Data Min Knowl Disc (2010) 21:345–370
DOI 10.1007/s10618-009-0157-y

Density-based semi-supervised clustering

Carlos Ruiz · Myra Spiliopoulou · Ernestina Menasalvas

Received: 2 February 2008 / Accepted: 31 October 2009 / Published online: 21 November 2009

Abstract Semi-supervised clustering methods guide the data partitioning and grouping process by exploiting background knowledge, among else in the form of *constraints*. In this study, we propose a semi-supervised density-based clustering method. Density-based algorithms are traditionally used in applications, where the anticipated groups are expected to assume non-spherical shapes and/or differ in cardinality or density. Many such applications, among else those on GIS, lend themselves to constraint-based clustering, because there is a priori knowledge on the group membership of some records. In fact, constraints might be the only way to prevent the formation of clusters that do not conform to the applications' semantics. For example, geographical objects, e.g. houses, separated by a borderline or a river may not be assigned to the same cluster, independently of their physical proximity. We first provide an overview of constraint-based clustering for different families of clustering algorithms. Then, we concentrate on the density-based algorithms' family and select the algorithm DBSCAN, which we enhance with Must-Link and Cannot-Link constraints. Our enhancement is seamless: we allow DBSCAN to build temporary clusters, which

Responsible editor: Eamonn Keogh.

Part of Ernestina Menasalvas work was funded by the Spanish ministry under project grant TIN2008-05924.

C. Ruiz · E. Menasalvas
Facultad de Informática, Universidad Politecnica de Madrid, Madrid, Spain
e-mail: cruiz@cettico.fi.upm.es

E. Menasalvas
e-mail: emenasalvas@fi.upm.es

M. Spiliopoulou (✉)
Faculty of Computer Science, Otto-von-Guericke-University Magdeburg, Magdeburg, Germany
e-mail: myra@iti.cs.uni-magdeburg.de

we then split or merge according to the constraints. Our experiments on synthetic and real datasets show that our approach improves the performance of the algorithm.

Keywords Constraint-based clustering · Semi-supervised clustering · Density-based clustering · Instance level constraints · Constraints

1 Introduction

Clustering with instance-level constraints has received a lot of attention in recent years. It is a subcategory of semi-supervised clustering, which allows the human expert to incorporate domain knowledge into the data mining process and thus guide it to better results (Anand et al. 2004; Kopanas et al. 2002). The use of instance-level constraints to this task is motivated by a specific form of background knowledge that can be found in many applications: while the majority of the records/instances to be clustered are unlabeled, there is a priori knowledge for some instances, such as their label or their relationship to other instances. The so-called "Must-Link" and "Cannot-Link" constraints capture relationships among data instances, e.g. that two houses located at different sides of a border should not be assigned to the same village/cluster, even if they are physically close to each other. As another example, Wagstaff et al. (2001) use constraints among vehicle instances to identify the road lanes (clusters) from GPS data.

Beyond contributing to better quality of the clustering results, constraints have also the potential to enhance computation performance Davidson and Ravi (2005) by speeding up convergence. Furthermore, as reported in Bennett et al. (2000), constraints can also be used to prevent the formation of empty clusters.

Semi-supervised clustering with constraints can be designed for any type of clustering algorithm. Most studies have concentrated on partitioning algorithms like K-means, but there are also works on hierarchical constraint-based clustering (Davidson and Ravi 2005). In this study, we propose the use of instance-level constraints in *density-based clustering* and present C-DBSCAN, a constraint-based variation of the clustering algorithm DBSCAN proposed by Ester et al. (1996) for the grouping of noisy data in (possibly not-spherical) clusters.

Density-based clustering lends itself to constraint enforcement: differently from partitioning algorithms, which strive a globally optimal partitioning of the data space, density-based algorithms like DBSCAN build solutions that are only locally optimal. Hence, they can exploit both Must-Link and Cannot-Link constraints between proximal instances. This is an inherent advantage towards constraint-based partitioning algorithms, which may fail to converge in the presence of Cannot-Link constraints (Davidson and Ravi 2005).

A further advantage of density-based clustering with constraints lays in the nature of the applications. Algorithms like DBSCAN are useful, among others, for geographic applications, where knowledge of some geographical formations may be a priori available. In Figs. 1 and 2, we depict clusters on real and artificial data that exhibit remarkable constructs: they contain "bridges", locations of higher/lower density, diffuse borders or specific shapes. Such constructs are not unusual in geographical

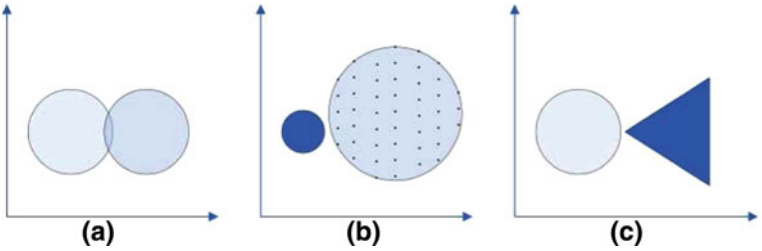

Fig. 1 Clusters that overlap (**a**), have different densities (**b**), have a bridge (**c**)

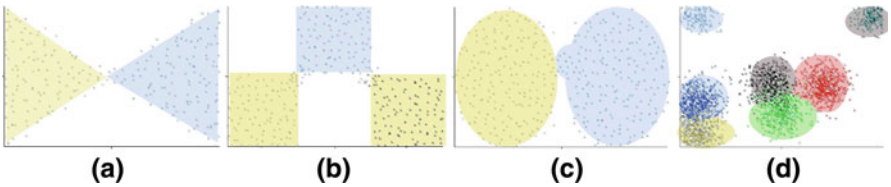

Fig. 2 Datasets where DBSCAN performs poorly: (**a**) DS1, (**b**) DS2, (**c**) DS3, (**d**) DS4

applications. These figures depict datasets for which DBSCAN is known to perform poorly. Hence, it is reasonable to attempt improvement through constraint exploitation. As we will show, our constraint-based variation of DBSCAN, C-DBSCAN, achieves indeed remarkable improvements for the above datasets. Earlier versions of our work have appeared in Ruiz et al. (2007a,b).

The paper is organized as follows: In Sect. 2, we provide an overview of constraint-based clustering methods for different families of algorithms. We elaborate on the different ways of incorporating the constraint enforcement procedure to an algorithm, on the different types of constraints being used and on methods for building the set of constraints to be enforced. Section 3 starts with a brief introduction to density-based clustering and a concise description of DBSCAN. We then present our constraint-based algorithm C-DBSCAN. In Sect. 4, we describe the framework of our experimental evaluation, including the evaluation criteria and the synthetic and real datasets we have used. Section 5 contains our experiments for different datasets and different sets of constraints, juxtaposing among else the performance of C-DBSCAN when only Must-Link or only Cannot-Link constraints are available. The last section concludes our study with a summary of our findings and a discussion on open issues.

2 Clustering with constraints

In Han et al. (1999) point out the need to organize data mining processes in an effective way, separating between computer labour and activities to be performed by human users. They argue that computers should undertake tasks like databases search, counting and algorithm execution, while users should specify the focus of data mining and guide the process. Constraint-based mining is an appropriate method for guiding a mining algorithm through the solution space.

This section contains a concise overview of constraint-based clustering. First, we discuss constraint types, including those proposed in the early work of Han et al. (1999). Next, we group the constraint-based clustering algorithms depending on how they enforce constraints (Sect. 2.2). Then we elaborate on methods that build an optimal set of constraints for the mining problem (Sect. 2.3).

2.1 Different types of constraints

Han et al. (1999) distinguish among constraints on the knowledge to be extracted from the data, on the selection of data that are relevant to the mining task (*data constraints*), on the feature space over these data and on the interestingness of the mining results. Data constraints, also termed *instance-level constraints*, have been the subject of much attention and research in recent years.

In the specific domain of spatial databases, *physical constraints* appear by nature, so there are many methods that model and exploit them (Zaïane and Lee 2002a,b). In particular, constraints are used to capture formations like rivers, woods or villages that can be described with help of two-dimensional polygons. Such methods assume a grid over the data space and look for geometrical shapes in it.

In Ruiz et al. (2006), we have proposed different types of constraints for constraint-based clustering over stream data, distinguishing among *constraints on pattern* (i.e. on the whole of a cluster), *constraints on attribute* (i.e. on the feature space) and *co-appearance* and *separation* constraints (i.e. constraints on sets of data instances). Hereafter, we concentrate on instance-level constraints for static data.

Another categorization of constraints can be made on the basis of the "object(s)" referenced by the constraints: *instance-level constraints* refer to individual data records. Traditionally, one distinguishes between *Must-Link constraints* on instances that must appear together and *Cannot-Link constraints* on instances that are not allowed to be in the same cluster (Wagstaff et al. 2001). The τ_n-constraint on cluster cardinality Bennett et al. (2000) refers to a whole cluster instead. The ϵ-constraint on the distance between points inside a cluster states that any two points must be closer than a distance threshold ϵ, while the δ-constraints on the distance between clusters requires that points in different clusters must be at a distance of at least δ (Davidson and Ravi 2005); both constraint types refer to whole clusters.

Clustering with instance-level constraints has recently received a lot of attention (Davidson and Basu 2005, 2006), also under the label *semi-supervised clustering* (Gunopulos et al. 2006). The core idea is that background knowledge on cluster membership may exist for a small number of data records. This knowledge can be effectively exploited to guide the clustering algorithm: while a traditionally clustering algorithm treats all records equally, a constraint points out that two given instances *must* be assigned to the same cluster or, conversely, they *must* be assigned to different clusters.

Example 1 Consider the queries Q_1, Q_2, Q_3, where Q_1 contains the keyword "Basel", Q_2 is on "risk management in banks" and Q_3 mentions "quality control" and "process". Although risk management and the city Basel in Switzerland are not related in general, one might know (or figure out by studying the query results) that

both users were interested in the Basel regulations on bank services, especially with respect to risk management.

Basel regulations include among else quality control and certification, so that Q_3 seems also related to Q_1. However, inspection may reveal that those who launched Q_3 were interested in quality control for manufacturing processes. Thus we conclude that Q_1 and Q_2 must belong to the same cluster, while Q_3 must belong to a different cluster than Q_1. This results in a Must-Link constraint on Q_1, Q_2 and in a Cannot-Link constraint on Q_1, Q_3.

The identification of the constraints is achieved by manual inspection. This cannot be afforded for all data. However, research on constraint-based clustering Wagstaff et al. (2001), Davidson and Ravi (2005), and Davidson and Basu (2006) shows that the identification and exploitation of a small number of constraints is adequate to enhance the clustering of the whole dataset.

2.2 Different ways of instance-level constraint enforcement

Clustering with enforcement of instance-level constraints encompasses methods that adhere to different principles. We first discuss methods that *embed* constraints into the optimization criterion of the clustering algorithm, distinguishing among algorithms that optimize a global objective function (Wagstaff et al. 2001; Basu et al. 2002; Davidson and Ravi 2005) and those that optimize locally (Demiriz et al. 1999; Wagstaff and Cardie 2000; Davidson and Ravi 2005).

We then turn to algorithms that use the constraints to "learn" a new distance metric for the clustering algorithm (Basu et al. 2004b; Bilenko et al. 2004; Halkidi et al. 2005). Under this approach, which is often termed *distance-based*, clustering is done in a conventional way, using the new distance function. Finally, we discuss algorithms that combine both approaches. The incorporation of constraints into the optimization criterion and learning of the distance function.

Our density-based algorithm C-DBSCAN Ruiz et al. (2007a,b) belongs to the first category above. It embeds instance-level constraints into the local optimization criterion of the density-based clustering algorithm DBSCAN.

2.2.1 Embedding constraints into the optimization criterion

Methods of this category enforce constraints by modifying the optimization criterion of the clustering algorithm. Wagstaff et al. have embedded constraints into the incremental COBWEB algorithm (Wagstaff and Cardie 2000). The new algorithm COP-COBWEB returns a clustering that satisfies all constraints; if no such clustering can be built, no solution is returned. Similarly, COP-K-means extends the partitioning algorithm K-means into a variation that enforces all constraints; if it is not possible to satisfy all constraints, no clustering is built (Wagstaff et al. 2001). The experiments of the authors on UCI datasets (Newman et al. 1998) show that the new algorithms are superior to their conventional variants in terms of both computational time and convergence speed.

Basu et al. (2002) investigate how constraints can be used to set the initial seeds for K-means. Instead of selecting the seeds randomly, they build the transitive closure of the instance-level constraints, compute clusters upon it and then use the centers of these clusters as seeds. They derive two methods, one that ignores the constraints after the initialization step ("Seeded-K-Means") and one that enforces them through all subsequent steps until convergence. Since datasets are prone to noise, Seeded-K-means is more appropriate for noisy datasets where the constraints may be less trustworthy.

Constraint-based variations of K-means are algorithms that try to build a globally optimal solution. This is not always convenient, though, because there is no guarantee that the constraint-based algorithm will converge at all. In Davidson and Ravi (2005), show that the satisfaction of all Must-Link and Cannot-Link constraints by K-means is an NP-complete problem, and that Cannot-Link constraints may prevent the algorithm from converging.

The convergence problem is trivially avoided by embedding constraints to algorithms that operate locally instead of building globally optimal solutions. Davidson and Ravi (2005) propose constraint enforcement with a hierarchical clustering algorithm and show that constraint satisfaction becomes a P-complete problem. Our C-DBSCAN is based on the same observation: C-DBSCAN operates locally upon the data and, as we show, builds clusters that satisfy all constraints. It also exhibits the inherent advantage of the base algorithm DBSCAN in being robust towards clusters of arbitrary shapes (Ester et al. 1996).

An early algorithm that embeds constraints indirectly into the optimization criterion is worth mentioning here. Demiriz et al. (1999) embed the constraints into the selection and cross-over of a genetic algorithm: they combine an unsupervised learner which minimizes within-cluster variance with a supervised technique that uses the records with known class label to minimize cluster impurity.

2.2.2 Embedding constraints into the distance function

Some clustering algorithms use the constraints to *learn* a new, adjusted distance/similarity function. These algorithms are often called *semi-supervised learning* algorithms.

In the new function, instances associated with a Must-Link constraint are "pushed closer" to each other while instances associated with a Cannot-Link constraint are "pulled away" from each other. The function is typically implemented in a look-up conversion matrix, thus allowing for individual distance values for any pair of records. The use of an adjusted function for constraint enforcement implies that constraints may be violated, e.g. by separating instances that should be linked, because they are still too far away from each other. Constraint violation may also be associated with some penalty value.

Different distance functions can be used as basis for this group of algorithms. Xing et al. (2003) model the learning of the distance function as a convex optimization problem; they use the Mahalanobis distance as basis. Klein et al. (2002) use the Euclidean distance as a basis and learn a distance function that computes the shortest path between points. It allows them to further generalize instance-level constraints

into a global relationship among points in a Euclidean space. They show that this generalization is superior to earlier approaches.

The asymmetric Kullback–Leibler divergence is used by Cohn et al. (2003). Their method is designed for document clustering with user feedback, so the learned distance function reflects the different perspectives of users upon documents.

In Bilenko and Mooney (2003), use semi-supervised learning to detect duplicate records in a database. For this task, they model the records as vectors of tokens, but recognize that some tokens are better indicators of similarity than others. To capture this into the model,they learn a new edit distance function that assigns more weights to those tokens. They incorporate this learnable edit distance into an Expectation-Maximization clustering algorithm. They have also developed a novel similarity measure that is learned with an SVM. Their results show that the semi-supervised variation is superior to conventional EM, while the semi-supervised SVM exhibits irregular performance (Bilenko and Mooney 2003).

In a later work, Bilenko et al. (2004) propose MPCK-means: this method incorporates the learning of the distance function on the labeled data and on the data affected by the constraints *in each iteration* of the clustering algorithm. Hence, this algorithm learns a different distance function for each cluster.

2.2.3 Hybrid methods

Algorithms that embed constraints to the distance function can penalize constraint violation but do not prohibit it per se. On the other hand, algorithms that embed constraints in the objective function may deliver no solution at all. Hybrid algorithms attempt to avoid both pitfalls.

Basu et al. (2004b) propose semi-supervised probabilistic clustering with Hidden Markov random fields (HMRFs). The authors propose a framework for the learning of the distance function, which allows the use of different metrics like cosine, Bregman divergence, etc. The learned function is used in a variation of K-means for clustering. The new algorithm, HMRF-K-means, minimizes an objective function that is derived from the joint probability of the model together with the penalty values for constraint violations. Bilenko et al. (2004) minimize the objective function of MPCK-means that uses learned distance functions.

A further method of the same category appears in Halkidi et al. (2005): constraint violation penalties are incorporated to the distance metric and then combined with a modified objective function. The authors use a hill-climbing algorithm to find an optimal solution. Optimality includes satisfying a maximal number of constraints. The objective function uses the S_Dbw measure proposed in Vazirgiannis et al. (2003); this measure takes into account both the cohesion and the separation of the clusters.

2.2.4 Constraints on streams

All of the above methods enforce constraints in a static dataset. In recent years, there has been much research on clustering data that are not static but rather arrive as a stream. Stream clustering algorithms lend themselves quite naturally to constraint enforcement, because knowledge on already seen data can be exploited to adapt the clusters

on data that arrive later on. In Ruiz et al. (2009) we propose C-DENSTREAM, a density-based stream clustering algorithm that exploits so-called "micro-constraints". These are instance-level constraints that involve instances seen at different time points.

2.2.5 Closing remarks

A major challenge for constraint-based clustering is the interplay between achieving a feasible solution (i.e. ensuring convergence), and satisfying all constraints. The convergence issue concerns only algorithms like K-means, which build a globally optimal solution. For K-means, Davidson and Ravi (2005) have shown that the satisfaction of both Must-Link and Cannot-Link constraints is an NP-complete problem and that it is the presence of Cannot-Link constraints which may prevent convergence. Clustering algorithms that embed the constraints into the distance function omit the convergence problem but allow for solutions that may violate some constraints. The same holds for algorithms that embed the constraints to both the objective function and the distance function.

Clustering algorithms that build local solutions do not face the convergence problem (by nature) and are thus very appropriate for constraint enforcement. We follow this line of thinking here by proposing a constraint-based variation of the density-based clustering algorithm DBSCAN. As we show in the next sections, C-DBSCAN satisfies all input constraints while demonstrating performance improvements over DBSCAN, especially in the cases where DBSCAN is known to perform poorly.

2.3 Determining the set of constraints

Research performed in the last years on the exploitation of domain information in the form of instance level constraints has shown that improvements in results highly depend on the selected set of constraints (Wagstaff 2002; Davidson et al. 2006). Therefore, most constraint-based methods calculate the average performance achievements of the results using random sets of constraints (Wagstaff and Cardie 2000; Wagstaff et al. 2001; Halkidi et al. 2005; Davidson and Ravi 2005; Ruiz et al. 2007a).

Instead of using a randomly-generated set of constraints, some methods build the constraints set in interaction with the user (Cohn et al. 2003; Davidson et al. 2007). Cohn et al. (2003) describe an interactive method that derives relationships among documents and document-group memberships and then returns them to the user for feedback. With help of user feedback, the method can then adjust the constraints and achieve a better, user-tailored clustering. The authors show that the human interaction leads to the discovery of more intuitive relationships. Obviously, the method has the disadvantage of requiring cluster recomputation any time the set of constraints is adjusted. Davidson et al. (2007) incorporate user feedback for constraint adjustment in an *incremental* clustering method, thus reducing the overhead of re-clustering.

In Davidson et al. (2006), propose two quantitative measures that calculate the expected benefit of each constraint on the clustering results. The *information measure* computes the additional amount of information that the algorithm obtains through the set of constraints, i.e. the amount of information that the algorithm would not be able

to acquire without the constraints. The *coherence measure* computes the number of times the constraints agree with (i.e. they *are compliant with*) the objective function. The authors show that the higher the coherence and amount of information contributed by the set of constraints is, the higher is also the performance improvement.

3 Semi-supervised clustering with C-DBSCAN

We perform semi-supervised clustering with constraint enforcement on the basis of a density-based algorithm. We first provide a short overview of density-based clustering methods and then describe our own algorithm C-DBSCAN.

3.1 A retrospective to density-based clustering

A number of successful density-based clustering algorithms have been proposed in the end of the previous decade, including DBSCAN (Ester et al. 1996), DENCLUE (Hinneburg and Keim 1998) and OPTICS (Ankerst et al. 1999). DBSCAN has been designed for the clustering of large noisy datasets with some emphasis on spatial data (Ester et al. 1996). DBSCAN introduced the concept of "neighbourhood" as a region of given radius (i.e. a sphere) and containing a minimum number of data points. Connected neighbourhoods form clusters, thus departing from the notion of spherical cluster. DENCLUE (Hinneburg and Keim 1998) is also based on neighbourhoods, concentrating on high-dimensional multimedia databases. Within the multidimensional data space, DENCLUE computes the *impact* of a data point upon its neighbourhood and uses it to assign data points to clusters. OPTICS (Ankerst et al. 1999) is not a clustering algorithm in the strict sense; it rather contributes in identifying the clustering structure by ordering points and reachability distances in a way that can be exploited by a density-based algorithm. Like DBSCAN and DENCLUE, it observes density as a regional/local phenomenon.

Dense neighbourhoods have been considered in STING (Wang et al. 1997), WAVECLUSTER (Sheikholeslami et al. 1998), CLIQUE (Agrawal et al. 1998) and DESCRY (Angiulli et al. 2004). STING (Wang et al. 1997) uses a hierarchical statistical information grid to reduce the computational cost of data querying. First, it recursively divides the data space into cells that contain statistics over sets of objects, such as cardinality, mean, deviation and other information on the distribution. Then, each cell points to a set of smaller cells at the next lowest level of the tree. WAVECLUSTER (Sheikholeslami et al. 1998) applies a wavelet transform to discover relative distances between objects at different levels of resolution. It divides the space using a grid structure and creates an n-dimensional feature space where wavelet transform is performed multiple times. CLIQUE (Agrawal et al. 1998) combines a density-based and a grid-based approach to find clusters embedded in subspaces of a high-dimensional data space: it partitions the data space into rectangular units that are considered dense if they contain a minimum number of points. Connected dense units form clusters.

The more recently proposed DESCRY (Angiulli et al. 2004) returns back to the idea of a neighbourhood as a spherical region, as it was used in DBSCAN (Ester et al. 1996). DESCRY builds clusters in four steps: First, the data are sampled, then they

are partitioned using a KD-Tree (Bentley 1975), whereupon the centers of the leaf nodes (the so-called "meta-points") are computed. Clustering is performed as a last step, using a hierarchical agglomerative algorithm. Similarly to DESCRY, we also use a KD-Tree to build the initial regions, but thereafter we use DBSCAN and a set of constraints to connect regions into clusters.

3.2 Introducing C-DBSCAN

The original DBSCAN discovers and connects dense neighbourhoods of data points (Ester et al. 1996). In particular, it identifies *core points*, i.e. those having at least $MinPts$ data points as neighbours within a radius Eps. It then connects overlapping neighbourhoods into clusters. Data points within the same neighbourhood are termed *density-reachable* from the core point, those in overlapping neighbourhoods are *density-connected* to each other.

Our algorithm C-DBSCAN (cf. Algorithm 1) extends DBSCAN in three steps. We first partition the data space into denser subspaces with help of a KD-Tree (Bentley 1975). From them we build a set of initial *local clusters*; these are groups of points within the leaf nodes of the KD-tree, split so finely that they already satisfy those Cannot-Link constraints that are associated with their contents. Then, we merge density-connected local clusters and enforce the Must-Link constraints. Finally, we merge adjacent neighbourhoods in a bottom-up fashion and enforce the remaining Cannot-Link constraints.

In the next subsection, we explain how C-DBSCAN partitions the data space. In Sect. 3.4, we discuss how the algorithm enforces constraints during cluster construction. Both subsections refer to the pseudo-code of Algorithm 1. We use as an example the data depicted in Fig. 3a, together with some constraints on data instances, as shown in Fig. 3b.

3.3 Partitioning the data space

For the partitioning step, we use the KD-Tree proposed in Bentley (1975). The motivation is to deal effectively with subareas that have different densities.

The KD-Tree construction algorithm divides the data space iteratively into cubic structures by splitting planes that are perpendicular to the axes. Each cube becomes a node and is further partitioned as long as each new node contains a minimum number of data points, specified as input parameter. The result is an unbalanced tree: small leaf nodes capture locally dense subareas while large leaf nodes cover the less dense subareas. In C-DBSCAN, we set the threshold value $MinPts$ to the same value as this parameter, sine the goals of the two are very similar—to define and select dense areas.

Since there are many possible ways to choose axis-aligned splitting planes, there are many different ways to construct KD-trees. In C-DBSCAN, we perform a depth-first tree traversal; at each level, we split first across the X-axis as long as the new nodes have a minimum number of points, choosing the median as cut-off for the plane perpendicular to the axis.

Algorithm 1: C-DBSCAN

Data:
A set of instances, D.
A set of Must-Link constraints, $ML = \{ML_0, \ldots, ML_n\}$.
A set of Cannot-Link constraints, $CL = \{CL_0, \ldots, CL_m\}$.
Result: A set of clusters over D satisfying ML and CL.
begin

Step 1 → Partition the data space with a KD-Tree
$kdtree$:= BuildKDTree(D).

Step 2 → Create local clusters in the KDTree
for *(each leaf node $v \in kdtree$ and each unlabeled point $p_i \in v$)* **do**
  $DR(p_i)$:= all points density-reachable from p_i within Eps.
  if *($|DR(p_i)| < MinPts$)* **then**
    Label p_i as `NOISE_POINT`.
  else if *(exists a constraint in CL among points in $DR(p_i)$)* **then**
    Each point in $DR(p_i)$ and p_i becomes one `LOCAL_CLUSTER`.
  else
    Label p_i as `CORE_POINT`.
    All of $\{p_i\} \cup DR(p_i)$ becomes one `LOCAL_CLUSTER`.
  end
end

Step 3a → Merge clusters and enforce the Must-Link constraints
for *(each constraint in ML)* **do**
  Let p, p' be the data points involved in the constraint.
  Find the clusters C, C' with $p \in C$ and $p' \in C'$.
  Merge C, C' into cluster $Cnew := C \cup C'$ and label it as `ALPHA_CLUSTER`.
end

Step 3b → Build the final clusters
while *(number of local clusters decreases)* **do**
  for *(each* `LOCAL_CLUSTER` *C)* **do**
    Let C' be the closest `ALPHA_CLUSTER` that is density-reachable from C.
    if *(exists no constraint in CL between points of C, C')* **then**
      Incorporate C into C', i.e. $C' := C \cup C'$.
    end
  end
end
return each `ALPHA_CLUSTER` and each remaining `LOCAL_CLUSTER`.
end

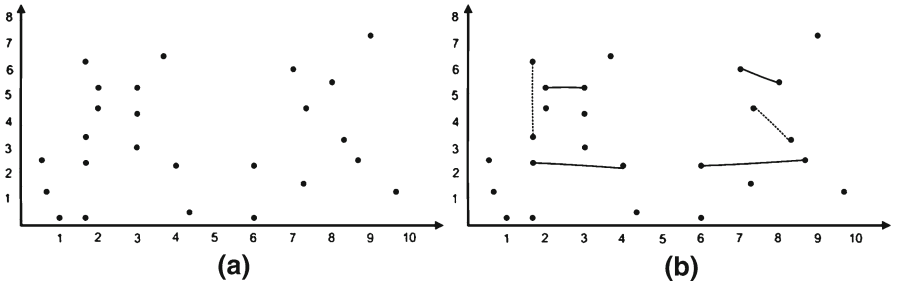

Fig. 3 C-DBSCAN: (**a**) Example dataset and (**b**) constraints on it—Must-Link constraints as *solid lines* and Cannot-Link constraints as *dotted lines*

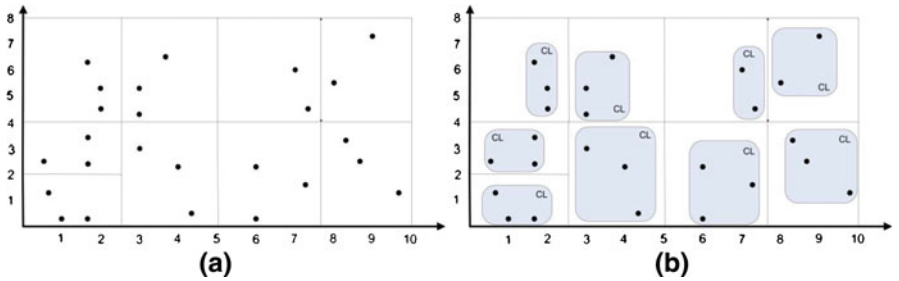

Fig. 4 C-DBSCAN: (**a**) KD-Tree leaf nodes built at Step 1 and (**b**) local clusters built at Step 2

In Fig. 4a we show the leaf nodes of the KD-Tree for the example dataset in Fig. 3, where the minimum number of points has been set to two. This corresponds to Step 1 of C-DBSCAN, as explained in the next subsection.

With this partitioning, it is possible to break thin bridges between dense subareas in the data: instead of connecting arbitrary adjacent neighbourhoods to build clusters, only neighbourhoods within the same node are considered at first. The merging of neighbourhoods belonging to different nodes is driven by the constraints, as explained in the description of Algorithm 1 below.

3.4 Applying instance-level constraints

We consider Must-Link and Cannot-Link constraints as in Wagstaff et al. (2001). A Must-Link constraint for a pair of data points p_i and p_j specifies that they must be assigned to the same cluster. A Cannot-Link constraint for p_i and p_j states they must be assigned to different clusters.

Must-Link constraints have the *transitivity property* (Wagstaff et al. 2001): if X and Y are two sets of such constraints and if $X \cap Y \neq \emptyset$, then $MustLink(X) \wedge MustLink(Y) \rightarrow MustLink(X \cup Y)$.

Cannot-Link constraints have the *entailment property* (Wagstaff et al. 2001): let X and Y be sets of Must-Link constraints as before and let p_i and p_j be data points, such that p_i appears in a constraint in X, p_j appears in a constraint in Y and it holds that $CannotLink(p_i, p_j)$. This implies that for each a appearing in a constraint of X and for each b appearing in a constraint of Y it holds that $CannotLink(a, b)$.

3.4.1 Creating local clusters under cannot-link constraints

As shown in Step 2 of Algorithm 1, the algorithm traverses each leaf node of the KD-Tree in turn. Inside the node, density-reachable points are grouped together, subject to Cannot-Link constraints. The result is a set of within-node clusters. We use the term *local clusters* for these preliminary clusters to stress that each of them consists of points that are within the same leaf node.

In particular, if there is a Cannot-Link constraint involving data points within the same leaf, then each data point of the leaf becomes a local cluster by itself, i.e. a single-

ton local cluster. If there are no Cannot-Link constraints to be satisfied, C-DBSCAN is invoked in the conventional way for each data point p_i in the leaf node: it is checked whether there is a neighbourhood with at least $MinPts$ points in a radius Eps around p_i. If the neighbourhood has too few points, then p_i is labeled as `NOISE_POINT`. Otherwise, all data points that are density-reachable from p_i constitute one local cluster. Hence, the output of Step 2 is a set of local clusters, some of which may contain only a single point.

Figure 4b depicts the local clusters built at the end of Step 2 for the dataset in Fig. 3.

3.4.2 Merging local clusters under must-link constraints

In Step 3a (cf. Algorithm 1) C-DBCSAN enforces the Must-Link constraints. It processes each Must-Link constraint in turn, identifies the data points involved in it and merges the clusters containing them. These merging operations can go beyond the borders of the KD-Tree leaf nodes, in the sense that a merge may involve local clusters of different KD-Tree leaf nodes.

We thereby take existing cannot-link constraints into account: if a cannot-link constraint is violated by a merge, we split again so that the constraint is satisfied.

We use the term *alpha_cluster* for the result of such a cluster merge. Obviously, the clusters being merged may be local clusters or alpha clusters built by a previous merge operation. They ensure that all Must-Link constraints are satisfied and form the basis of the final clustering done in Step 3b. In Fig. 5a we see the result of Step 3a for the example in Fig. 3.

3.4.3 Merging clusters under cannot-link constraints

In the last step of C-DBSCAN (cf. Algorithm 1), the remaining local clusters are merged into the alpha clusters built in Step 3a, while respecting the Cannot-Link constraints.

For cluster merging, we use hierarchical agglomerative clustering with single linkage. However, we only consider density-reachable data points when we compute distances. Moreover, the cluster merging process is driven by the local clusters, since our objective is to incorporate them into the alpha clusters built in Step 3a. Therefore, the iterative cluster merging process of the hierarchical clustering algorithm considers for

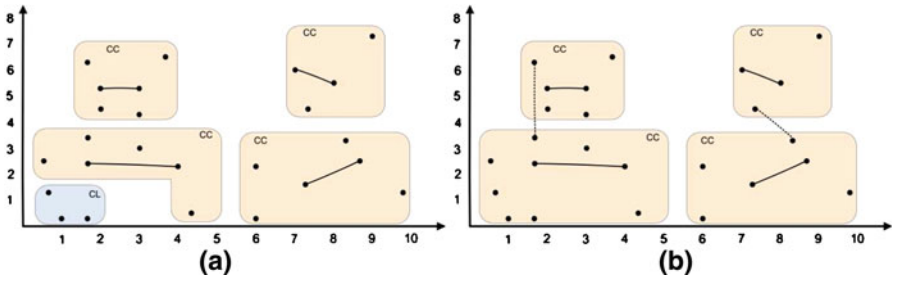

Fig. 5 C-DBSCAN: (**a**) Step 3a and (**b**) Step 3b of the algorithm on the example dataset of Fig. 3

each local cluster only the alpha clusters that are close to it as candidates to be merged with.

For each local cluster and candidate alpha cluster, C-DBSCAN checks whether they contain data points involved in a Cannot-Link constraint. If this is the case, the clusters are not merged, otherwise the local cluster becomes part of the alpha cluster. Hence, the algorithm stops when the number of local clusters does not change any more. In Fig. 5b we depict the final clusters for the example dataset in Fig. 3.

3.4.4 Discussion on constraint enforcement

Most constraint-based clustering algorithms make a best effort to enforce all constraints. C-DBSCAN rather ensures that *all* constraints are satisfied: Step 3a enforces all Must-Link constraints; Step 2 enforces Cannot-Link constraints within the same KD-tree node; Step 3b enforces the remaining Cannot-Link constraints by preventing the merging of clusters that would violate them. This is achieved at the cost of building many small clusters, since each Cannot-Link constraint results in a singleton local cluster. These small clusters may be absorbed by alpha clusters in Step 3b, as can be seen in the example of Fig. 5b.

C-DBSCAN may still be forced to retain many singleton clusters. For example, assume that the dataset contains n Cannot-Link constraints. In Step 2, they result in $m \leq 2n$ local clusters $C_1, \ldots, C_m$; the number m may be less than $2 \times n$ because a data point may be involved in more than one Cannot-Link constraints but becomes itself one local cluster. We consider the extreme case that all of the remaining data points belong to a single alpha cluster C_{large}, built after enforcing the Must-Link constraints in Step 3a and absorbing the remaining local clusters in Step 3b. Obviously, C-DBSCAN can incorporate only one of $C_1, \ldots, C_m$ into C_{large}. Hence, the final result will consist of one large cluster and $m - 1$ singleton clusters.

Different merging schemes might lead to a better result in such a case. First, some of the singleton local clusters could be merged to each other; unless there are Must-Link constraints upon them, though, this will not be done. Second, splitting the alpha cluster and merging its partitions to the singleton local clusters might result in a better clustering with respect to homogeneity. In our future work, we intend to enhance C-DBSCAN with heuristics that allow for more homogeneous clusterings while respecting the Cannot-Link constraints.

4 Evaluation framework

We have studied the impact of constraints on the clustering results by applying C-DBSCAN and the original DBSCAN on artificial and real datasets. Since there were no user defined constraints in these datasets, we have generated constraint-sets of different sizes. Constraint generation is explained in Sect. 4.3.

To compare C-DBSCAN to DBSCAN we have tuned DBSCAN to achieve its best possible performance. Ester et al. (1996) propose a heuristic method for tuning the DBSCAN parameters $MinPts$ and Eps on unlabeled data. However, since our intention is not to evaluate DBSCAN but rather use it as a reference, we exploit the class

labels for parameter tuning: we have performed several runs with different parameter values, compared the results with the true partitioning and chose the values that gave the best approximation of the classes. We have used the same parameters for C-DBSCAN. Thus, any performance difference between DBSCAN and C-DBSCAN can be attributed to the constraints.

In this section, we describe the evaluation method and the datasets. The experiments are presented in the next section.

4.1 Evaluation method

The datasets used in our experiments are labeled, so the correct data partitions are known a priori. Hence, we compute the quality of the clusterings generated by DBSCAN (no constraints) and C-DBSCAN (for various percentages of data points involved in constraints) towards the real classes on the basis of the Rand Index measure proposed in Rand (1971). The Rand Index measures the similarity of partitions, acquiring the highest value 1 when the clusters are exactly the same as the real classes.

More formally, let ζ_1 and ζ_2 be two clusterings over the same dataset D, not necessarily containing the same number of clusters. We count the number of "agreements" and "disagreements" between ζ_1 and ζ_2, distinguishing two types of agreement and two types of disagreement. In particular, we count as "agreements" the number a of data points that appear together in the same cluster for both ζ_1 and in ζ_2 and the number b of data points that appear in different clusters for both ζ_1 and in ζ_2. We count as "disagreements" the number c of data points that appear in the same cluster of ζ_1 and in different clusters of ζ_2, and the number d of data points that appear in the same cluster of ζ_2 and in different clusters of ζ_1. Then:

$$\text{Rand}(\zeta_1, \zeta_2) = \frac{\text{agreements}}{\text{agreements} + \text{disagreements}} = \frac{a+b}{a+b+c+d}$$

We also experimented with the Jaccard Coefficient but found similar results, so we only present the experiments with RandIndex hereafter.

4.2 Datasets for the evaluation

4.2.1 Synthetic and public domain datasets

We have used two groups of data sets for experimentation. The first group contains artificial datasets, for which DBSCAN is known to perform poorly, as explained in Sect. 1. The clusters in these datasets are depicted in Fig. 2. The second group consists of real datasets from the Machine Learning Repository UCI (Newman et al. 1998) or used in previous research work. They are depicted in Fig. 6 and summarized in Table 1.

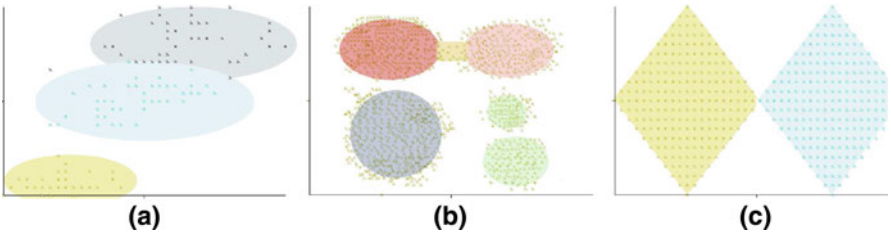

Fig. 6 Datasets: (**a**) IRIS, (**b**) CURE, (**c**) CHAMELEON

Table 1 Data sets used in the experiments

Data sets	Features	Parameters
Synthetic	DS1 (2 classes, bridge between clusters)	5–100% of the instances involved in constraints
	DS2 (3 classes -bridge between clusters, 2 overlapped)	
	DS3 (2 classes -bridge between clusters)	
	DS4 (7 gaussian clusters, bridge between clusters, 4 overlapped, 2 by 2, different notions of density)	
From UCI	Iris (Newman et al. 1998) (3 classes, 2 overlapped)	5–100% of the instances involved in constraints
	Cure (Guha et al. 1998) (5 classes, 4 overlapped, 2 by 2)	
	Chameleon (Karypis et al. 1999) (2 classes, bridge between clusters)	
Real-world applications	Subset of a query log	5–10% of the instances involved in constraints

4.2.2 A query log dataset

For our experiments on real data we have also used a query log dataset of initially 69,790 records. This log was recorded at the site of a real-estate agent that operates a local Google search appliance (GSA). Next to query requests, the log also contained information about further activities of the users on the search page, e.g. selecting among alternatives (locations, price, categories, etc) to restrict their query. During data preparation, the log was cleaned and the interaction of users with the search engine was reconstructed. After preprocessing, 45,450 "query sessions" were retained, each one composed of a query and the activities performed by the user on the query results.

The queries were further enhanced with help of an ontology. In particular, the ontology contained a taxonomy of concepts and a set of properties describing the objects of the site, including types of real-estate, prices, locations, etc. For each page in a query result, we used the ontology to eliminate stopwords from the page, to map the remaining keywords in it to concepts and to extend the objects depicted in the page with the property data.

After these enhancements, we assigned queries to categories/classes automatically. This was done as follows. Each query in the log was sent to the live search engine, the 10 first results were analyzed and their contents were juxtaposed to the ontology;

the query was then labeled with the best matching ontology concept. Finally, arbitrary pairs of instances were chosen to derive Must-Link constraints (for queries with the same label) and Cannot-Link constraints (for queries with different labels). This allowed us to compare the clusters of queries to a ground truth of query classes.

4.3 Generating constraints

Similarly to Wagstaff et al. (2001) and Halkidi et al. (2005), we use the class labels of the test datasets to generate constraints. In particular, we select pairs of data points randomly and check the label of each pair: if the two data points have the same label, we generate a Must-Link constraint for them, otherwise we generate a Cannot-Link constraint. To study the impact of the number of constraints on the performance of our algorithm, we vary the percentage of the data for which constraints are generated, from 5 to 100% (with a step of 5%). For each set of constraints thus generated we perform 10 runs with cross-validation and compute the average of the achieved Rand Index values.

5 Experimentation

In our experiments, we have first studied the impact of constraint enforcement on clustering quality—measured with the Rand Index. We have further studied the contribution of Must-Link constraints alone and of Cannot-Link constraints alone. Finally, we considered constraints that have not been selected randomly.

5.1 Experiments on synthetic data and public data

In the upper part of Fig. 7 we show the performance of C-DBSCAN for the four artificial datasets DS1, . . . , DS4. The horizontal axis depicts the percentage of data points involved in constraints. In the vertical axis, we see the Rand Index values towards the real partitions. The Rand Index value(s) of DBSCAN appear at point zero of the horizontal axis, since DBSCAN does not consider constraints. In the lower part of the same Figure we show the performance results for the datasets IRIS, CURE and CHAMELEON of the UCI Repository.

The curves show that C-DBSCAN alleviates the shortcomings of DBSCAN and achieves very good partitioning of the datasets, reaching Rand Index values of more than 0.8 for all datasets. It can also be seen that already a small amount of domain information is adequate for very high performance: After considering constraints on 10% of the data points, a saturation is already achieved.

Since the data points involved in the constraints are selected randomly, our results indicate that a small amount of arbitrary labeled data suffices to guide C-DBSCAN towards a good partitioning and that an increase in the number of labeled data is not necessary. These results verify the findings of Wagstaff and Cardie (2000) and Wagstaff et al. (2001) but reflect higher performance improvements over the original method.

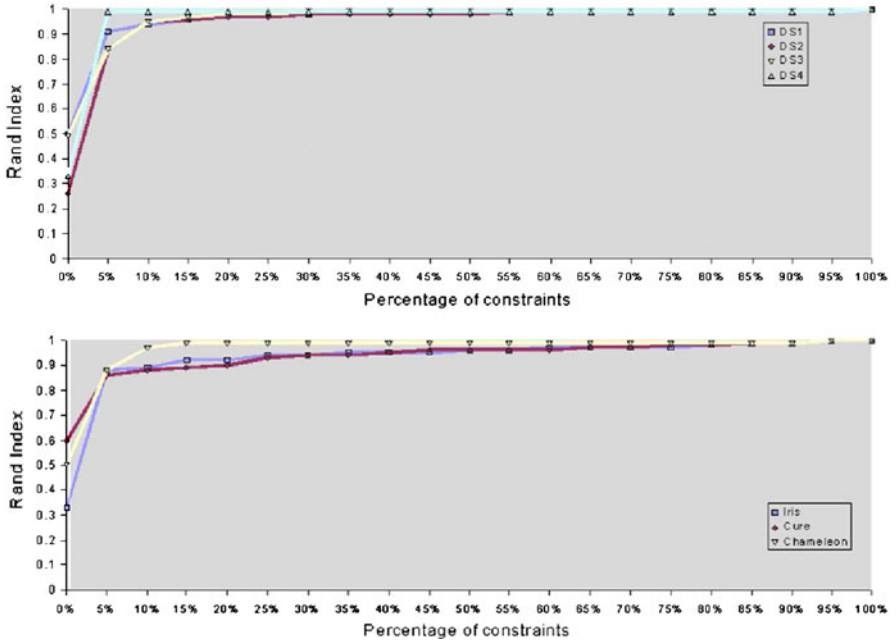

Fig. 7 C-DBSCAN performance for constraint-sets of different size: the Rand Index shows performance improvement over DBSCAN (see position at 0% constraints) and a fast saturation for both synthetic datasets (*upper part* of the figure) and UCI datasets (*lower part*)

5.1.1 Experiments using only must-link constraints

We next experimented with Must-Link constraints only. Our results are in Fig. 8, where the upper part is again on the synthetic data and the lower part on the UCI datasets. The curves show that C-DBSCAN still improves DBSCAN, except for DS2 and DS3. We suspect that these datasets contain data points belonging to different partitions but being proximal enough to be put in the same local cluster. Cannot-Link constraints could prevent the formation of such local clusters.

5.1.2 Experiments using only cannot-link constraints

We next experimented with C-DBSCAN using only Cannot-Link constraints. The curves in Fig. 9 (upper part: synthetic datasets, lower part: UCI datasets) show that C-DBSCAN is still superior to DBSCAN, but also that using only one type of constraints yields results of less quality than the combination of both types of constraints.

5.2 Experiments on the query log

We used the query log to study the influence of two constraint-set generation methods. In *Experiment 1*, we disclosed the label of some instances, generated Must-Link

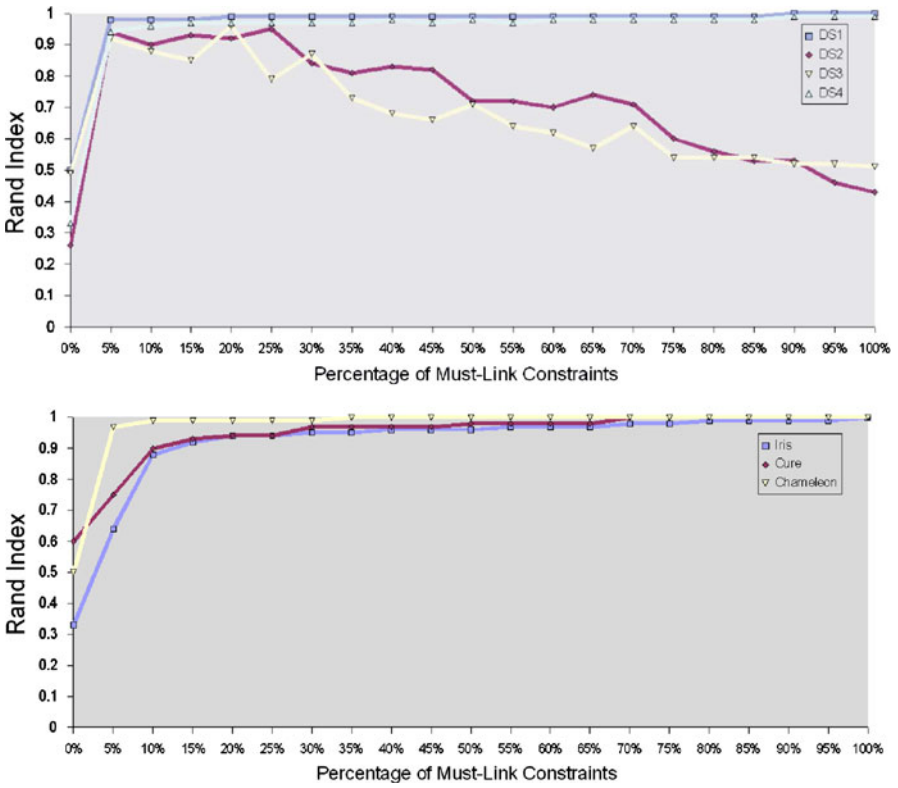

Fig. 8 C-DBSCAN performance when only Must-Link constraints are used: the Rand Index shows improvement over DBSCAN (0% constraints) but performance degrades for the synthetic datasets (upper part of the figure), while the performance on the UCI datasets is rather stable (lower part of the figure)

constraints for some instances having the same label and Cannot-Link constraints for instances having different labels. In *Experiment 2*, we derived association rules and built Must-Link constraints from the extracted associations.

For these experiments, we created 10 subsets of the Query Log á 4,000 records, $QS_1, \ldots, QS_{10}$. For *Experiment 1*, we used datasets $QS_1, \ldots, QS_5$. For *Experiment 2*, we used $QS_6, \ldots, QS_{10}$. We evaluated the clusters built by DBSCAN and C-DBSCAN against the labeled data, using again the Rand Index (Rand 1971) on the similarity between clusters and classes in each of QS_i, $i = 1 \ldots 5$.

5.2.1 Experiment 1

Goal of this experiment was to study how knowledge on the label of some queries may influence clustering. For each QS_i, $i = 1 \ldots 5$, we disclosed some labels, identified queries that had the same label and combined them into Must-Link constraints. This approach was also used in Wagstaff and Cardie (2000), Halkidi et al. (2005), Ruiz et al. (2007a). For each QS_i, we generated a set of 200 Must-Link constraints $ML_{i,[200]}$

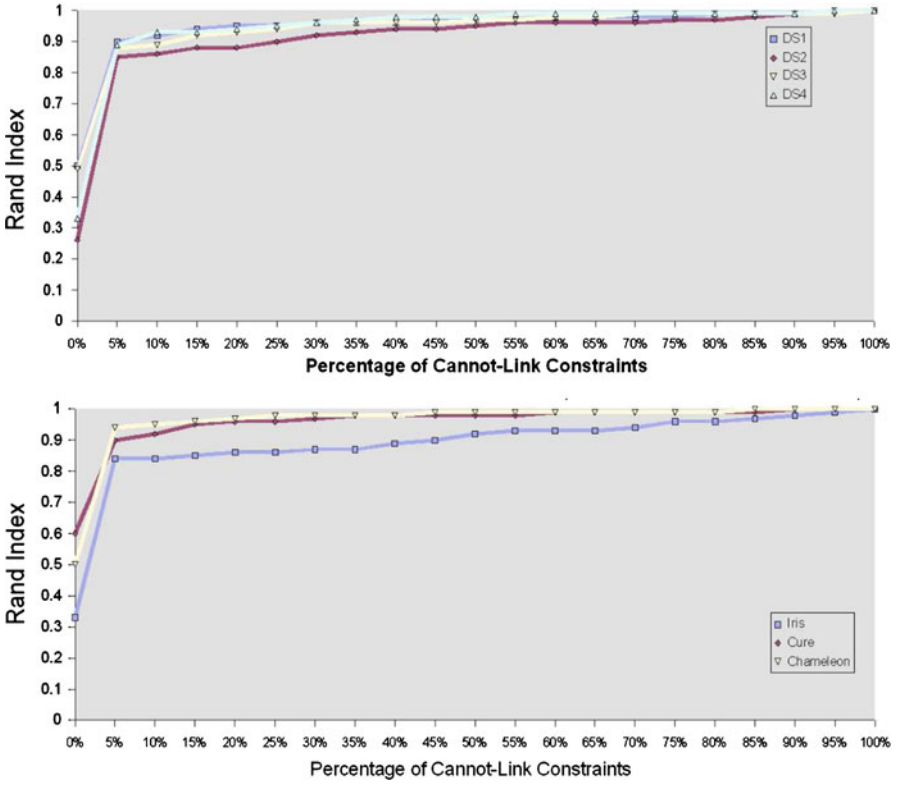

Fig. 9 C-DBSCAN performance when only Cannot-Link constraints are used: the Rand Index shows improvement over DBSCAN for both types of dataset but of lower quality than when both types of constraints are used—cf. Fig. 7

and a superset of 400 Must-Link constraints $ML_{i,[400]}$. Pairs of the selected queries that had different labels gave raise to Cannot-Link constraints as before.

For each QS_i, we repeated the constraint-set generation five times and computed the average Rand Index value. The results are depicted in Fig. 10: C-DBSCAN outperforms DBSCAN and the Rand Index increases with constraint-set size. The improvement varies among the partitions but the trend is the same.

5.2.2 Experiment 2

Goal of this experiment was to study heuristics for constraint generation. We first discovered association rules over the whole query log. From them we formed a constraint-set with 200 Must-Link constraints, $ML_{[200]}$, and a superset of it with 400 Must-Link constraints, $ML_{[400]}$. We used these constraint-sets in the clustering of $QS_6, \ldots, QS_{10}$. A similar method for the evaluation of constraint impact was used in Wagstaff and Cardie (2000), Davidson and Ravi (2005). The results are shown in Fig. 11.

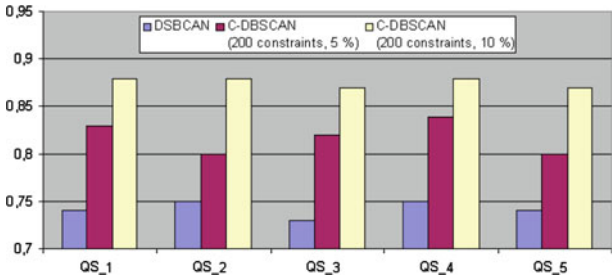

Fig. 10 Performance improvement (Rand Index) of C-DBSCAN over DBSCAN for five partitions of the query log dataset when the constraints are selected randomly

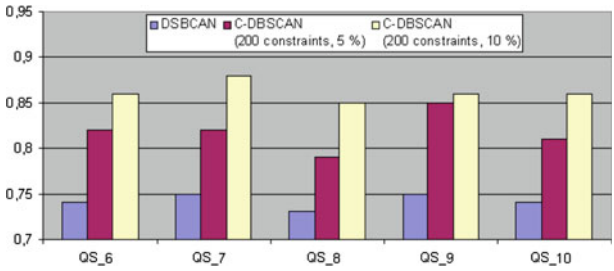

Fig. 11 Performance improvement (Rand Index) of C-DBSCAN over DBSCAN for five partitions of the query log dataset when the constraints are selected heuristically

Similarly to Fig. 10, we see that the constraint-set improves the quality of the clustering results and the larger set of constraints leads to clusters of higher quality. But there are differences in the influence of constraints on quality; we see the highest values for QS_9 and the lowest for QS_8. The Rand Index shows the largest variations for these two partitions: For QS_8, the Rand Index improves by more than 5% when adding 200 constraints, while the Rand Index for QS_9 has almost the same values for $ML_{[200]}$ and $ML_{[400]}$. It seems that despite the random partitioning, there have been spurious structures in the partitions. In that sense, those findings are in lieu with the observations of Wagstaff and Cardie (2000), who observed that the same set of constraints can have different influence upon the clustering results for different datasets.

5.3 Comparison with other methods

We have considered the comparison of C-DBSCAN with PCK-means (Basu et al. 2004a) and with MPCK-means (Bilenko et al. 2004). PCK-means (Basu et al. 2004a) uses constraints for seeding the initial clusters and in the subsequent steps of the algorithm. MPCK-means (Bilenko et al. 2004) incorporates seeding and metric learning in a unified framework. We have used the code provided by the authors under http://www.cs.utexas.edu/users/ml/risc/code/.

We must keep in mind that the basic methods DBSCAN and K-means are different in nature, so one of them may partition the data better than the other, even without consid-

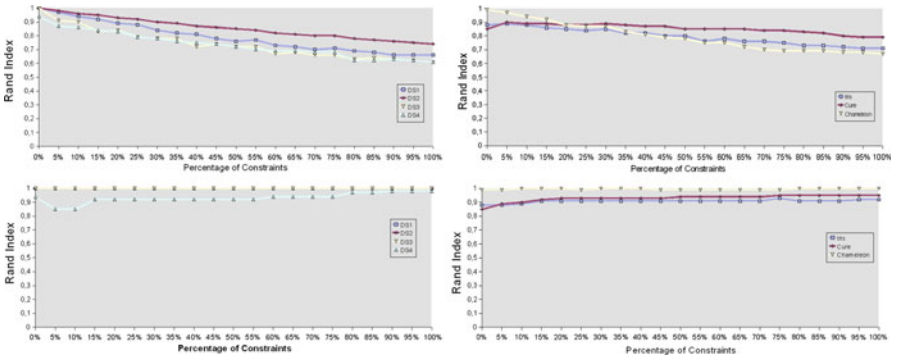

Fig. 12 Performance of MPCK-means (*top*) and PCK-means (*bottom*) for synthetic datasets (*left part*) and UCI datasets (*right part*): the improvement over the baseline is marginal or even negative

ering constraints. This was verified in our preliminary experiments with PCK-means and MPCK-means on the datasets we have used initially for DBSCAN: in Fig. 12 we see that the improvement of the baseline method (0% constraints) is marginal and for some datasets there is even a performance degradation. Furthermore, MPCK-means and PCK-means behave very similarly. We have therefore concentrated on comparing C-DBSCAN with MPCK-means and we have run our comparative experiments on UCI datasets used in Bilenko et al. (2004), namely on the datasrets Iris, Wine, Ionosphere and Protein.

In Fig. 13 we compare C-DBSCAN with MPCK-means on Iris and Wine (upper part) and on Ionosphere and Protein (lower part of the figure).

The comparison reveals many interesting aspects. For Iris and Wine, DBSCAN had inferior performance than K-means (0% constraints), but even a small set of constraints lead to superior performance. At the same time, the performance of MPCK-means deteriorates as the number of constraints increases, while the performance of C-DBSCAN remains stable. These curves indicate that C-DBSCAN exploits constraints better than MPCK-means for these datasets.

For the Ionosphere and Protein datasets, the differences in the exploitation of constraints is more dramatic. As can be seen at the lower part of Fig. 13, K-means was superior to DBSCAN (0% constraints); the trend reverses with even a small set of constraints. For Protein, constraints improve the performance of both algorithms, but the performance of MPCK-means saturates fast, while the performance of C-DBSCAN keeps improving as more constraints are added.

Basu et al. (2004b) suspect that it is easier to learn the metric if the number of constraints is small and that the metric learner may be confused by too many constraints, so more because some of them may be unreliable. On the other hand, too few and unreliable constraints may result in a poor clustering. This indicates an inherent advantage of C-DBSCAN that does not need to learn the metric: unreliable constraints may have only local impact. It can be seen that the quality increase for C-DBSCAN is satisfactory even with few constraints. Since building constraints is a human-resource intensive process, this advantage has practical relevance.

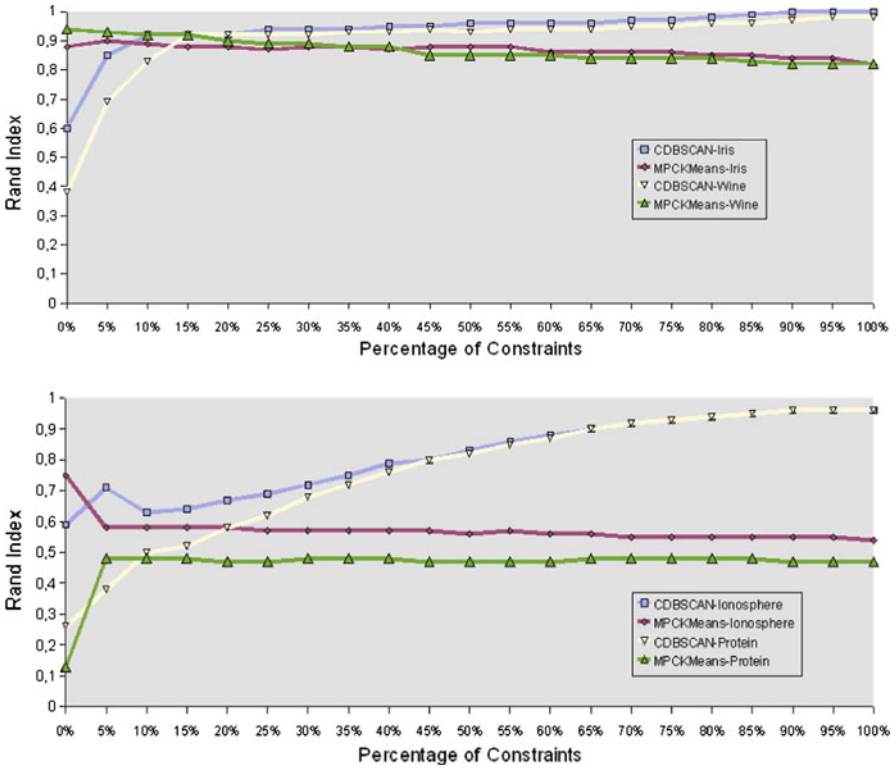

Fig. 13 Impact of constraints upon the performance of C-DBSCAN and MPCK-Means for the UCI datasets Iris and Wine (*upper part* of the figure) and Ionosphere and Protein (*lower part* of the figure)

6 Conclusion

Constraint-based clustering methods exploit background knowledge to guide the grouping of data records into clusters. We have proposed a variation of a density-based clustering algorithm that enforces Must-Link and Cannot-Link constraints. C-DBSCAN is an extension of DBSCAN, an algorithm designed for effective clustering of large and noisy datasets (Ester et al. 1996).

Our C-DBSCAN uses a KD-Tree (Bentley 1975) to partition the dataspace into neighbourhoods with a minimum number of data points and then builds local clusters, in which Cannot-Link constraints are enforced. Must-Link constraints are enforced by merging local clusters. Clusters are further merged, as long as no further Cannot-Link constraints are violated. Our first experimental results have shown that constraints improve the clustering quality substantially, especially on datasets where DBSCAN is known to perform poorly.

In a real-world scenario, constraints on the data are the result of tedious human inspection. We have studied the impact of constraint-sets of different sizes and compared the influence of Must-Link constraints only and of Cannot-Link constraints only. Our results on artificial datasets and on datasets from the UCI repository indicate that a

small number of constraints is sufficient for a remarkable increase in performance and even lead to performance saturation. Using only one type of constraints though leads to lesser performance improvements, so constraints of both types must be devised.

Our experiments on a complex real-world dataset show that performance does not saturate as the number of constraints increases. Furthermore, the performance varies for different samples of the same dataset. However, we have also observed that Must-Link constraints generated through association rules discovery (i.e. in a non-manual way) do improve the clustering performance. This is a promising finding, so we intend to study further methods for constraint-set generation with data mining methods.

Differently from best-effort constraint-based clustering algorithms which allow for constraint violations, C-DBSCAN ensures that all input constraints are satisfied. However, this has the side-effect of generating many singleton clusters, which correspond to Cannot-Link constraints. Such clusters can still be merged with other clusters without any constraint violation. We intend to study heuristics that minimize the number of singleton clusters while still satisfying all constraints.

References

Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high dimensional data for data mining applications. In: SIGMOD´98: proceedings ACM SIGMOD international conference on management of data, pp 94–105

Anand SS, Bell DA, Hughes JG (1995) The role of domain knowledge in data mining. In: CIKM '95: proceedings of the fourth international conference on information and knowledge management, pp 37–43

Angiulli F, Pizzuti C, Ruffolo M (2004) DESCRY: a density based clustering algorithm for very large data sets. In: IDEAL'04: proceedings of intelligent data engineering and automated learning, pp 203–210

Ankerst M, Breunig MM, Kriegel H-P, Sander J (1999) OPTICS: ordering points to identify the clustering structure. In: SIGMOD'99: proceedings of the 1999 ACM SIGMOD international conference on management of data, pp 49–60

Basu S, Banerjee A, Mooney RJ (2002) Semi-supervised clustering by seeding. In: ICML'02: proceedings international conference on machine learning, pp 19–26

Basu S, Banerjee A, Mooney RJ (2004a) Active semi-supervision for pairwise constrained clustering. In: SDM'04: proceedings of the 4th SIAM international conference

Basu S, Bilenko M, Mooney RJ (2004b) A probabilistic framework for semi-supervised clustering. In: KDD'04: proceedings of 10th international conference on knowledge discovery in databases and data mining, pp 59–68

Bennett K, Bradley P, Demiriz A (2000) Constrained K-means clustering. Technical report, Microsoft Research. MSR-TR-2000-65

Bentley JL (1975) Multidimensional binary search trees used for associative searching. Commun ACM 18(9):509–517

Bilenko M, Basu S, Mooney RJ (2004) Integrating constraints and metric learning in semisupervised clustering. In: ICML'04: proceedings of the 21th international conference on machine learning, pp 11–19

Bilenko M, Mooney RJ (2003) Adaptive duplicate detection using learnable string similarity measures. In: KDD'03: proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, New York, NY, pp 39–48

Cohn D, Caruana R, McCallum A (2003) Semi-supervised clustering with user feedback. Technical report TR2003-1892, Cornell University

Davidson I, Basu S (2005) Clustering with constraints. In: ICDM'05: tutorial at the 5th IEEE international conference on data mining

Davidson I, Basu S (2006) Clustering with constraints: theory and practice. In: KDD'06: tutorial at the international conference on knowledge discovery in databases and data mining

Davidson I, Ravi SS (2005) Agglomerative hierarchical clustering with constraints: theoretical and empirical results. In: PKDD'05: proceedings of principles of knowledge discovery from databases, pp 59–70
Davidson I, Ravi SS (2005) Clustering with constraints: feasibility issues and the K-means algorithm. In: SIAM'05: society for industrial and applied mathematics international conference on data mining
Davidson I, Ravi SS, Ester M (2007) Efficient incremental constrained clustering. In: KDD'07: proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, pp 240–249
Davidson I, Wagstaff K, Basu S (2006) Measuring constraint-set utility for partitional clustering algorithms. In: PKDD'06: proceeding of principles of knowledge discovery from databases, pp 115–126
Demiriz A, Bennett KP, Embrechts MJ (1999) Semi-supervised clustering using genetic algorithms. In: ANNIE'99: artificial neural networks in engineering, pp 809–814
Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algortihm for discovering clusters in large spatial database with noise. In: KDD'96: proceedings of 2nd international conference on knowledge discovery in databases and data mining
Guha S, Rastogi R, Shim K (1998) CURE: an efficient clustering algorithm for large databases. In: SIGMOD´98: proceeding of the 1998 ACM SIGMOD international conference on management of data, pp 73–84
Gunopulos D, Vazirgiannis M, Halkidi M (2006) From unsupervised to semi-supervised learning: algorithms and evaluation approaches. In: SIAM'06: tutorial at society for industrial and applied mathematics international conference on data mining
Halkidi M, Gunopulos D, Kumar N, Vazirgiannis M, Domeniconi C (2005) A framework for semi-supervised learning based on subjective and objective clustering criteria. In: ICDM'2005: proceedings of the 5th IEEE international conference on data mining, pp 637–640
Han J, Lakshmanan LVS, Ng RT (1999) Constraint-based, multidimensional data mining. Comput IEEE Computer Soc Press 32(8):46–50
Hinneburg A, Keim DA (1998) An efficient approach to clustering in large multimedia databases with noise. In: KDD'98: proceedings of 4th international conference on knowledge discovery in databases and data mining, pp 58–65
Karypis G, Hang E-H, Kumar V (1999) Chameleon: hierchachical clustering using dynamic modeling. IEEE Comput 12(8):68–75
Klein D, Kamvar SD, Manning C (2002) From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering. In: ICML'02: proceedings of the 19th international conference on machine learning, pp 307–314
Kopanas I, Avouris NM, Daskalaki S (2002) The role of domain knowledge in a large scale data mining projects. In: Vlahavas IP, Spyropoulos CD (eds) Methods and applications of artificial intelligence. Second hellenic conference on AI, SETN 2002, volume 2308 of Lecture Notes in Computer Science. Springer
Newman D, Hettich S, Blake C, Merz C (1998) UCI repository of machine learning databases
Rand WM (1971) Objective criteria for the evalluation of clustering methods. J Am Stat Assoc 66:846–850
Ruiz C, Menasalvas E, Spiliopoulou M (2009) C-DenStream: using domain knowledge over a data stream. In: DS'0r97: proceedings of the international discovery science conference porto, Portugal, Oct 2009. to appear
Ruiz C, Spiliopoulou M, Menasalvas E (2006) User constraints over data streams. In: IWKDDS'06: proceedings of the 4th workshop on knowledge discovery from data streams at ECML/PKDD'06, pp 117–226
Ruiz C, Spiliopoulou M, Menasalvas E (2007a) C-DBSCAN: density-based clustering with constraints. In: RSFDGrC'07: proceedings of the international conference on rough sets, fuzzy sets, data mining and granular computing held by JRS'07
Ruiz C, Spiliopoulou M, Menasalvas E (2007b) Constraint-based query clustering. In: AWIC'07: proceedings of the 5th Atlantic web intelligence conference
Sheikholeslami G, Chatterjee S, Zhang A (1998) WaveCluster: a multi-resolution clustering approach for very large spatial databases. In: VLDB'98: proceedings of 24th international conference on very large data bases, pp 428–439
Vazirgiannis M, Halkidi M, Gunopoulos D (2003) Quality assessment and uncertainty handling in data mining. Springer, LNAI Series
Wagstaff K (2002) Intelligent clustering with instance-level constraints. PhD thesis, Universidad de Cornell

Wagstaff K, Cardie C (2000) Clustering with instance-level constraints. In: ICML'00: proceedings of 17th international conference on machine learning, pp 1103–1110

Wagstaff K, Cardie C, Rogers S, Schroedl S (2001) Constrained K-means clustering with background knowledge. In: ICML'01: proceedings of 18th international conference on machine learning, pp 577–584

Wang W, Yang J, Muntz RR (1997) STING: a statistical information grid approach to spatial data mining. In: VLDB'97: proceedings of the 23rd international conference on very large data bases, pp 186–195

Xing EP, Ng AY, Jordan MI, Russell S (2003) Distance metric learning, with application to clustering with side-information. Adv Neural Inf Process Syst 15:505–512

Zaïane OR, Lee C-H (2002) Clustering spatial data in the presence of obstacles: a density-based approach. In: IDEAS'02: proceedings of the 2002 international symposium on database engineering and applications. IEEE Computer Society, Washington, DC, pp 214–223

Zaïane OR, Lee C-H (2002) Clustering spatial data when facing physical constraints. In: ICDM'02: proceedings of the 2002 IEEE international conference on data mining (ICDM'02). IEEE Computer Society, Washington, DC, p 737