

# Constraint-based Subspace Clustering

Elisa Fromont\*

Adriana Prado<sup>†</sup>

Céline Robardet<sup>‡</sup>

## Abstract

In high dimensional data, the general performance of traditional clustering algorithms decreases. This is partly because the similarity criterion used by these algorithms becomes inadequate in high dimensional space. Another reason is that some dimensions are likely to be irrelevant or contain noisy data, thus hiding a possible clustering. To overcome these problems, subspace clustering techniques, which can automatically find clusters in relevant subsets of dimensions, have been developed. However, due to the huge number of subspaces to consider, these techniques often lack efficiency. In this paper we propose to extend the framework of bottom-up subspace clustering algorithms by integrating background knowledge and, in particular, instance-level constraints to speed up the enumeration of subspaces. We show how this new framework can be applied to both density and distance-based bottom-up subspace clustering techniques. Our experiments on real datasets show that instance-level constraints cannot only increase the efficiency of the clustering process but also the accuracy of the resultant clustering.

## 1 Introduction

Clustering techniques are widely used unsupervised classification techniques to discover groupings of similar objects in data. However, when the dimensionality of the data become too high, usual criteria to define similarity between objects based on distance or density become irrelevant [1]. Besides, some dimensions may be too noisy to clearly identify clusters in the original data. Subspace clustering techniques have been developed to overcome these problems. The idea, related to feature selection or dimension reduction, is to look for clusters in subsets of dimensions.

Subspace clustering techniques have been successfully applied to text mining [18] and gene expression analysis [19]. In these applications, the user may already have some knowledge of how the clustering should be, although he or she may not have enough labeled examples to use a supervised classification method. As pointed out in [15], there exist many ways to describe or constraint a clustering of inter-

est, for example: the expected number of clusters, the minimal or maximal cluster size, weights for different objects and dimensions, constraints on the clustering parameters (density threshold, chosen distance function, entropy threshold etc.), but also instance-level constraints, such as *Must-link* (two objects must be in the same cluster) and *Cannot-link* (two objects must be in different clusters), are useful to take into account user's preferences or background knowledge on a subject. With these two last constraints, the algorithm becomes semi-supervised.

To the best of our knowledge, there exist no constraint-based subspace clustering algorithms, in particular semi-supervised subspace algorithms, even though the integration of instance-level constraints in traditional clustering algorithms has proven to be successful in [30, 21, 7, 28, 26] and for another type of semi-supervised clustering in [9]. In this context, the aim of this work is to investigate how instance-level constraints can influence the subspace clustering process, making it not only more efficient but also more accurate. Towards this goal, we propose to extend the common framework of bottom-up subspace clustering algorithms by integrating instance-level constraints into the mining process. The extended framework is able to consider several evaluation criteria (e.g. density, distance) in order to identify meaningful clusters in the data.

The article is organized as follows. The next section is dedicated to the related work on the subject. Section 3 describes a common framework for subspace clustering and shows how constraints can be added to this framework. Section 4 presents a data mining algorithm called SC-MINER, which can mine subspace clusters under instance-level constraints, and shows how it can be applied to density-based and distance-based subspace clustering techniques. Section 5 shows some results which prove that the integration of constraints into the bottom-up subspace clustering techniques can improve not only their efficiency but also the quality of the resultant clustering. The last section presents some conclusions and future work.

## 2 Related work

In [25], Parson et al. give a survey of many subspace clustering techniques. These techniques can be divided into two categories. Top-down subspace clustering techniques start with an initial approximation of the clusters in the full fea-

\*Université de Lyon, Université de St-Etienne F-42000, UMR CNRS 5516, Laboratoire Hubert-Curien, France

<sup>†</sup>Universiteit Antwerpen, Belgium

<sup>‡</sup>Université de Lyon, CNRS, INSA Lyon, LIRIS, UMR5205, F-69621, France

ture space and iteratively refine the clustering by assigning a weight to each of the dimensions. These techniques do not guarantee to find the best clustering and tend to find only hyper-spherical clusters [25].

On the contrary, bottom-up subspace clustering algorithms start by first detecting interesting clusters in low-dimensional subspaces according to (grid-based) density criteria [2, 23, 12, 27, 19, 5] or a distance measure [22]. The subspace clusters  $(O, D)$  where  $O$  is a set of objects and  $D$  a subspace, *i.e.* a set of dimensions (attributes), are then iteratively merged to form higher-dimensional subspace clusters. The bottleneck of these algorithms is the NP-completeness of the enumeration of the subspace clusters. To make the bottom-up algorithms more efficient, strong constraints have to be pushed in the enumeration process to prune large parts of the search space.

All existing subspace clustering algorithms use input parameters that can be seen as constraints. For example, CLIQUE [2] uses a threshold on the minimum density a cluster can have. Although small changes in these parameters might completely change the resulting clustering, the values of these thresholds are typically never known in advance by the user. On the other hand, more intuitive user constraints, such as knowledge of the a-priori grouping of some objects within clusters has, to the best of our knowledge, not yet been taken into account in subspace clustering. These constraints, known as instance-level constraints (IL constraints), were introduced in [30] and successfully applied to different traditional partitioning or agglomerative clustering algorithms, [21, 7, 26] and predictive clustering trees [28] as well. The authors have shown that although the constraints increase the complexity of the algorithms [13], they allow to obtain clusters that are more meaningful.

In the remainder of this paper we focus on bottom-up subspace clustering algorithms since their exhaustive consideration of the search space allow us to answer constraint user queries without uncertainty.

### 3 Common Framework for Constraint-based Subspace Clustering

Subspace clustering aims at identifying subspace projections of the original dataset, *i.e.* sets of attributes or intervals within the range of the attributes, where relevant clusters of objects can be found.

In this section, we describe a common framework for subspace clustering that motivates the constraint-based algorithm proposed in the next section. As depicted in Figure 1, most of the bottom-up subspace clustering algorithms can be described in terms of a (facultative) pre-processing phase that prepares the data, a data mining phase that mines for clusters and a (facultative) post-processing phase to merge clusters or remove the redundant ones.

In the following, we call  $\mathcal{S}$  a raw dataset where

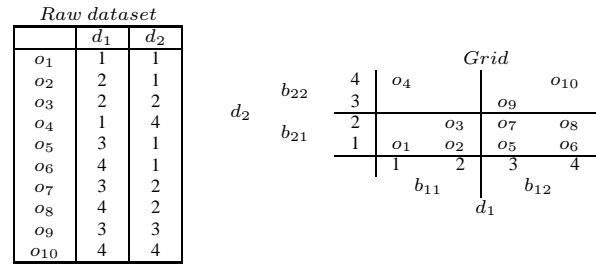


Figure 2: Raw dataset and its resulting partitioning with 2 bins per dimension using CLIQUE.

$\mathcal{O} = \{o_1, o_2, \dots, o_m\}$  is a set of objects and  $\mathcal{D} = \{d_1, d_2, \dots, d_n\}$  a set of symbolic or continuous attributes also called dimensions.

**3.1 Pre-processing** The pre-processing step mainly concerns algorithms which first partition the raw dataset into a (flexible) grid suitable to be used as input by standard itemset mining algorithms (e.g. Apriori [3], Eclat [31], Fp-growth [16]). In this case, a new binary data table is created. The dimensions of the raw data are partitioned into intervals, called bins, which are the attributes of this new table.

The partitioning process depends on the subspace clustering algorithms. For example, grid-based algorithms, such as CLIQUE [2] and its successors (e.g. [23, 12, 27]) start by discretizing each dimension into bins. The number of bins can be user-defined [2, 12, 27] or based on the distribution of the data [23]. The  $j^{th}$  bin of the dimension  $d_i$  is denoted by  $b_{ij}$ ,  $i \in [1, n]$ ,  $j \in [1, p_i]$ , where  $p_i$  is the number of bins of the dimension  $d_i$ . The resulting partitioning of the original data space can be seen as a multi-dimensional grid, where the intersection of exactly one of the bins of every dimension is called a unit:  $u = b_{1l} \times \dots \times b_{nq}$  where  $l \in [1, p_1]$ ,  $q \in [1, p_n]$ . A  $k$ -dimensional unit is a unit created by intersecting  $k$  bins from  $k$  different dimensions. Figure 2 represents the partitioning of the given raw dataset made by CLIQUE when the number of bins is 2. In this example, unit  $\{b_{11}, b_{21}\}$  contains three objects, namely  $o_1$ ,  $o_2$  and  $o_3$ .

After the partitioning, the raw dataset is encoded into a Boolean dataset that states the presence or absence of each object in each of the units. Figure 3 shows the pre-processed dataset constructed by CLIQUE from the grid illustrated in Figure 2. It is worth noting that the pre-processed dataset can be constructed in a different format, such as the vertical format used by SCHISM [27].

Distance-based subspace clustering algorithms, such as NCLUSTER [22], first compute the distance between each object pairs on each dimension. For each dimension  $d_i$ , two different objects are within the same set if the distance between them on that dimension is below a user-defined threshold  $\delta \times R$  (where  $R$  is the range length of  $d_i$ ).

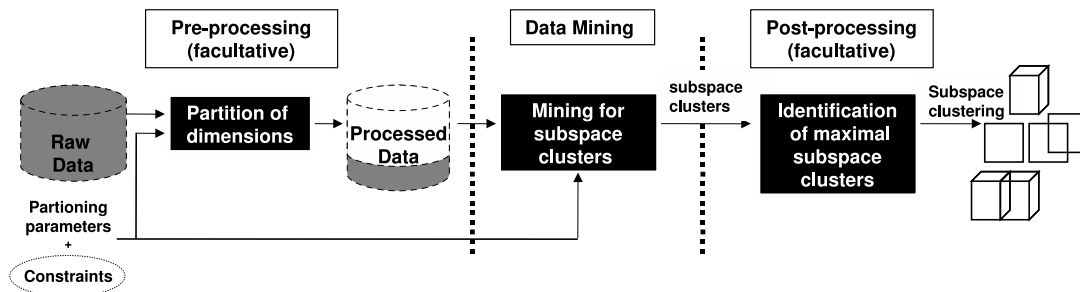


Figure 1: Framework for bottom-up subspace clustering algorithms.

*Boolean Dataset for CLIQUE*

	$b_{11}$	$b_{12}$	$b_{21}$	$b_{22}$
$o_1$	1	0	1	0
$o_2$	1	0	1	0
$o_3$	1	0	1	0
$o_4$	1	0	0	1
$o_5$	0	1	1	0
$o_6$	0	1	1	0
$o_7$	0	1	1	0
$o_8$	0	1	1	0
$o_9$	0	1	0	1
$o_{10}$	0	1	0	1

Figure 3: Boolean dataset constructed by CLIQUE, from the grid in Figure 2.

*Boolean Dataset for NCLUSTER*

	$b_{11}$	$b_{12}$	$b_{13}$	$b_{21}$	$b_{22}$	$b_{23}$
$o_1$	1	0	0	1	0	0
$o_2$	1	1	0	1	0	0
$o_3$	1	1	0	1	1	0
$o_4$	1	0	0	0	0	1
$o_5$	0	1	1	1	0	0
$o_6$	0	0	1	1	0	0
$o_7$	0	1	1	1	1	0
$o_8$	0	0	1	1	1	0
$o_9$	0	1	1	0	1	1
$o_{10}$	0	0	1	0	0	1

Figure 5: Boolean dataset constructed by NCLUSTER, from the bins in Figure 4.

The number of maximal sets that can be created for each dimension  $d_i$  determines the number of bins of  $d_i$ . Figure 4 shows the bins created by NCLUSTER, with  $\delta = 0.5$ , for the raw dataset in Figure 2. In this example, dimensions  $d_1$  and  $d_2$  have 3 bins each. Observe that some objects can end up in multiple bins if they are close to multiple other sets of objects in the same dimension. For instance, object  $o_2$  is within bins  $b_{11}$  and  $b_{12}$ . In this case, we say that the bins are overlapping. In [22], the authors also consider an input parameter to control the overlap between bins.

bins	object sets
$b_{11}$	$\{o_1, o_2, o_3, o_4\}$
$b_{12}$	$\{o_2, o_3, o_5, o_7, o_9\}$
$b_{13}$	$\{o_5, o_6, o_7, o_8, o_9, o_{10}\}$
$b_{21}$	$\{o_1, o_2, o_3, o_5, o_6, o_7, o_8\}$
$b_{22}$	$\{o_3, o_7, o_8, o_9\}$
$b_{23}$	$\{o_4, o_9, o_{10}\}$

Figure 4: Bins constructed by NCLUSTER, for the raw dataset in Figure 2.

Again, the raw dataset is finally encoded into a Boolean table where the Boolean values state the presence or absence of each object in every bin. The corresponding table is shown in Figure 5.

Note that both distance and density-based approaches can be adapted to the use of symbolic dimensions by creating

one bin for each different value of each dimension.

**3.2 Data Mining** Bottom-up algorithms start by enumerating the relevant 1-dimensional subspaces and iteratively generate higher dimensional ones that satisfy the cluster criteria [2, 23, 12, 27, 19, 5, 22]. To make the enumeration process efficient, the criteria should consist in anti-monotonic constraints that are used to prune the search space.

When the dimensions have been partitioned beforehand (see Subsection 3.1), the enumeration is, in fact, not done on subspaces but on candidate subspace clusters. A subspace cluster is a  $k$ -dimensional unit that has, for example in the case of density-based clustering, a high density of objects. The density constraint is provided to the algorithms. If we regard the pre-processed dataset as transactional data, then mining for  $k$ -dimensional dense units reduces to the task of mining frequent  $k$ -itemsets [3]. In our case, the transaction set is equal to the object set  $\mathcal{O}$  and the items are the previously computed bins. In the following, instead of using the term itemset, we will denote the output of this mining task by *binset*. A  $k$ -dimensional unit with density greater than  $\sigma$  can be viewed as a frequent binset of size  $k$  with a frequency greater than  $\sigma$ . We can observe that as bins do not overlap, two different bins of the same dimension cannot occur in the same frequent binset and thus the computed frequent binset can be viewed as a dense  $k$ -dimensional unit. CLIQUE

takes advantage of the anti-monotonic property of the density constraint: if a  $k$  dimensional space is dense, then any of its  $(k - 1)$ -dimensional subspaces is also dense [2].

In the case of the distance-based algorithm, the bins may overlap. This can result in more bins than in the density-based approaches, increasing the complexity of the algorithm. To overcome this problem, NCLUSTER directly mines for maximal clusters, by adopting a closed itemset mining strategy [29]. NCLUSTER uses the following anti-monotonic property: if a set of objects  $O$  and a set of bins  $D$  form a  $\delta - nCluster$  (i.e. for every two objects  $o_w$  and  $o_z$  of  $O$  and every bin  $b_{ij} \in D$ ,  $o_w$  and  $o_z$  are neighbors w.r.t. the distance measure on  $d_i$  and the threshold  $\delta$ ), then  $O$  forms a  $\delta - nCluster$  with every subset of  $D$  and  $D$  forms a  $\delta - nCluster$  with every subset of  $O$ . Additionally, it regards a subspace cluster as being meaningful if it is composed by  $m_r$  objects and  $m_c$  dimensions ( $m_c$  and  $m_r$  are provided to the algorithm).

We believe that the addition of more constraints to this data mining step can make the clustering process not only more efficient but also more accurate. In Section 4, we present this idea in more details.

**3.3 Post-processing** The post-processing step can have several purposes. One of them is to identify maximal clusters and generate their corresponding descriptions. CLIQUE, for example, generates  $k$ -dimensional-maximal clusters by connecting  $k$ -dimensional dense units (obtained in the previous step) that have common faces. In the case of NCLUSTER, the applied closed set mining algorithm computes a closed set of bins associated to a closed set of objects, but some of them can actually be not maximal while considering the dimensions instead of the bins computed on them. Therefore, non-maximal subspace clusters are removed in a post-processing step.

At the end of this step, the maximal clusters are described as a pair  $(O, D)$  where  $O$  is a set of objects and  $D$  a set of bins of different dimensions (subspace). In the case of NCLUSTER, the bins  $b_{ij}$  are substituted by the corresponding original dimension  $d_i$ . For example, the maximal cluster  $(\{b_{11}, b_{21}\}, \{o_1, o_2, o_3\})$  becomes  $(\{d_1, d_2\}, \{o_1, o_2, o_3\})$ .

This step can also be used to prune redundant subspace clusters or those that, for example, do not fulfill certain constraints that could not be efficiently enforced into the mining process (e.g., non-monotonic constraints).

#### 4 Constraint-based Subspace Clustering Algorithm

In this section, we present SC-MINER, a data mining algorithm that mines subspace clusters under instance-level constraints and can be integrated into the common framework presented in the previous section.

As pointed out in Subsection 3.2, the exploitation of anti-monotonic constraints within the existing subspace clus-

tering algorithms increases their efficiency, by pruning the search space. For existing bottom-up subspace clustering algorithms, the few constraints currently considered do not attempt to increase the accuracy of the final clustering by integrating expert knowledge, since they are mainly related to the parameters of the algorithms, not taking into account external information. Motivated by this need, we propose to extend the common framework for bottom-up subspace clustering techniques, by integrating IL constraints such as Must-link and Cannot-link constraints into the data mining step, so as to obtain not only more efficient algorithms but also more accurate results. These two constraints enable the end-user to guide the unsupervised subspace clustering task by adding some expert knowledge.

In a subspace clustering, we formally define these constraints as follows:

**DEFINITION 1. (CANNOT-LINK)** A Cannot-link constraint between two objects  $o_i$  and  $o_j$  written  $CL(o_i, o_j)$  is satisfied by a subspace clustering  $SC$  iff for every subspace cluster  $(O, D) \in SC$ ,  $\{o_i, o_j\} \not\subseteq O$ .

**DEFINITION 2. (MUST-LINK)** A Must-link constraint between two objects  $o_i$  and  $o_j$  written  $ML(o_i, o_j)$  is satisfied by a subspace clustering  $SC$  iff for every subspace cluster  $(O, D) \in SC$ ,  $\{o_i, o_j\} \subseteq O$  or  $\{o_i, o_j\} \cap O = \emptyset$ .

The following properties show how Must-link and Cannot-link constraints can be used to efficiently prune the enumeration of the subspace clusters.

**PROPERTY 4.1.** The Cannot-link constraint  $CL(o_i, o_j)$  is anti-monotonic w.r.t.  $\subseteq$ :  $\forall P \subseteq O: \{o_i, o_j\} \not\subseteq O \Rightarrow \{o_i, o_j\} \not\subseteq P$ .

**PROPERTY 4.2.** The Must-link constraint  $ML(o_i, o_j)$  is a disjunction of a monotonic and a anti-monotonic constraints w.r.t.  $\subseteq$ .  $\{o_i, o_j\} \subseteq O$  is monotonic and  $\{o_i, o_j\} \cap O = \emptyset$  is anti-monotonic:

- $\forall P \subseteq O: \{o_i, o_j\} \subseteq P \Rightarrow \{o_i, o_j\} \subseteq O$
- $\forall P \subseteq O: \{o_i, o_j\} \cap O = \emptyset \Rightarrow \{o_i, o_j\} \cap P = \emptyset$

Property 4.1 states that to fulfill the constraint  $CL(o_i, o_j)$ , we should look for subspace clusters where the objects  $o_i$  and  $o_j$  are never present together. Since a  $k$ -dimensional cluster can never contain more objects than any of its  $k - 1$ -dimensional projections, the Cannot-link constraint is anti-monotonic.

Property 4.2 states that to fulfill the constraint  $ML(o_i, o_j)$ , we should look for subspace clusters where the objects  $o_i$  and  $o_j$  are either both present or both absent. If one of the objects is present but not the other, the subspace cluster is not relevant. As for  $CL(o_i, o_j)$ , since a  $k$ -dimensional

cluster can never contain more objects than any of its  $k - 1$ -dimensional projections, the Must-link constraint is a disjunction of an anti-monotonic constraint ( $o_i$  and  $o_j$  are in the same subspace cluster) and a monotonic constraint (objects  $o_i$  and  $o_j$  are never present in the same subspace cluster).

However, the efficient use of these constraints may depend on the enumeration process of the algorithms. Indeed, algorithms such as SUBCLU [19] or DUSC [5] directly detect clusters in the enumerated subspace by applying a DBSCAN-like algorithm. As a consequence, IL constraints should be pushed directly into that algorithm. This has already been tackled, for example, in C-DBSCAN [26] and will not be developed in this paper. Instead, we focus on the family of algorithms which partition the data beforehand.

As mentioned in Subsection 3.2, the existing bottom-up techniques use a (closed) itemset mining algorithm to enumerate the possible subspace clusters. Usual itemset mining algorithms (e.g., Apriori [3]), however, do not handle anti-monotonic constraints together with monotonic constraints, since pushing monotone constraints can lead to a reduction of anti-monotone pruning [17, 10]. However, the computation of subspace clusters  $(O, D)$  satisfying IL constraints requires the use of monotonic and anti-monotonic constraints on the set of bins  $D$  as well as on the object set  $O$ .

In this context, we have thus decided to extend D-MINER [6], a closed-set mining algorithm that can take into account both types of constraints, towards SC-MINER, which handles IL constraints and enumerates both bins and objects. We opted to mine only closed itemsets, to avoid redundant subspaces that can occur with previous techniques.

Observe that the proposed extension can be applied to either density or distance-based bottom-up algorithms, since it affects only the data mining step of the whole framework. Moreover, the current exploited anti-monotonic constraints can still be considered here.

**4.1 Candidate generation** The core technique used by SC-MINER to handle IL constraints is based on the “divide-and-conquer” generic algorithm proposed in [6, 14]. In a nutshell, SC-MINER recursively enumerates, in depth-first manner, all subspace clusters  $(O, D)$  that contain an element (object or bin)  $a$ , and then all subspace clusters that do not contain  $a$ . During the enumeration process, elements that have already been enumerated are differentiated to the ones that still have to be enumerated. To this end, we define a candidate of the enumeration process by a triplet  $\langle X, Y, N \rangle$  composed of three couples of sets:

- the couple  $X = (O, D)$  is the set of objects and the set of bins contained in the candidate and its descendants (those obtained by recursive calls). These elements have already been enumerated as members of the subspace clusters under construction.

- the couple  $Y = (O', D')$  contains the objects and the bins that still have to be enumerated.
- the couple  $N = (O_N, D_N)$  is used to ensure the closeness constraint, which is neither monotonic nor anti-monotonic. These elements have already been enumerated as elements that do not belong to any subspace cluster under construction.

---

**Algorithm 1** SC-MINER  $(\langle X, Y, N \rangle)$

---

```

if not PRUNE( $\langle X, Y, N \rangle$ ) then
  if  $Y = (\emptyset, \emptyset)$  then
    output  $X = (O, D)$ 
  else
    Choose  $a$  from  $Y$ 
    SC-MINER( $\langle X, Y \setminus \{a\}, N \cup \{a\} \rangle$ )
    PROPAGATION( $X, Y$ )
    SC-MINER( $\langle X \cup \{a\}, Y \setminus \{a\}, N \rangle$ )
  end if
end if

```

---

The pseudo-code of the algorithm is given in Algorithm 1. At the first line, the pruning function (PRUNE) presented in Subsection 4.2 is called to check if the recursion process can be stopped. If the recursion is not stopped, the second test evaluates if there are still elements to be enumerated. If not, the candidate is a valid subspace cluster that is thus output. Otherwise, an element  $a$  is picked up from  $O'$  or  $D'$  and the function is recursively called twice: once without  $a$  in the candidate and its descendants, and once with  $a$ . In both calls,  $a$  is removed from  $Y$ , but before the second one, the PROPAGATION function is called to push the IL constraints as explained in Subsection 4.3.

Subspace clusters are made of maximal sets of objects and bins that are in relation: each object of  $O$  must belong to the  $k$ -dimensional unit defined by  $D$  and each bin of  $D$  must contain every object of  $O$ . This is also handled by the function called PROPAGATION (see Subsection 4.3). The couple of sets  $N$  is used in the PRUNE function to ensure that the subspace clusters are maximal, as explained in Subsection 4.2.

For the first call,  $X = (\emptyset, \emptyset)$  and  $Y = (\mathcal{O}, \mathcal{B})$ , where  $\mathcal{O}$  is the original set of objects and  $\mathcal{B}$  is the whole set of bins.

**4.2 Pruning** Let us now explain how the density-based subspace constraint and the  $\delta$ -nCluster constraint are checked in SC-MINER.

SC-MINER algorithm can push monotonic and anti-monotonic constraints on the set of objects or on the set of bins. This interesting property relies on the fact that, from a given candidate  $\langle X, Y, N \rangle$ , the subspace clusters  $(O_i, D_i)$  that can be generated are those that satisfy:

$$O \subseteq O_i \subseteq O \cup O' \text{ and } D \subseteq D_i \subseteq D \cup D'$$

As in many data mining algorithms, the contraposition of the monotonic and anti-monotonic constraints is checked to eventually stop the recursion process and thus prune the search space:

- Recall that if  $q$  is monotone,  $A$  and  $B$  sets, thus  $\forall A \subseteq B: \neg q(B) \Rightarrow \neg q(A)$ . In SC-MINER, the monotonic constraints are evaluated on the couple of sets  $(O \cup O', D \cup D')$ , abbreviated in the following by  $X \cup Y$ . If the evaluation of the constraint on  $X \cup Y$  is false ( $\neg q(X \cup Y)$ ), we have the guarantee that every subspace cluster generated from this candidate can be pruned since none of them satisfies  $q$ . For example, if  $\{o_i, o_j\} \not\subseteq (O \cup O')$ , then  $\{o_i, o_j\}$  does not belong to any subset of  $O \cup O'$  and consequently to any subspace clusters that are under construction.
- Recall that if  $q$  is anti-monotone,  $A$  and  $B$  sets, thus  $\forall A \subseteq B: \neg q(A) \Rightarrow \neg q(B)$ . In SC-MINER, anti-monotone constraints are evaluated on the couple of sets  $X = (O, D)$ . If  $q(X)$  is false, we have the guarantee that every subspace cluster generated from this candidate can be pruned since none of them satisfies  $q$ . For example, if  $\{o_i, o_j\} \cap O \neq \emptyset$ , we have  $\{o_i, o_j\} \cap O_i \neq \emptyset, \forall O_i \subseteq O \cup O'$ .

The density-based subspace constraint consists in considering binsets  $(O, D)$  that have a density greater than  $\sigma$  i.e.  $\frac{|O|}{|O|} \geq \sigma$ . This constraint is monotonic w.r.t.  $\subseteq$  and thus if  $|O \cup O'| < \sigma \times |O|$ , the function PRUNE return **true**. The constraint ensuring that  $(O, D)$  forms a binset (i.e. whether there exists  $o \in O$  and  $d \in D$  such that  $o$  and  $d$  are not in relation in the dataset) is guaranteed by the PROPAGATION function (see Subsection 4.3).

As explained in [22], the  $\delta$ -nCluster are closed binsets supported by a minimum number of objects  $m_r$ . SC-MINER algorithm checks the closeness constraint by ensuring that every element that has been discarded during the candidate generation process cannot be added to the current candidate, i.e. it has to check whether each such element, in  $N$ , is not in relation with at least one element of  $X \cup Y$ . Otherwise, the candidate and all its descendants are not closed (since this element from  $N$  can be added to form a larger binset having the same support) and they can be safely pruned. Furthermore, if  $|O \cup O'| < m_r$ , the function PRUNE return **true**.

**4.3 Propagation** SC-MINER ensures that  $(O, D)$  is a binset in the following way: when an element  $a$  is enumerated and put in  $X$ , the PROPAGATION function removes all the elements of  $Y$  that are not in relation with  $a$  in the dataset.

Furthermore, the PROPAGATION takes advantages of the two IL constraints as described below:

- For a Cannot-link constraint  $CL(o_i, o_j)$ , when the can-

didate  $\langle X \cup \{o_i\}, Y \setminus \{o_i\}, N \rangle$  is generated, the PROPAGATION function automatically removes  $o_j$  from  $Y$ .

- For a Must-link constraint  $ML(o_i, o_j)$ , when the candidate  $\langle X \cup \{o_i\}, Y \setminus \{o_i\}, N \rangle$  is generated, the PROPAGATION function automatically pushes  $o_j$  in  $X$  and removes the element from  $D'$  that are not in relation with  $o_j$ . When the candidate  $\langle X, Y \setminus \{o_i\}, N \cup \{o_i\} \rangle$  is generated, the PROPAGATION function automatically removes also  $o_j$  from  $Y$ .

Next, we provide some experiments on the usefulness of the proposed algorithm to improve the techniques CLIQUE and NCLUSTER. We show that the use of Must-Link and Cannot-Link constraints increases the relevance of the computed subspace clusters. We also show that these constraints enable the computation of such clusters in some contexts where it would not be possible for the aforementioned techniques to provide any results.

## 5 Experimental Results

This experimental section aims at investigating the advantages of the proposed contributions with respect to existing subspace clustering methods. In particular, we would like to answer five main questions:

1. Does the generation of frequent closed patterns instead of frequent patterns modify the performance of the CLIQUE [2] method?
2. Does the use of instance-level constraints influence the efficiency of existing subspace clustering methods?
3. Does the use of instance-level constraints influence the quality of the resulting subspace clustering?
4. Does SC-MINER scale on real high-dimensional data?
5. Does the use of instance-level constraints lead to the computation of meaningful clusters?

To answer these questions, we have implemented three subspace clustering methods.

- CLIQUE: the original technique described in [2] re-implemented in C++ using the implementation of APRIORI available in <sup>1</sup>.
- SC-CLIQUE: an extension of CLIQUE that uses SC-MINER to handle IL constraints.
- SC-NCLUSTER: an extension of NCLUSTER that also uses SC-MINER, with which the search of subspace clusters can be guided by IL constraints.

<sup>1</sup><http://www.adrem.ua.ac.be/~goethals/software/>

Notice again that the implementation differences between SC-CLIQUE and SC-NCLUSTER only lie in the pre-processing and post-processing phases as discussed in Section 3.

All experiments were done on a Pentium 3 with 2 Giga of memory running on Linux. We used four real benchmark datasets with numerical attributes and a real gene expression dataset, called PLASMODIUM [11]. The characteristics of the benchmark datasets and the gene expression dataset are presented in Subsections 5.1 and 5.2, respectively.

For every dataset, we only consider the descriptive attributes for clustering, while the class attribute is used in order to generate IL constraints. As in [28], the constraints are generated as follows: two instances are picked up at random. If they belong to the same class, then the constraint becomes a must-link constraint; if they belong to two different classes then a cannot-link constraint is generated between those two examples. We augment the set of constraints by adding all entailed constraints. In other words, for each pair of constraints, we generate all the constraints that can be deduced by transitivity:

- For two constraints  $ML(o_i, o_j)$  and  $ML(o_i, o_k)$ , the constraint  $ML(o_j, o_k)$  is added.
- For two constraints  $ML(o_i, o_j)$  and  $CL(o_i, o_k)$ , the constraint  $CL(o_j, o_k)$  is added.

**5.1 Benchmark datasets** In this section, we present the results obtained over the benchmark datasets, whose characteristics are shown in Table 1.

Dataset	#objects	#dimensions	#classes	source
PENDIGITS	7494	16	10	[24]
IMAGE	2310	19	7	[24]
GLASS	214	9	7	[24]
SHAPES	160	17	9	[20]

Table 1: Benchmark datasets used in the experiments and its corresponding characteristics.

**5.1.1 Non-closed vs. closed patterns** This section aims at evaluating the impact of extracting frequent closed itemsets instead of frequent itemsets in the CLIQUE algorithm. To evaluate the results, we compare the number of resulting subspace clusters as well as the quality of the resulting clustering generated by CLIQUE and SC-CLIQUE.

As in [5], the resulting clustering is evaluated in terms of quality and coverage. The *quality* measure takes into account the purity of the subspace clustering  $C$  w.r.t. the class values and it is determined, for each subspace cluster,

by the entropy:

$$(5.1) \quad H(\mathbf{C}) = - \sum_{i=1}^k p(i|\mathbf{C}) \cdot \log(p(i|\mathbf{C}))$$

where  $p(i|\mathbf{C})$  is the proportion of objects of  $C$  that are labeled by the  $i^{th}$  class and  $k$ , the number of different classes. For readability, the authors of [5] take the inverse entropy and normalize it by dividing it by the maximum entropy:

$$(5.2) \quad (1 - H(\mathbf{C})/\log(k))$$

To obtain the global quality value for a subspace clustering, we compute the average entropy of each subspace cluster weighted by the number of objects per subspace cluster and expresses it in percentage. Notice that the number of “true” clusters for those datasets corresponds to the number of classes (see Table 1). Subspace clustering algorithms, as well as clustering algorithms for which the number of clusters is not given beforehand, tend to find more than one clusters per class. To evaluate the quality of the results, the used *quality* measure is more appropriate than looking for an exact match with the true clusters.

The *coverage* measure is the percentage of objects in any subspace cluster, which indicates the ratio of clustered objects to noise.

As expected theoretically, Fig. 6 shows that the number of resulting subspace clusters generated by SC-CLIQUE is always equivalent or lower than the number of clusters generated by CLIQUE. What is more surprising is that the quality is equivalent or higher, for the same coverage value (always 100%). On PENGIDITS, both methods provide the same results. In the case of IMAGE and GLASS datasets, SC-CLIQUE generates less subspace clusters than CLIQUE when the minimum density  $\sigma$  is low with a higher quality. Finally, on SHAPES, CLIQUE was not able to complete the execution in less than two hours for a density  $\sigma$  of 10%.

The obtained results show that enumerating only closed itemsets is sufficient to obtain meaningful subspace clustering while providing better scalability to the techniques.

**5.1.2 Efficiency** In this section, we evaluate the efficiency of the implemented methods, by comparing the number of candidate subspace clusters generated during the enumeration process, for different numbers of IL constraints. Since the IL constraints are picked randomly, the results are averaged for 60 different runs with different random constraints. Table 2 shows the thresholds set for each dataset according to the methods. We opted to consider low density  $\sigma$  (SC-CLIQUE) and  $\delta$  (SC-NCLUSTER) thresholds in order to get a reasonable number of final subspace clusters and, consequently, be able to notice the behavior of the methods when different numbers (depending on the number of objects in the dataset) of IL constraints are considered.

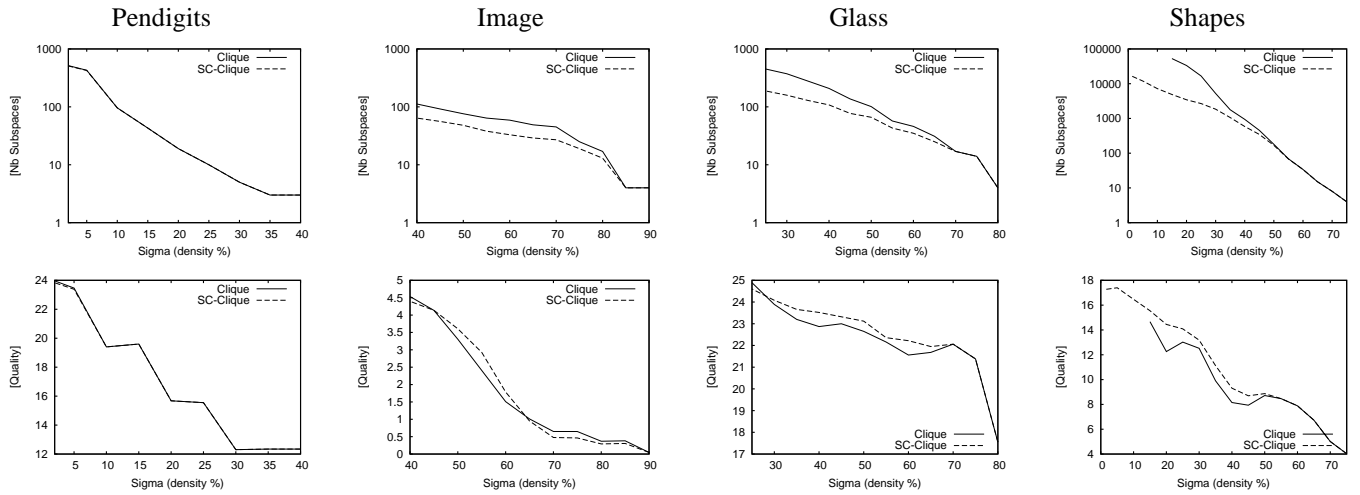


Figure 6: CLIQUE vs. SC-CLIQUE (without instance-level constraints).

SC-CLIQUE		
Dataset	$\sigma$	Nb bins
PENDIGITS	0.67%	10
IMAGE	0.43%	4
GLASS	0.47%	20
SHAPES	1.25%	4

SC-NCLUSTER			
Dataset	$\delta$	$m_r$	$m_c$
PENDIGITS	0.1%	80	2
IMAGE	0.75%	20	2
GLASS	0.02%	20	2
SHAPES	1.75%	10	2

Table 2: Thresholds used for the computation of Figures 7, 8, 9 and 10.

Figures 7 and 8 show the results obtained for SC-CLIQUE and SC-NCLUSTER, respectively. Observe that SC-NCLUSTER, by allowing overlapping bins, generates more candidates. Nevertheless, for both methods, the number of candidates generated decreases in inverse proportion to the number of IL constraints. As a result, we can conclude that IL constraints are useful, in addition to other monotonic criteria, to improve the efficiency of the methods. Besides, note that for GLASS and SHAPES datasets, the difference between the number of candidates and the number of final subspace clusters remains equal to one order of magnitude as the number of constraints increases. It is an indication that the new constraints are useful to prune the search space in a significant way.

**5.1.3 Quality and Coverage** In this section, we evaluate whether the use of IL constraints leads to more meaningful

clusters. The relevance of the resulting subspace clustering is evaluated through the quality and coverage measures presented in Subsection 5.1.1. Figures 9 and 10 show the results obtained for SC-CLIQUE and SC-NCLUSTER, respectively.

Observe that the quality of the resulting clustering increases along with the number of constraints, in almost all experiments. The only exception is SC-NCLUSTER over the IMAGE dataset. In this case, some small fluctuations in the quality values can be observed: adding Must-Link or Cannot-Link constraints slightly increases or decreases the quality of the clustering. This can be explained by the fact that the constraints are randomly generated, pruning a higher or lower number of subspace clusters depending on the distribution of the data.

It is also worth noting that, as depicted in Figures 7 and 8, the use of IL constraints enables a significant reduction of the number of generated subspace clusters.

We can therefore conclude that the new constraints enable the extraction of a smaller collection of subspace clusters that are of higher quality than those extracted without considering them.

Concerning the coverage values, we can see that Must-Link and Cannot-Link constraints can have an important impact: by discarding subspace clusters that do not satisfy these constraints, this approach provides subspace clusterings that do not cover the whole set of objects. In our experiments, we considered varying numbers of constraints in order to validate the robustness of our approach. In real situation, however, an expert would have provided a lower number of constraints.

In summary, we can conclude that IL constraints implemented in both a density-based and a distance-based sub-

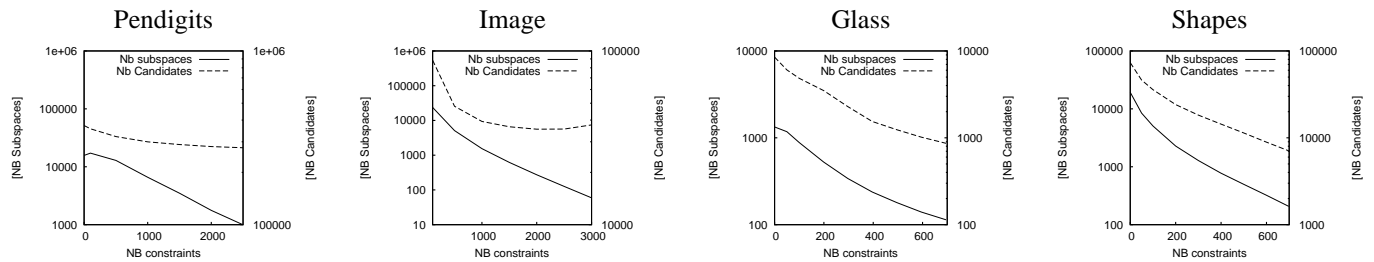


Figure 7: SC-CLIQUE: number of candidates and resulting subspace clusters as the number of instance-level constraints increases.

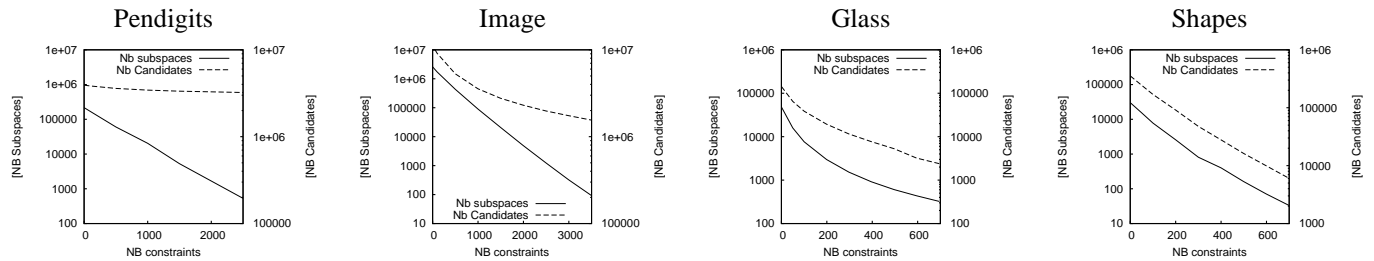


Figure 8: SC-NCLUSTER: number of candidates and resulting subspace clusters as the number of instance-level constraints increases.

space clustering methods can improve the relevance of the resulting subspace clustering.

**5.2 Gene Expression Dataset** In this section, we aim at answering the two last questions posed at the beginning of Section 5. To this end, we ran experiments over a real high dimensional gene expression data, called PLASMODIUM [11], which measures the transcriptome of the intraerythrocytic developmental cycle of the Plasmodium Falciparum, a parasite that causes the human malaria. For our experiments, we consider 46 biological samples (46 objects). Each sample corresponds to the expression profiles of 476 genes<sup>2</sup> of a population of such parasites, which were evaluated in a specific time point of the developmental cycle. After the invasion of the red blood cells, the developmental cycle is divided into three phases: the Ring (until the 17<sup>th</sup> time point), Trophozoite (until the 29<sup>th</sup> time point) and Schizont phases, as shown on Figure 11. These three phases determine the class of each of the samples.

**5.2.1 Scalability** To evaluate the scalability of our proposed algorithm, we ran SC-CLIQUE and SC-NCLUSTER over the PLASMODIUM dataset using different numbers of IL constraints. Again, the thresholds were set so as to no-

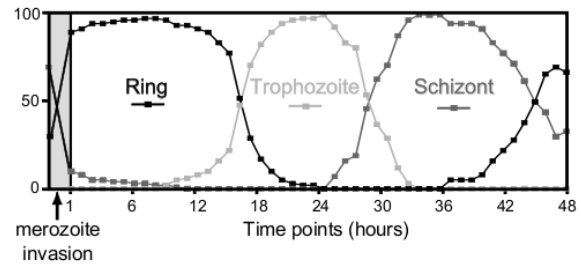


Figure 11: Major developmental phases of Plasmodium Falciparum parasites (drawing from [11]). The three curves represent the percentage of parasites (y-axis) that are at the Ring, Trophozoite, or Schizont phase, at every time point.

tice the behavior of the subspace clustering methods when different numbers of IL constraints are considered.

As shown in Figure 12, for both methods, the number of final subspace clusters decreases in inverse proportion to the number of IL constraints. As a consequence, the total execution time also decreases, which indicates the improvement in efficiency offered by the methods with the use of the new constraints.

Considering the quality of the final clustering, SC-CLIQUE was able to extract subspace clusters with higher quality and higher coverage than those extracted by SC-

<sup>2</sup>The selected genes are known to play an important role in the developmental cycle of the Plasmodium.

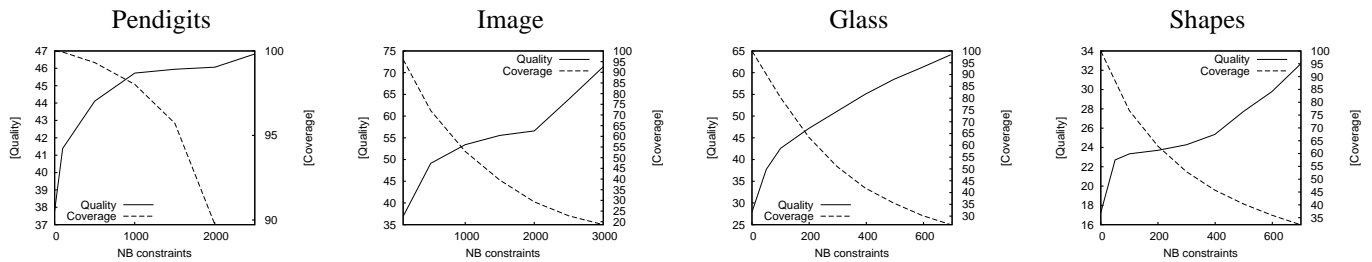


Figure 9: SC-CLIQUE: quality (%) and coverage (%) as the number of instance-level constraints increases.

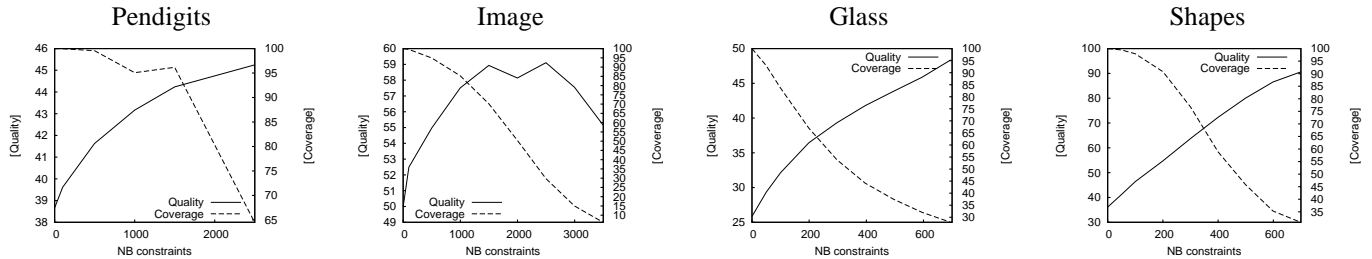


Figure 10: SC-NCLUSTER: quality (%) and coverage (%) as the number of instance-level constraints increases.

NCLUSTER. This can be justified by the fact that, for this experiment, SC-NCLUSTER generates a high number of overlapping bins, which contributes to the low quality and coverage of the final clustering. Yet, for both methods, the quality of the resulting clustering increases along with the number of constraints.

In order to also evaluate how efficient SC-CLIQUE (the best implementation for the experiments done in this section) was in comparison to CLIQUE, we ran both algorithms with the same parameters (number of bins set to 4 and minimum density  $\sigma$  set to 26%), providing 50 randomly generated constraints for SC-CLIQUE. Not surprisingly, CLIQUE was not able to finish the computation, running out of memory while generating candidates of size 5. This is mainly due to the fact that CLIQUE does not prune as many candidates as SC-CLIQUE, as SC-CLIQUE takes advantage of IL constraints and uses a more efficient enumeration strategy, that is, it enumerates objects as well as bins.

**5.2.2 Evaluation of Computed Subspace Clusters** This section evaluates the meaningfulness of some computed clusters extracted from the PLASMODIUM data. For this dataset, relevant subspace clusters should be of the form  $(O, D)$ , where  $O$  is a set of samples belonging to the same class (the same developmental phase) and  $D$  a set of genes that are known to be related to the developmental phase represented by  $O$ .

For this evaluation, we selected, from the result of SC-CLIQUE in the last experiment reported in the previous sec-

tion, the subspace clusters that contained at least 35 dimensions, that is, 35 genes. We obtained 26 different such subspace clusters having a quality of 77.18% and a coverage of 91.30%. Each subspace cluster contained consecutive samples. Seven of these subspace clusters concerned samples corresponding to the Ring phase, two corresponding to the Trophozoite phase, eight were pure Schizont subspace clusters and finally nine were Schizont samples plus the two initial samples.

Table 3 presents the average number of genes (according to their corresponding functional group) present in each of the selected subspace clusters (grouped by the corresponding developmental phase they concern). The values that match the biological knowledge, illustrated in Figure 13, are in bold. We can observe that the subspace clusters gathered genes whose functions are known to be important in the corresponding developmental phase. For instance, the cytoplasmic translation and the transcription machinery are known to be active in the Ring and early Trophozoite phases. The proteasome functions play an important role during the Trophozoite phase. Analogously, the plastid genome, mitochondrial, deoxynucleotide synthesis and dna replication are functions that characterize the Schizont phase.

From this experiment, we can conclude that our proposed algorithm is capable to compute meaningful subspace clusters over a real high-dimensional gene expression data.

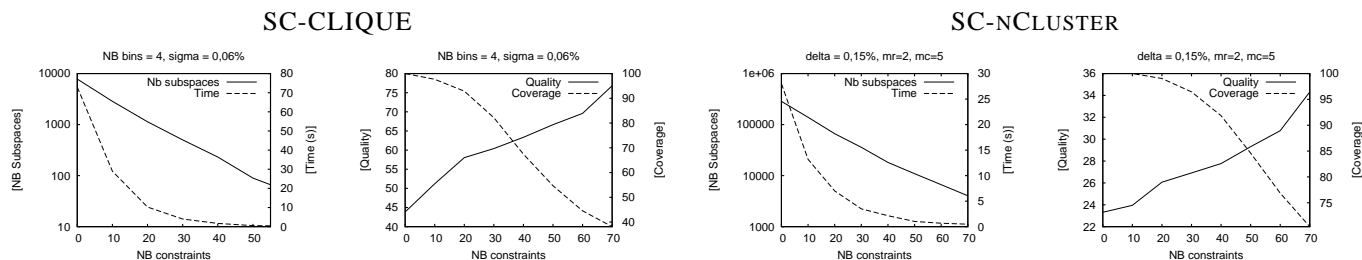


Figure 12: SC-CLIQUE and SC-NCLUSTER over a real high-dimensional gene expression data.

Functional Group	Ring	Trophozoite	Schizont	Schizont+beginning
cytoplasmic translation	<b>15,000</b>	10,500	9,375	13,045
transcription machinery	<b>4,143</b>	<b>3,500</b>	1,875	2,331
proteasome	2,286	<b>3,500</b>	2,0	2,981
ribonucleotide synthesis	1,143	<b>1,5</b>	0,625	1,513
actin myosin motility	0,143	<b>1,000</b>	<b>0,875</b>	<b>1,195</b>
mitochondrial	0,143	<b>1,5</b>	<b>0,625</b>	<b>1,968</b>
deoxynucleotide synthesis	0,000	0,000	<b>1,250</b>	0,000
dna replication	2,143	2,000	<b>5,00</b>	<b>4,558</b>
plastid genome	1,286	1,0	<b>1,75</b>	0,481
glycolytic pathway	0,000	0,500	2,000	1,955
merozoite invasion	10,714	5,000	7,625	3,792

Table 3: Average number of genes of a given functional group in the selected subspace clusters. The subspace clusters are identified by the developmental phase they concern.

## 6 Conclusion

In this paper, we proposed to extend the common framework of bottom-up subspace clustering techniques to also consider instance-level constraints during the enumeration process. This type of constraints enables the end-user to guide the unsupervised subspace clustering task by adding some expert knowledge. The new framework was applied to both density and distance-based bottom-up subspace clustering techniques. Experimental results provided evidence that the proposed framework can increase not only the efficiency of the techniques but also the quality of the resulting subspace clustering.

We believe that our proposed exhaustive constraint-based subspace clustering algorithm can be successfully integrated in an inductive database framework [8] as it allows to give an exact answer to a large range of user-defined queries when dealing with high-dimensional data. To give even more flexibility to the user, all the constraints mentioned in the introduction of this paper should be implemented as parameters to our constraint-based subspace clustering algorithm. Furthermore, the constraint generation used in this article makes our algorithm sensitive to possible noise in the data. Constraints directly given by a user will reduce this sensitivity. In the presence of noise, the use of soft constraints [4] could also improve our performance.

## 7 Acknowledgements

During this work, Elisa Fromont was working at the Katholieke Universteit Leuven. This work has been partially supported by the BINGO2 project (ANR-07-MDCO 014-02), the GOA project 2003/8, “Inductive Knowledge bases” and the FWO project “Foundations for inductive databases”.

## References

- [1] C. C. Aggarwal, A. Hinneburg, and D.A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *Proc. ICDT*, 420–434, 2001.
- [2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. ACM SIGMOD*, pages 94–105, 1998.
- [3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. VLDB*, pages 487–499, 1994.
- [4] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD*, pages 59–68, 2004.
- [5] I. Assent, R. Krieger, E. Muller, and T. Seidl. Dusc: Dimensionality unbiased subspace clustering. In *Proc. IEEE ICDM*, pages 409–414, 2007.
- [6] J. Besson, C. Robardet, J-F. Boulicaut, and S. Rome. Constraint-based concept mining and its application to microarray data analysis. *Intell. Data Anal*, 9(1):59–82, 2005.

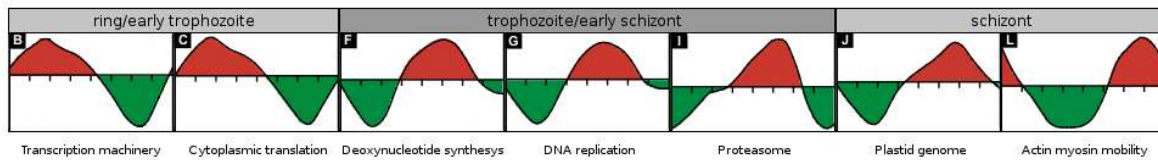


Figure 13: The temporal ordering of functional groups of genes. Each graph corresponds to the average expression profile for the genes in each functional group. The graphs are assigned to the developmental phase in which the corresponding functional group achieves its highest expression value (adapted from [11]).

- [7] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proc. ICML*, page 11, 2004.
- [8] H. Blockeel, T. Calders, E. Fromont, B. Goethals, and A. Prado. Mining views: Database views for data mining. In *Proc. IEEE ICDE*, pages 1608–1611, 2008.
- [9] C. Böhm and C. Plant. Hissclu: a hierarchical density-based method for semi-supervised clustering. In *Proc. EDBT*, pages 440–451, 2008.
- [10] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Adaptive constraint pushing in frequent pattern mining. In *Proc. PKDD*, pages 47–58, 2003.
- [11] Z. Bozdech, M. Llinás, B. Lee Pulliam, E.D. Wong, J. Zhu, and J.L. DeRisi. The transcriptome of the intraerythrocytic developmental cycle of plasmodium falciparum. *PLoS Biology*, 1(1):1–16, 2003.
- [12] C-H. Cheng, A. W.Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proc. KDD*, pages 84–93, 1999.
- [13] I. Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proc. SDM*, 2005.
- [14] A. Gély. A generic algorithm for generating closed sets of a binary relation. In *Proc. ICFCA*, pages 223–234, 2005.
- [15] J. Han and M. Kamber. *Data Mining: Concepts and Techniques, 2nd ed.* Morgan Kaufmann Publishers Inc., 2006.
- [16] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. ACM SIGMOD*, pages 1–12, 2000.
- [17] B. Jedy and J-F Boulicaut. Optimization of association rule mining queries. *Intell. Data Anal.*, 6(4):341–357, 2002.
- [18] L. Jing, M. K. Ng, J. Xu, and J. Z. Huang. Subspace clustering of text documents with feature weighting -means algorithm. In *Proc. PAKDD*, pages 802–812, 2005.
- [19] K. Kailing, H. P. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *Proc. SDM*, pages 246–257, 2004.
- [20] E. Keogh, L. Wei, X. Xi, S-H Lee, and M. Vlachos. Lb\_keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *Proc. VLDB*, pages 882–893, 2006.
- [21] D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proc. ICML*, pages 307–314, 2002.
- [22] G. Liu, J. Li, K. Sim, and L. Wong. Distance based subspace clustering with flexible dimension partitioning. In *Proc. IEEE ICDE*, pages 1250–1254, 2007.
- [23] H. Nagesh, S. Goil, and A. Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets. Tech. Rep. CPDC-TR-9906-010, Northwestern University, 1999.
- [24] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.
- [25] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explor. Newsl.*, 6(1):90–105, 2004.
- [26] C. Ruiz, M. Spiliopoulou, and E. M. Ruiz. C-dbscan: Density-based clustering with constraints. In *RSFDGrC*, volume 4482, pages 216–223, 2007.
- [27] K. Sequeira and M. J. Zaki. Schism: A new approach for interesting subspace mining. In *Proc. IEEE ICDM*, pages 186–193, 2004.
- [28] J. Struyf and S. Dzeroski. Clustering trees with instance level constraints. In *ECML*, pages 359–370, 2007.
- [29] T. Uno, M. Kiyomi, and H. Arimura. Lcm ver.3 In *Proc. ACM OSDM workshop*, pages 77–86, 2005.
- [30] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proc. ICML*, pages 577–584, 2001.
- [31] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.