequences of 2-dimensional points. The second space (called $\mathcal{X}_F$) is defined as a set of M numerical features which describe some trajectory properties. We choose to use a density-based approach to cluster trajectories in the 2-dimensional geographic space, and the well-known k-means algorithm to cluster them in the numeric M-feature space. In this case study, we use the non-iterative version of our approach. Our framework is then instantiated as follows: we use the clustering results of the density-based approach on $\mathcal{X}_{2D}$ to generate a set of pairwise constraints. These constraints are then used by a constraint-based version of the k-means approach. In the following the two algorithms adopted are described in more detail.

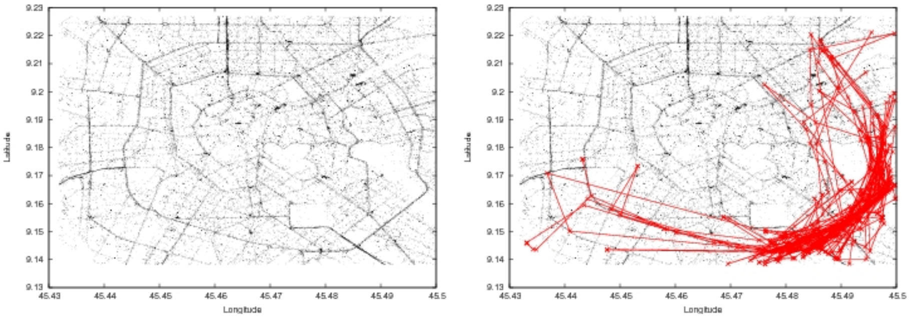
The first approach is a simplified, greedy density-based clustering algorithm for trajectory data. The approach iteratively selects a random trajectory τ and check whether there exist at least k_{min} trajectories not farther than a threshold δ from it. The k_{min} and δ parameters express the minimum "density" of trajectories required around τ. Then, if τ results to be "dense", all trajectories at distance δ or less from τ are grouped to form a cluster, and are removed from the input dataset before passing to the next iteration of the algorithm. At the end of the computation, the trajectories not assigned to any cluster are considered as noise and simply removed. The algorithm is parametric on the trajectory distance function adopted. In our experiments we used the EDR distance [7], essentially consisting of a variant of the classical edit distance. The clustering schema described above has already been used in [1], to group trajectories (with a different distance function) for anonymization purposes.

The second clustering algorithm is MPCK-Means [5], a constraint-based algorithm which integrates both constraint processing and metric learning [5]. Constraint processing is realized by adding some penalties to the Euclidean distance in case of constraint violation. The metric learning step consists in adapting the parametric Euclidean distance function by estimating the set of weights.

3.1 Experimentation setting

We applied our framework to a dataset describing the movement of several vehicles in the city center of Milan, Italy, during one day. The dataset contains around 3800 trajectories having an average length of 17 points. In Figure 1(a) the set of single space locations contained in the dataset is plotted.

The experimentation was carried out by representing each trajectory on two distinct spaces. The first space consists of the original trajectories, i.e., the full sequences of spatial coordinates. The second one describes the motion of each trajectory by means of four features: total distance travelled, average speed,



(a) 2D projection of Milan GPS data

(b) Sample spatial cluster

Fig. 1. Input trajectory data and sample cluster (spatial projection)

average number of stops performed per km, number of stops per hour. In particular, a stop occurs when the speed of a trajectory is below a given threshold. In our experiment it was set to 5 mt/sec.

Similar trajectories in the first space are characterized by the fact that they share some segment of the paths they follow (though the timings can be very different). That is visible, for instance, in the sample cluster shown in Figure 1(b), obtained by clustering over the space of trajectories. Similar trajectories in the second space, instead, have similar general behaviours, with no direct relation to the geographical location. For instance, two trajectories can have similar speeds while moving in completely different parts of the city.

3.2 Results

A comparison of the results obtained by clustering on the first domain only, on the second domain only and then on both by means of the schema in Algorithm 1 (using a set of 300 pairwise constraints), reveals several differences in the output and provides some insights about the effects of the spatial clustering-driven constraints over the feature-based clustering step.

In particular, we executed 20 randomly initialized instances of the MPCK-Means and the *k-means* algorithm and we chose the best result w.r.t. the objective function. Then, we compared these two results with the 20 resulting partitions produced by *k-means* and with the 20 results produced by MPCK-Means. To measure the partition similarity we used the Adjusted Rand Index, defined in [10]. The results (see Table 3.2) show that the partitions we discovered with our approach on $\mathcal{X}_F$ are quite different from those that have been discovered by the unconstrained *k-means*, even if the Adjusted Rand Index is still high. We analyze now these differences.

In Figure 2 we show the effects of combining the two clustering methods from a different perspective, by comparing the tree variations of a cluster obtained in three contexts: (1) over the trajectory domain; (2) over the feature domain, with no information about the clusters found on trajectory data; (3) over the

Selected results	No constraints	300 constraints
No.const.	0.764	0.688
300const.	0.619	0.798

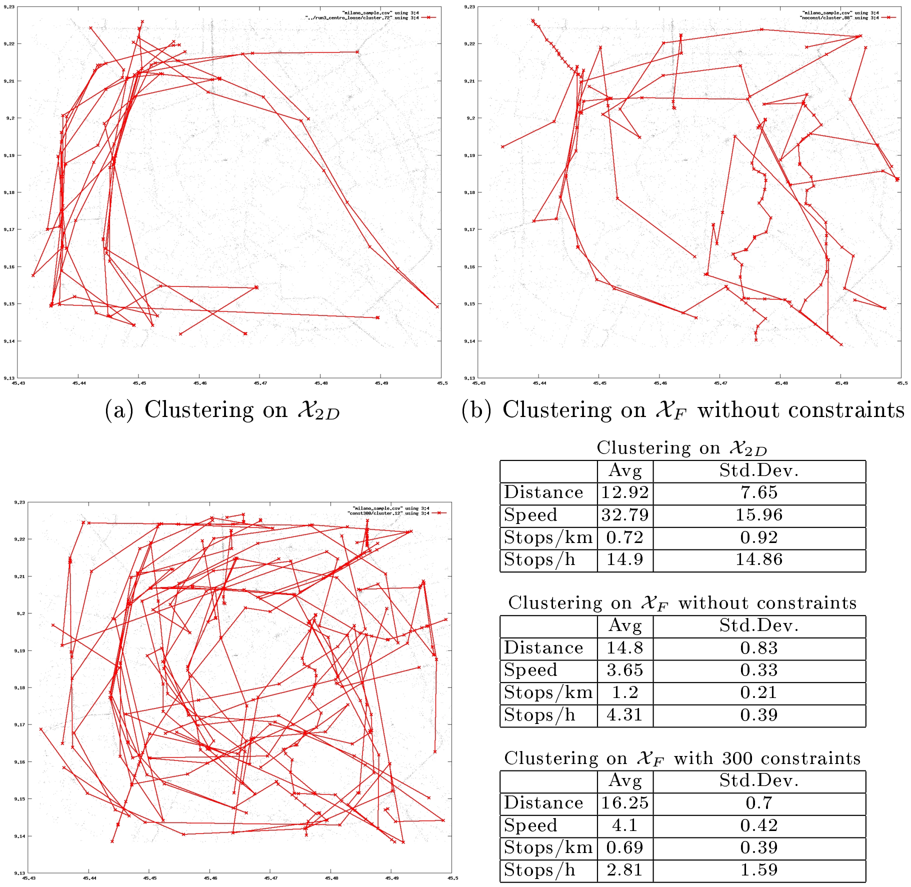
Table 1. Adjusted Rand Index for different clustering results.

feature domain and integrating some knowledge of the above mentioned clusters on trajectories, in the form of constraints. These three *variations* where found by selecting a cluster in (3) and its most similar counterparts in (2) and in (1), similarity being computed as Jaccard's similarity between the corresponding sets of trajectories.

Comparing Figures 2(a) and 2(b), we can see that the clusters found on the two universes, separately, are rather different, since contain different trajectories and also cover different regions of space. However, the cluster obtained by joining information from both the spaces (Figure 2(c)) results in a hybrid group, containing some of the trajectories of Figure 2(a) and most of those in Figure 2(b), thus conciliating, to some extent, the structure of the two clusters found on different spaces. The latter result was obtained by sampling 300 constraints from the clusters found on the trajectory space. Finally, in Figure 2(d) we report a summary of the three clusters shown. As we can see, the average speed in the cluster found on the trajectory space is much higher than the corresponding value in the features-only-generated cluster, and so is the number of stops per hour. Moreover, dispersion is very high, as compared to the results obtained on the feature space. The introduction of information (constraints) about the trajectory space changes the average statistics of the cluster significantly — apparently, by enforcing a selective inclusion of faster and longer trajectories, that stop much less than their counterpart in the features-only-generated cluster. Moreover, excepting the case of total distance, the dispersion in the cluster over features with constraints increases significantly, due to the feature heterogeneity of the newly introduced trajectories, though keeping on far lower values than the cluster on the trajectory space.

4 Conclusions and open problems

Clustering in Parallel Universes is a recent field of research. Discovering groups of similar data that coexist in different descriptor spaces is challenging since usually, different spaces give rise to very different clustering results. Moreover, each space has its own metrics and often some clustering algorithms are preferred for some representations of the data. In this paper we introduced a framework that enables to combine different spaces possibly using different clustering algorithms. We presented an instance of the algorithm on a real data set of trajectories. We showed that using a pairwise clustering constraints is a promising way to tackle this problem. However a several problems have to be solved. First of all, constraint processing has not been integrating in all clustering algorithms, yet. A number of constraint-based solutions have been proposed in



(a) Clustering on $\mathcal{X}_{2D}$

(b) Clustering on $\mathcal{X}_F$ without constraints

(c) Clustering on $\mathcal{X}_F$ with 300 constraints

Clustering on $\mathcal{X}_{2D}$

	Avg	Std.Dev.
Distance	12.92	7.65
Speed	32.79	15.96
Stops/km	0.72	0.92
Stops/h	14.9	14.86

Clustering on $\mathcal{X}_F$ without constraints

	Avg	Std.Dev.
Distance	14.8	0.83
Speed	3.65	0.33
Stops/km	1.2	0.21
Stops/h	4.31	0.39

Clustering on $\mathcal{X}_F$ with 300 constraints

	Avg	Std.Dev.
Distance	16.25	0.7
Speed	4.1	0.42
Stops/km	0.69	0.39
Stops/h	2.81	1.59

(d) Statistics of the clusters

Fig. 2. Sample cluster found through clustering on space, on features and on both

the case of partitional approaches for standard euclidean metrics [17, 16, 5, 12, 2]. Also hierarchical clustering under constraints have been investigated [9], as well as density-based approaches [15, 6]. Concerning sequence similarities, several work exist that propose to learn an edit distance [3, 14]. Future work will mainly focus on the integration of constraints in existing clustering algorithms for trajectories (see, e.g., [13, 8, 11]). Finally, an additional, interesting perspective is to investigate the possibility of discoverying local patterns (such as frequent itemsets, bi-clusters, sequential patterns, frequent subgraphs) that coexist in Parallel Universes.

Acknowledgments This research is partially funded by the EU contract GeoPKDD IST-FP6-014915.

References

1. Osman Abul, Francesco Bonchi, and Mirco Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, pages 376–385, 2008.
2. S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings ACM SIGKDD 2004*, pages 59–68, Seattle, USA, 2004.
3. Marc Bernard, Amaury Habrard, and Marc Sebban. Learning stochastic tree edit distance. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *ECML*, volume 4212 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2006.
4. Steffen Bickel and Tobias Scheffer. Multi-view clustering. In *ICDM*, pages 19–26. IEEE Computer Society, 2004.
5. Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In Carla E. Brodley, editor, *ICML*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.
6. Christian Böhm and Claudia Plant. Hissclu: a hierarchical density-based method for semi-supervised clustering. In *EDBT '08: Proceedings of the 11th international conference on Extending database technology*, pages 440–451, New York, NY, USA, 2008. ACM.
7. Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502, New York, NY, USA, 2005. ACM.
8. D. Chudova, S. Gaffney, E. Mjolsness, and P. Smyth. Translation-invariant mixture models for curve clustering. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pages 79–88. ACM, 2003.
9. I. Davidson and S. S. Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *Proceedings PKDD 2005*, volume 3721 of *LNCS*, pages 59–70, Porto, Portugal, 2005. Springer.
10. L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.
11. P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. In *Proceedings of 9th International Symposium on Spatial and Temporal Databases (SSTD'05)*, pages 364–381. Springer, 2005.

12. Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In Claude Sammut and Achim G. Hoffmann, editors, *ICML*, pages 307–314. Morgan Kaufmann, 2002.
13. M. Nanni and D. Pedreschi. Time-focused density-based clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289, 2006.
14. Michel Neuhaus and Horst Bunke. Automatic learning of cost functions for graph edit distance. *Inf. Sci.*, 177(1):239–247, 2007.
15. Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas Ruiz. C-dbscan: Density-based clustering with constraints. In *11th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, pages 216–223, 2007.
16. K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings ICML 2001*, pages 577–584, Williamstown, USA, 2001.
17. Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *Proceedings ICML 2000*, pages 1103–1110, Standford, USA, 2000.
18. Bernd Wiswedel and Michael R. Berthold. Fuzzy clustering in parallel universes. *Int. J. Approx. Reasoning*, 45(3):439–454, 2007.