

# Finding Subgroups having Several Descriptions: Algorithms for Redescription Mining

Arianna Gallo\*

Pauli Miettinen†

Heikki Mannila‡

## Abstract

Given a 0-1 dataset, we consider the redescription mining task introduced by Ramakrishnan, Parida, and Zaki. The problem is to find subsets of the rows that can be (approximately) defined by at least two different Boolean formulae on the attributes. That is, we search for pairs  $(\alpha, \beta)$  of Boolean formulae such that the implications  $\alpha \rightarrow \beta$  and  $\beta \rightarrow \alpha$  both hold with high accuracy. We require that the two descriptions  $\alpha$  and  $\beta$  are syntactically sufficiently different. Such pairs of descriptions indicate that the subset has different definitions, a fact that gives useful information about the data. We give simple algorithms for this task, and evaluate their performance. The methods are based on pruning the search space of all possible pairs of formulae by different accuracy criteria. The significance of the findings is tested by using randomization methods. Experimental results on simulated and real data show that the methods work well: on simulated data they find the planted subsets, and on real data they produce small and understandable results.

## 1 Introduction

Discovering interesting subgroups is one of the key concepts in data mining. Subgroups can be discovered by clustering, where the dataset is typically partitioned into disjoint sets, or by pattern discovery, where each pattern defines the subgroup in which the pattern is true, and thus the patterns can be overlapping. The interestingness of a subgroup is typically measured by how likely or unlikely such a subgroup would be to arise at random.

In general, a subgroup is defined either by the explicit values of the variables of the data (subgroups defined by patterns) or by distances between rows (clusters). Here we consider only subgroups defined by using formulae on the values of the variables. Any predicate  $\alpha(t)$  defined for the rows  $t$  of the dataset  $D$  defines a subgroup  $\{t \in D \mid \alpha(t) \text{ is true}\}$ . For example, if the data is 0-1, then the formula  $A = 1 \wedge (B = 0 \vee C = 1)$  defines a subgroup.

In this paper we consider the redescription mining task introduced by Ramakrishnan, Parida, and Zaki [16, 22, 15] that is the task of finding subgroups having several descriptions. That is, we want to find formulae  $\alpha$  and  $\beta$  such that the sets  $\{t \in D \mid \alpha(t) \text{ is true}\}$  and  $\{t \in D \mid \beta(t) \text{ is true}\}$  are about the same. If  $\alpha$  and  $\beta$  are logically equivalent, this holds trivially, but we are interested in finding formulae that are not equivalent, but still happen to be satisfied by about the same rows of  $D$ . (One might call such formulae  $D$ -equivalent.)

Another way of looking at the task is that we search for formulae  $\alpha$  and  $\beta$  such that the rules  $\alpha \rightarrow \beta$  and  $\beta \rightarrow \alpha$  both hold with high accuracy.

Consider the following two transactional datasets, given here in matrix form, where 1 means the presence and 0 the absence of an item:

tid	A	B	C	D	E
1	1	0	1	0	0
2	0	0	1	0	0
3	1	1	0	0	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	1	0

tid	F	G	H	I
1	0	0	1	0
2	1	1	0	1
3	0	1	1	0
4	1	1	0	0
5	1	0	1	0
6	1	0	0	1

The datasets have the same transaction (row) identifiers but different sets of attributes. They might arise from two different studies on the same entities.

Note that the subgroup  $\{1, 3, 4\}$  is defined by the formula  $A = 1 \wedge (B = 0 \vee E = 1)$  in the first table. The formula  $F = 0 \vee (G = 1 \wedge I = 0)$ , using the attributes of the second table, is true for exactly the same set  $\{1, 3, 4\}$  of rows. We say that the subset  $\{1, 3, 4\}$  has *several descriptions* and that the formulae are *redescriptions* of each other.

As another example, consider the following table:

\*University of Torino, Italy. Part of this work was done when the author was with Helsinki Institute for Information Technology. Email: gallo@di.unito.it

†Helsinki Institute for Information Technology, University of Helsinki, Finland. Email: Pauli.Miettinen@cs.helsinki.fi

‡Helsinki Institute for Information Technology, Helsinki University of Technology and University of Helsinki, Finland. Email: Heikki.Mannila@cs.helsinki.fi

tid	A	B	C	D	E	F
1	1	0	1	1	0	0
2	0	1	1	0	1	0
3	0	1	0	1	1	1
4	1	0	0	0	0	1
5	1	1	0	0	1	1
6	0	1	0	1	0	0
7	1	1	1	1	1	1
8	1	0	1	0	0	0
9	1	1	1	1	1	0

Here, the subset  $\{1, 4, 5, 8\}$  of rows can, for example, be defined in three different ways:

$$\begin{aligned}
 A &= 1 \wedge (B = 0 \vee D = 0) \\
 B &= 0 \vee (A = 1 \wedge C = 0) \quad \text{and} \\
 (C = 1 \wedge E = 0) \vee (D = 0 \wedge F = 1).
 \end{aligned}$$

Why search for groups with several descriptions? The fact that in  $D$  the set of rows satisfying  $\alpha$  and  $\beta$  are about the same tells us something interesting about the data, provided  $\alpha$  and  $\beta$  are not logically equivalent. If, for example, the set of persons whose genetic marker data satisfies a Boolean query for certain markers happens to coincide more or less exactly with the persons whose marker data satisfies another query for a completely different markers, then there is some indication that the regions of the genome might somehow be functionally connected.

In this paper we present algorithms for finding subgroups with several (typically two) descriptions. That is, we give methods that will produce pairs of formulae that are almost equivalent on the given dataset  $D$ . The methods are based on a search in the space of possible formulae; different pruning strategies are used to avoid useless paths in that space. Our goal is to produce a *small* set of pairs of formulae, where both formulae in each pair define about the same subset of rows. We define several notions of strength and interestingness for the pairs of formulae, and show how these measures can be used in pruning the search space.

Our algorithms are based on heuristic methods and our emphasis is on finding approximate redescription of arbitrary Boolean formulae. Thus our algorithms work on a different basis than many previously-proposed algorithms for redescription mining (e.g. [22, 15]), that usually concentrate on finding exact redescription of special type (e.g., monotone DNFs or only conjunctions).

If we allow arbitrarily complex formulae, then any dataset will have lots of subgroups with multiple descriptions. Thus we have to control for spurious results arising just by chance. We do this by using randomization methods, in particular swap randomization [8]. We generate a large set of randomized versions of the

input data, and run our algorithms on them. Only if the results on the real data are stronger than on, say, 99% of the randomized datasets we report a subgroup.

The rest of this paper is organized as follows. We introduce the notation and other preliminaries and define the problems in Section 2. Section 3 explains two algorithms and results from experiments with these algorithms are reported in Section 4. Some of the related work is covered in Section 5 and Section 6 contains concluding remarks.

## 2 The Problems

This section contains the notation used, describes the terminology used in redescription mining, and gives our definitions for the redescription mining problems. Our definitions differ slightly from the previous definitions [16, 22, 15], most notably by introducing thresholds for support and  $p$ -value of a redescription.

**2.1 Notation.** Given a set  $U$  of items, a transaction  $t$  is a pair  $(\mathbf{tid}, X)$ , where  $\mathbf{tid}$  is a transaction identifier and  $X \subseteq U$  is the set of items of the transaction. A dataset  $D$  is a set of transactions:  $D = \{t_1, \dots, t_n\}$ . Multiple datasets are distinguished via a subscript, i.e.,  $D_1$  and  $D_2$  are different datasets.

We consider multiple datasets  $D_1, D_2, \dots$  over itemsets  $U_1, U_2, \dots$ . We assume that the datasets have an equal number of transactions and that the sets of their transaction ids are identical. We use  $n$  to denote the number of transactions in each dataset. The itemsets  $U_i$  and  $U_j$ , on the other hand, are supposed to be disjoint for all  $i \neq j$ .

We identify items as variables in Boolean queries. We let  $\Gamma(U)$  denote the set of all possible Boolean formulae over the variables corresponding to items in  $U$ . Such formulae are called *descriptions*. Set  $\Gamma_k(U) \subseteq \Gamma(U)$  contains all queries that have at most  $k$  variables. If dataset  $D$  is over itemset  $U$ , then by  $\Gamma(D)$  we mean  $\Gamma(U)$ . Given a query  $\alpha$ , we denote by  $I(\alpha) \subseteq U$  the set of items appearing in  $\alpha$ .

A set  $I$  of items can also be viewed as a truth assignment assigning the value true to the items in  $I$  and false to all other items. We say that  $\alpha(I)$  is true if  $\alpha$  is satisfied by a truth assignment corresponding to  $I$ . If  $t_j = (\mathbf{tid}_j, I_j)$ , we say that  $\alpha(t_j)$  is true if  $\alpha(I_j)$  is true. We let  $\alpha(D)$  to be the set of transaction ids from  $D$  for those transactions which satisfy  $\alpha$ , that is,  $\alpha(D) = \{\mathbf{tid}_j : (\mathbf{tid}_j, I_j) \in D \text{ and } \alpha(I_j) \text{ is true}\}$ . The frequency  $freq(\alpha)$  is defined as  $|\alpha(D_i)|$ . Furthermore, if  $\alpha_i \in \Gamma(D_i)$ , then by  $freq(\alpha_1, \alpha_2, \dots, \alpha_n)$  we mean  $|\bigcap_{i=1}^n \alpha_i(D_i)|$ , i.e., the size of the set of all transaction ids such that those transactions satisfy each of the formulae  $\alpha_i$ .

**2.2 Jaccard similarity and redescrptions.** If  $X$  and  $Y$  are two sets, the Jaccard similarity [7] between them is defined to be

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}.$$

It is a commonly used measure of the similarity of  $X$  and  $Y$ : if  $X = Y$ ,  $J(X, Y) = 1$  and if  $X \cap Y = \emptyset$ ,  $J(X, Y) = 0$ .

If  $\alpha \in \Gamma(D_1)$  and  $\beta \in \Gamma(D_2)$  are Boolean formulae over the items in two datasets, the Jaccard similarity for the queries is defined as

$$J(\alpha, \beta) = J(\alpha(D_1), \beta(D_2)).$$

As the transaction ids of  $D_1$  are identical to those of  $D_2$ , the value  $J(\alpha(D_1), \beta(D_2))$  lies in the interval between  $[0, 1]$ . If  $J(\alpha(D_1), \beta(D_2)) = 1$ ,  $\alpha$  and  $\beta$  are called *exact redescrptions* of each other, and if  $J(\alpha(D_1), \beta(D_2)) \geq J_{\min}$  for some  $J_{\min} \in (0, 1)$ , they are called *approximate redescrptions* (with respect to  $J_{\min}$ ). In the following, we refer to tuples of Boolean formula simply as redescrptions.

For multiple databases  $D_i$  and queries  $\alpha_i \in \Gamma(D_i)$  for  $i = 1, \dots, m$ , we define

$$\begin{aligned} J(\alpha_1, \dots, \alpha_m) &= J(\alpha_1(D_1), \dots, \alpha_m(D_m)) \\ &= \frac{|\bigcap_{i=1}^m \alpha_i(D_i)|}{|\bigcup_{i=1}^m \alpha_i(D_i)|}. \end{aligned}$$

**2.3 The  $p$ -values.** Two queries  $\alpha$  and  $\beta$  can be true for exactly the same set of transactions, but the pair  $(\alpha, \beta)$  can still be uninteresting for our purposes. The first case in which this happens is when  $\alpha$  and  $\beta$  are logically equivalent; we avoid this possibility by requiring that the set of variables (items) used in  $\alpha$  and  $\beta$  are disjoint.

The second case is when the equality of the queries is a straightforward consequence of the marginal probabilities of the items. Suppose for example that two items  $A$  and  $B$  are almost always 0. Then the queries  $A = 0$  and  $B = 0$  have high support, their Jaccard similarity is large, but there is no new information in the pair.

To avoid this type of results we prune our result set using  $p$ -values. For calculating the  $p$ -value we consider a *null model* for the database, representing the “uninteresting” situation in which the items occur independently from each other in the database. The  $p$ -value of an itemset  $I$  represents the surprisingness of  $I$  under this null model. The smaller the  $p$ -value, the more “significant” the itemset is considered to be. More details on the different ways for computing the  $p$ -value of a given pattern are given in [4] and [5]; we summarize

the approach here briefly. See [19, 20, 10] for other approaches.

Let  $p_i$  be the marginal probability of an item  $i$ , i.e., the fraction of rows in the database that contain item  $i$ . Then the probability that an itemset  $I$  is contained in a transaction is  $p_I = \prod_{i \in I} p_i$ . The probability that an itemset  $I$  occurs in the dataset at least  $freq(I)$  times is then obtained by using the binomial distribution:

$$(2.1) \quad p\text{-value}(I) = \sum_{s=freq(I)}^n \binom{n}{s} p_I^s (1 - p_I)^{n-s}.$$

Equation (2.1) encodes the situation where the items occurring in a transaction are independent from each other, and gives the “strength” of the surprisingness of an itemset  $I$  under this assumption.

Note that we could consider also other  $p$ -values. For instance, another  $p$ -value could measure the significance of a set of transactions containing a certain item, this time with the assumption that the transactions contain that item independently of each other. The  $p$ -value of an itemset can now be computed as the maximum of these two  $p$ -values [4].

As we already pointed out, an itemset  $I$  can be interpreted as a truth assignment assigning the value true to all the items in  $I$  and false to all other items. For instance, if the itemset  $I$  is defined as the set of items  $\{A, B, C\}$ , the rule  $\alpha(I)$  corresponds to  $A = 1 \wedge B = 1 \wedge C = 1$ . Such formula is satisfied in a given transaction only if all the three literals are present in the transaction. For a certain conjunctive query  $\alpha$ , the probability  $p_\alpha$  that  $\alpha$  is true for a transaction is computed under the assumption of independence of variables occurring in  $\alpha$ . The probability can be defined recursively:

$$p_\alpha = \begin{cases} p_\beta p_\gamma & \text{if } \alpha = \beta \wedge \gamma \\ 1 - p_\beta & \text{if } \alpha = \neg \beta \\ p_\beta + p_\gamma - p_\beta p_\gamma & \text{if } \alpha = \beta \vee \gamma. \end{cases}$$

We can now generalize (2.1) for formulae:

$$(2.2) \quad p\text{-value}(\alpha) = \sum_{s=freq(\alpha)}^n \binom{n}{s} p_\alpha^s (1 - p_\alpha)^{n-s}.$$

**2.4 Problem definitions.** In this subsection we give the definition of redescription mining that is used throughout this paper. Our definition has minor differences to previous definitions [16, 22, 15].

We look for redescrptions that are true for a reasonable large set of transactions, whose Jaccard

similarity is high enough, and the  $p$ -value is small enough. We will also use a threshold for maximum frequency; the purpose of this threshold partly coincides with that of  $p$ -value, as it is used to prune results that are not significant because they cover most of the data. Apart the Jaccard similarity, these thresholds were not presented in previous definitions of redescription mining [16, 22, 15]. Thus, our definitions can be seen as generalizations of those definitions.

The first problem version is formulated for the case where there are several databases with the same set of transactions, and the second version is for the case of a single database. In the single database case, we search for queries that do not share items.

**PROBLEM 2.1.** *Given  $m \geq 2$  datasets  $D_1, D_2, \dots, D_m$ , each containing the same set of transaction ids and disjoint set of items, and thresholds  $\sigma_{min}, \sigma_{max}, J_{min}$ , and  $P_{max}$  for minimum and maximum support, Jaccard similarity, and  $p$ -value, respectively. Find a set of  $m$ -tuples of Boolean formulae,  $(\alpha_1, \alpha_2, \dots, \alpha_m) \in \Gamma(D_1) \times \Gamma(D_2) \times \dots \times \Gamma(D_m)$ , so that*

$$(2.3) \quad freq(\alpha_1, \dots, \alpha_m)/n \in [\sigma_{min}, \sigma_{max}]$$

$$(2.4) \quad J(\alpha_1, \dots, \alpha_m) \geq J_{min}$$

$$(2.5) \quad p\text{-value}(\alpha_1, \dots, \alpha_m) \leq P_{max}.$$

A slightly different version of this problem arises when we are given only one dataset. In that case we search for Boolean formulae satisfied by approximately the same set of transactions, but being defined by using disjoint sets of items.

**PROBLEM 2.2.** *We are given a dataset  $D$ , an integer  $m \geq 2$ , and the thresholds as in Problem 2.1. The task is to find a set of  $m$ -tuples of Boolean formulae,  $(\alpha_1, \dots, \alpha_m), \alpha_i \in \Gamma(D)$ , so that the conditions (2.3)–(2.5) hold and that for  $i \neq j$*

$$(2.6) \quad \alpha_i(D) \cap \alpha_j(D) = \emptyset.$$

Notice that the condition (2.6) is implied in Problem 2.1 by the assumption of mutual disjointness of  $U_i$ 's.

### 3 Algorithms

In this section we present two algorithms for the redescription mining, namely the GREEDY algorithm and the MID algorithm. The algorithms are heuristic search methods that try to form useful pairs or tuples of queries.

**3.1 The Greedy Algorithm.** The GREEDY algorithm performs a level-wise search using a greedy heuristic to prune the search space. In addition to the threshold values, it also uses two other parameters to control

**ALGORITHM 3.1** The GREEDY algorithm for Problem 2.1.

---

**Input:** Datasets  $D_1, \dots, D_m$ , thresholds  $\sigma_{min}, \sigma_{max}, J_{min}$ , and  $P_{max}$  and a maximum number of formula tuples  $N$  and a maximum number of variables per query  $K$ .  
**Output:** A set  $\mathcal{F}$  of  $m$ -tuples of Boolean queries with at most  $K$  variables.

```

1:  $\mathcal{F} \leftarrow \{(\alpha_1, \dots, \alpha_m) \in \Gamma_1(D_1) \times \dots \times \Gamma_1(D_m) : (\alpha_1, \dots, \alpha_m) \text{ satisfies (2.3)–(2.5)}\}$ 
2: Order  $\mathcal{F}$  using  $J(\alpha_1, \dots, \alpha_m)$  and keep top  $N$  redescrptions.
3: for  $(\alpha_1, \dots, \alpha_m) \in \mathcal{F}$  do
4:    $V_i \leftarrow U_i \setminus I(\alpha_i), i = 1, \dots, m$ 
5:   for  $i \leftarrow 1, \dots, m$  do
6:      $W_i \leftarrow V_i$ 
7:     for  $i \in V_i$  do
8:       Try to expand  $\alpha_i$  to  $\alpha'_i$  using  $i$  with  $\wedge, \vee$ , and  $\neg$ 
9:       if  $J(\alpha_1, \dots, \alpha'_i, \dots, \alpha_m) \geq J(\alpha_1, \dots, \alpha_i, \dots, \alpha_m)$  and  $freq(\alpha_1, \dots, \alpha'_i, \dots, \alpha_m)/n \in [\sigma_{min}, \sigma_{max}]$  then
10:         $W_i \leftarrow W_i \setminus \{i\}$  and save  $\alpha'_i$ 
11:      Select the  $\alpha'_i$  with highest  $J(\alpha_1, \dots, \alpha'_i, \dots, \alpha_m)$ 
12:       $\alpha_i \leftarrow \alpha'_i$  and  $V_i \leftarrow V_i \setminus (W_i \cup I(\alpha'_i))$ 
13:      if  $\max\{I(\alpha_i)\} < K$  and there is  $V_i \neq \emptyset$  then
14:        goto line 5
15: return  $\mathcal{F}$ 

```

---

the result, namely the maximum number of initial redescrptions to process and the maximum number of variables per Boolean formula. The pseudo-code of the algorithm is given as Algorithm 3.1.

The GREEDY algorithm works as follows. First (in lines 1–2) it finds the initial redescrptions, that is, tuples of formulae with each formula having only one variable. This is done via an exhaustive search. The redescrptions are ordered according to their Jaccard similarity, and only the best ones are considered in the remaining of the algorithm.

In the next step, the algorithm considers each initial redescription separately. In each redescription, it starts by considering formula  $\alpha_1$  which it tries to expand using the items from set  $V_1$  (line 8). So if  $\alpha_1 = (A = 0)$ , the algorithm creates formulae  $(A = 0 \vee B = 1)$ ,  $(A = 0 \wedge B = 1)$ ,  $(A = 0 \vee B = 0)$ , and  $(A = 0 \wedge B = 0)$ . If any of the created formulae does not decrease the Jaccard similarity of the redescription and keeps the support within the limits, then the item  $i$  is removed from the set  $W_1$  of “useless” items and the formula is saved with the new Jaccard similarity. When all items and formulae are tried, the algorithm selects the one that improves the Jaccard similarity most and replaces the old formula with that. It also removes from the set  $V_1$  of available items the item added to  $\alpha_1$  and the items from set  $W_1$ , i.e., the items that only decreased the Jaccard similarity (lines 11–12). Algorithm then proceeds to formula  $\alpha_2$ , and so on.

When all formulae are considered, algorithm checks if the maximum number of variables per formula is reached. If not, and if there still are items that can be added to the formulae, algorithm starts again from formula  $\alpha_1$ . Otherwise algorithm returns all redescrptions

that are in  $\mathcal{F}$ . If wanted, the redescrptions in  $\mathcal{F}$  can once more be pruned according to the  $P_{\max}$  threshold.

The way the GREEDY algorithm prunes its search space can lead to sub-optimal results in many ways. First, of course, the algorithm can prune initial pairs that could be used to create good descriptions. Second, the algorithm prunes the level-wise search tree in a way that can yield to pruning of items that could be used later. The pruning is similar to that of many frequent itemset mining algorithm's (e.g., APRIORI) but, unlike in frequent itemset mining, the anti-monotonicity property does not hold here because of the disjunctions.

The time complexity of the algorithm depends heavily on the efficiency of the pruning phase. If  $K$ , the maximum number of variables per formula, is not limited, then the algorithm can take exponential running time with respect to the maximum number of items in datasets. Another variable that has exponential effect to the running time of GREEDY is the number of datasets,  $m$ . Assuming all datasets have equal number  $k$  of items, just creating the initial pairs takes  $O(2^m n^m)$  time. However, in practice  $m$  is usually small and does not cause serious slowdown of the algorithm.

If  $m$  is very small, say  $m = 2$ , a simple improvement is applicable to the GREEDY algorithm: instead of only iterating over formulae  $\alpha_i$  in fixed order, try all permutations of the ordering and select the best. We employ this improvement in our experiments, where  $m$  is always 2. With larger values of  $m$  the number of permutations, of course, becomes quickly infeasible.

**Modifications for Single Dataset.** We can modify the GREEDY algorithm to work with the setting of Problem 2.2, i.e., when only one dataset  $D$  is given. First, the initial  $m$ -tuples are computed over the  $2^m \binom{U}{m}$  possible Boolean formulae in  $\Gamma_1(D)$ . The sets  $V_i$  and  $W_i$  in Algorithm 3.1 are replaced with two sets,  $V$  and  $W$ , which are used with all formulae. In that way the algorithm ensures that the formula tuples satisfy the disjointness condition (2.6). Also in this case, if  $m$  is small, we can, and in the experiments will, apply the aforementioned improvement of iterating over all permutations of the ordering of the formulae.

**3.2 The MID Algorithm.** The GREEDY algorithm starts from singleton queries, i.e., queries with only one variable, and iteratively adds new variables using Boolean operators.

The approach described in this section starts instead from a set of itemsets, i.e., a set of positive conjunctive queries, and iteratively adds new itemsets using Boolean operators. At each step, the  $p$ -value of the query is used for pruning non-interesting queries. Since in real-world tasks the number of all the possible item-

sets is often massive, we only extract a condensed and lossless representation of them, i.e., a set of closed itemsets [14]. A closed itemset is an itemset with no frequent superset having same support. We can focus only on closed itemsets since for any non-closed itemset there exists a closed itemset with lower  $p$ -value [4].

In this section we describe MID, an algorithm for Mining Interesting Descriptors of subgroups. The sketch of the MID algorithm is given in Algorithm 3.2. The algorithm starts by mining the set of frequent closed itemsets from the given dataset(s) and sorting them by  $p$ -value (line 1). In this step we compute the  $p$ -value of an itemset  $I$  as the maximum of the two  $p$ -values computed under the assumptions of independence of items and transactions, as explained in Section 2.3.

The set  $R_i^0$  contains the closed itemsets mined from the dataset  $D_i$ . For each query  $\alpha \in R_i^0$ , the query  $(\neg\alpha)$  is added to  $R_i^0$  if  $p\text{-value}(\neg\alpha) < P_{\max}$  (lines 2–4). The  $p$ -values in line 3 are computed as in the previous step. Only the first  $N$  queries constitute the input for the next steps (line 5). At each step  $j = 1, \dots, K-1$ , (line 6) new queries are added to the set  $R_i^{j+1}$  of “surprising” queries using Boolean operators  $\wedge$  and  $\vee$ . Such queries have  $p$ -value higher than the given threshold and are built by using  $j+1$  closed itemsets. That is, at each step  $j$ , the set  $R_i^{j+1}$  is built in a level-wise fashion, starting from the set  $R_i^j$  (lines 6–13). In these steps, when a positive disjunctive query is extended by adding to its formula another itemset, the sharing of any items between the two itemsets is avoided. For this reason, the  $p$ -values in lines 9 and 11 are computed without taking into account the second  $p$ -value described in Section 2.3, i.e., the one that considers the transactions to be independent, but only that computed under the assumption that items are independent.

In the last step (lines 15–22), the subgroups having redescrptions are mined. If only one dataset is given as input, i.e.  $m = 1$ , the descriptions are computed using the queries  $(\alpha_1, \alpha_2)$  found in the previous steps (lines 20–22). Instead,  $m$  descriptions,  $\alpha_1, \alpha_2, \dots, \alpha_{K-1}$ , will be sought in lines 16–18 taking into account the set of queries  $R_i^{K-1}$  found in the previous steps from each dataset  $D_i$ , respectively.

The final result  $R$  is sorted by Jaccard similarity. However, this measure could suggest misleading results. Indeed, a set of queries with high Jaccard similarity is not necessarily an “interesting” result, as mentioned in Section 2.3. Thus, the MID algorithm uses both the Jaccard similarity and  $p$ -value to find accurate and interesting redescrptions.

Let us note that the result of the MID algorithm is slightly different to that of GREEDY. While MID tries to find the descriptions starting from itemsets, the

---

**ALGORITHM 3.2** The MID Algorithm for Problem 2.1 and Problem 2.2

---

**Input:** Datasets  $D_1, \dots, D_m$ , thresholds  $\sigma_{\min}$ ,  $J_{\min}$ , and  $P_{\max}$  and a maximum number of formula tuples  $N$  and a maximum number of itemsets per query  $K$

**Output:** A set  $\mathcal{F}$  of  $m$ -tuples of Boolean queries with at most  $K$  itemsets.

```

1: for  $D_i \in \mathcal{D}$  do  $R_i^0 \leftarrow$  set of closed itemsets mined from  $D_i$  with
   support threshold  $\sigma_{\min}$  and sorted by  $p$ -value  $< P_{\max}$ 
2:   for  $\alpha \in R_i^0$  do
3:     if  $p$ -value( $\neg\alpha$ )  $< P_{\max}$  then
4:        $R_i^0 \leftarrow R_i^0 \cup (\neg\alpha)$ 
5:    $R_i^1 \leftarrow$  first  $N$  queries of  $R_i^0$ 
6:   for  $j = 1, \dots, (K - 1)$  do
7:      $R_i^{j+1} \leftarrow R_i^j$ 
8:     for  $\alpha \in R_i^j$  and  $\beta \in R_i^j$  do
9:       if  $p$ -value( $\alpha \wedge \beta$ )  $< P_{\max}$  then
10:         $R_i^{j+1} \leftarrow R_i^{j+1} \cup (\alpha \wedge \beta)$ 
11:       if  $p$ -value( $\alpha \vee \beta$ )  $< P_{\max}$  then
12:         $R_i^{j+1} \leftarrow R_i^{j+1} \cup (\alpha \vee \beta)$ 
13:      $R_i^{j+1} \leftarrow$  first  $N$  queries of  $R_i^{j+1}$ 
14:  $R \leftarrow \emptyset$ 
15: if  $m > 1$  then
16:   for  $\alpha_1 \in R_1^{K-1}, \alpha_2 \in R_2^{K-1}, \dots, \alpha_m \in R_m^{K-1}$  do
17:     if  $J(\alpha_1, \alpha_2, \dots, \alpha_m) > J_{\min}$  then
18:        $R \leftarrow R \cup (\alpha_1, \alpha_2, \dots, \alpha_m)$ 
19: else
20:   for  $\alpha_1 \in R_1^{K-1}, \alpha_2 \in R_2^{K-1}$  do
21:     if  $J(\alpha_1, \alpha_2) > J_{\min}$  then
22:        $R \leftarrow R \cup (\alpha_1, \alpha_2)$ 

```

---

GREEDY algorithm considers single items. Therefore, the descriptions returned by the GREEDY algorithm will have a maximum number  $K$  of items, while those resulting from the MID algorithm contain a maximum number  $K$  of itemsets.

**3.3 Redundancy Reduction Methods.** The re-descriptions (formula tuples) returned by the algorithms can still be very redundant, e.g., the formulae in different descriptions can be almost the same. For a simple example of redundant results, consider two pairs of formulae,  $(\alpha_1 = (A = 1 \wedge B = 1 \vee C = 0), \alpha_2 = (D = 1))$  and  $(\beta_1 = (A = 1 \wedge B = 1 \vee D = 1), \beta_2 = (C = 0))$ . We say that  $(\beta_1, \beta_2)$  is redundant with  $(\alpha_1, \alpha_2)$ , if it contains (almost) the same set of variables and is satisfied in (almost) the same set of transactions as  $(\alpha_1, \alpha_2)$ . To remove the redundancy, we use a simple postprocessing method similar to that used in [4].

The algorithm is very straightforward. It takes as an input the re-descriptions from GREEDY or MID, the datasets  $D_i$ , and the thresholds. It sorts the re-descriptions according to their Jaccard similarity, and starts from the one with highest value. Then, for each formula  $\alpha_i$  and each transaction  $t \in \{t = (\text{tid}, X) : \text{tid} \in \alpha_i(D_i)\}$ , it removes the items of  $I(\alpha_i)$  from the items of  $t$ . The algorithm then starts iterating over the remaining re-descriptions. For each re-description, it first checks if it still admits the thresholds in the new data (with some items removed from transactions),

and if it does, then removes the items as with the first re-description. If a re-description fails to meet the thresholds, then it is discarded. After the algorithm has gone throughout all re-descriptions, it reports only those that were not discarded.

## 4 Experimental Evaluation

In this section we describe the experimental evaluation of the algorithms. Our results show that, despite the simplicity of the algorithms, we can find interesting and intuitively appealing results from real data. We also assess the significance of our results via swap randomization (see, e.g., [3, 8]).

**4.1 Synthetic Data.** As the first test to our algorithms, we used two synthetic datasets. The goal of the experiments was to show that (1) we can find planted descriptions from the data, and (2) we will not find too many spurious descriptions. To this end, we created two data instances, both of which consist of two small 0-1 matrices. In the first data we planted two simple descriptions describing the same set of rows; one description in both matrices. The columns not appearing in the formulae constituting the descriptions were filled with random noise. We then added some noise to the columns appearing in the formulae and tried to find the correct descriptions from the data.

To measure the quality of the answers, we ordered all re-descriptions returned by the algorithms according to their Jaccard similarity. The quality of the answer was then the position in the list where our planted re-description appeared. Thus, quality 1 means that the algorithm performed perfectly and the planted re-description was on the top of the list, and higher values mean worse results.

The formulae we planted were of the type  $(A = 1 \vee B = 0)$  and  $(C = 1 \vee D = 1)$ . As we can as well consider the complements of these formulae,  $(A = 0 \wedge B = 1)$  and  $(C = 0 \wedge D = 0)$ , we also accepted them as a correct answer. As the noise can make the full formula of the description less significant of an answer than a sub-formula of it, we considered the highest-ranked pair of (possible) sub-formulae of the planted description when computing the quality. The results can be seen in Table 1.

As we can see from Table 1, the GREEDY algorithm finds the correct answer well with low values of noise. It does, however, occasionally fail to find the description at all, as is the case with 8% of noise. With higher levels of noise the algorithm's performance deteriorates, as expected. The MID algorithm also performs well at first, but when the noise level increases, its performance drops faster than GREEDY's. Part of this worse

Table 1: The position of the planted redescription in the results ordered according to Jaccard similarity. “–” denotes that the algorithm was unable to find the redescription.

Algorithm	Noise level											
	0	0.02	0.04	0.06	0.08	0.1	0.12	0.14	0.16	0.18	0.2	
GREEDY	1	1	1	1	–	1	3	1	–	67	–	
MID	1	1	1	331	–	–	–	–	–	–	–	

Table 2: The results with completely random data and varying  $P_{\max}$ .

Algorithm	$P_{\max}$				
	1	0.5	0.25	0.125	0.0625
GREEDY	158	64	23	5	1
MID	0	0	0	0	0

Table 3: The results with completely random data and varying  $J_{\min}$ .

Algorithm	$J_{\min}$				
	0.4	0.5	0.6	0.7	0.8
GREEDY	158	4	0	0	0
MID	0	0	0	0	0

performance can be attributed to MID’s stronger use of  $p$ -values in pruning the search space: as the noise level increases, the findings start more and more look like the ones obtained by a mere change.

The other synthetic data we used consisted of two matrices with fully random data. The idea of this data is to show that from a random data one cannot find any significant descriptions. The support thresholds for this experiment were set to  $\sigma_{\min} = 20\%$  and  $\sigma_{\max} = 80\%$ . Thresholds  $P_{\max}$  and  $J_{\min}$  were varied: first,  $J_{\min}$  was fixed to 0.4, and  $P_{\max}$  varied from 1 to 0.0625, halving at every step; then  $P_{\max}$  was fixed to 1 and  $J_{\min}$  varied from 0.4 to 0.8. The goal of this experiment was not to find anything, and the MID algorithm performed perfectly in this sense: it never found any descriptions. The GREEDY algorithm did find some descriptions; its results are given in Tables 2 and 3.

Tables 2 and 3 show that while GREEDY does find descriptors with the minimum values of the thresholds, the number of descriptors it reports decreases rapidly when the thresholds are increased. The  $J_{\min}$  threshold is especially effective, as there are no descriptions reported after  $J_{\min} \geq 0.6$ .

**4.2 Real Data.** We tested the methods on real data. The first test is whether the methods find intuitively interesting results from real data. That is, do the algorithms output something that passes the pruning criteria and seems interesting for a user who understands (at least part of) the application domain.

As this test can be somewhat subjective, we also wanted to verify that the results are not just due to the presence of sufficient amounts of data about phenomena that we are familiar with. To test this, we used randomization methods to check that on the real data there are clearly more pairs of formulae passing the pruning criteria than on randomized versions of the data. For the second type of test, we used swap randomization [3, 8], a method that generates matrices having the same row and column sums as the original 0-1 data set.

**The Data.** We used three real data sets, called **Courses**, **Web**, and **DBLP**. **Courses** data contains students’ course enrollment data of CS students at University of Helsinki. The data contains a single dataset of 106 courses (items) and 2401 students (transactions). Thus it is applicable for testing our algorithms in the framework of Problem 2.2.

The **Web** data<sup>1</sup> is used in [2] and is collected from web pages of US universities’ CS departments. It contains two datasets. The first dataset is about the terms appearing in the web pages. The transactions (rows) are the web pages (1051 pages) and the items are the terms occurring in the pages (1798 terms). The other dataset contains the terms from hyperlinks pointing to the web pages: the transactions are again the pages, and the items are the terms occurring in the hyperlinks (438 terms). The **Web** data thus gives two parallel views to a single web page: the terms used in the page and the terms used to describe the page in hyperlinks. All terms were stemmed and the most frequent and infrequent terms were removed.

The **DBLP** data was collected from the DBLP Bibliography database<sup>2</sup>. It also contains two datasets. The

<sup>1</sup><http://www.cs.cmu.edu/afs/cs/project/theo-11/www/wkwb/>

<sup>2</sup><http://www.informatik.uni-trier.de/~ley/db/>

transactions (rows) in the first dataset are authors, and the items in the first dataset are CS conferences: the presence of an item in a transaction means that the author has published in that conference. The other dataset indicates the co-authorship relations: transactions are again authors and as items we also have authors. The dataset is symmetric (i.e., the matrix representation of it is symmetric). As the whole dataset is huge, we selected only 19 conferences in our sample. The conferences were WWW, SIGMOD, VLDB, ICDE, KDD, SDM, PKDD, ICDM, EDBT, PODS, SODA, FOCS, STOC, STACS, ICML, ECML, COLT, UAI, and ICDT. We then removed all authors that had published less than 5 times in these conferences, resulting in a dataset containing 2345 authors. Thus the dimensions of the two tables are  $2345 \times 19$  and  $2345 \times 2345$ . The properties of the real data are summarized in Table 4.

**The Results.** With the `Courses` data, the GREEDY algorithm reports a lot of descriptions. However, as discussed earlier, many of these results are in fact redundant, and can thus be pruned using the redundancy reduction method from Section 3.3. Another way to prune the results is to select the accuracy threshold  $J_{\min}$  to a high enough value. Our experiments with swap randomized data suggest setting it to as high as 0.9, as with lower values of  $J_{\min}$  the results are not found significant, but while there still are more than 50 redescrptions having Jaccard similarity at least 0.9, none of such redescrptions were found from randomized data, as can be seen from Table 5. After applying these two pruning criteria, we were left with 5 redescrptions. The results are given in Table 6.

Both formulae in the first pair shown in Table 6, for example, describe a set of students that have studied their first-year courses according to the old curriculum. The second pair of formulae is about the students that have studied their first-year courses according to the new curriculum. The fifth pair of formulae is also describing the similar set of students, but this time with somewhat different set of attributes. The results with MID were similar in fashion, and they are omitted here. From the randomized data MID did find many redescrptions but, as all of those redescrptions had  $p$ -value higher than the highest  $p$ -value of the redescrptions from the original data, they were pruned and thus all results by MID can be considered significant.

While the results from the GREEDY algorithm with the `Courses` data originally contained lots of redundancy, the results from the `Web` data were free of redundancy. The similarity of the results was also lower than in the results from `Courses`, hinting that the data does not contain subgroups that are easy to describe.

However, the results can be considered significant, as neither of the algorithms found any results from the randomized data (Table 7). The GREEDY algorithm gave 9 redescrptions, some of which are reported in Table 8.

Due to the nature of the data, the redescrptions in Table 8 mostly refer to courses' home pages: course pages constitute a substantial part of the data and they tend to be very homogeneous, all containing similar words. Describing, for example, people's personal home pages is much harder due to the heterogeneity of them. All of the examples in Table 8 describe some set of course home pages: the first pair, for example, is about courses held in autumn or winter and not in Seattle.

The MID algorithm reported 56 redescrptions having similarity at least 0.4. Some of the redescrptions are given in Table 9. The first two redescrptions, both having very high similarity, exploit the usual structure in web pages: universities' and people's home pages tend to mention some address. The third redescrption is similar to those reported by GREEDY, though with smaller similarity.

The third data, the DBLP data, has a different structure. A redescrption in this data describes a group of computer scientists via their publication patterns and co-authorships. As a such, it is easier to describe what a certain computer scientist is *not* than to say what she is. However, saying that those who do not publish in data mining conferences have not published with certain data miners is not as interesting as it is to say the opposite. Therefore, for this data, we applied an extra restriction to the descriptions: For the GREEDY algorithm the formulae must be monotone, i.e., they can only test if variable equals to 1, and for the MID algorithm the itemsets can be joined using only conjunctions and negations.

The GREEDY algorithm again reported very small set of redescrptions, only 8, and the similarities were quite low, varying between 0.35 to 0.25 (0.2 being the minimum allowed). The results were, again, very significant, as nothing could be found from the randomized data even with as low of a minimum similarity as 0.1 (Table 10). Some of the results by GREEDY are reported in Table 11.

The results in Table 11 show three groups of conferences and authors: machine learning (redescrption 1), theoretical computer science (redescrption 2), and data mining (redescrption 3).

The results from the MID algorithm are similar to those of GREEDY, but with somewhat more structure and less similarity. As with GREEDY all results can be considered significant, as no results with accuracy higher than 0.1 were found from the randomized data (Table 10). Some example results of MID are given in

Table 4: Properties of the real datasets.

Data	Description	Rows	Columns	Density
Courses	students $\times$ courses	2401	106	0.1223
Web	pages $\times$ terms in pages	1051	1798	0.0378
	pages $\times$ terms in links	1051	438	0.0060
DBLP	authors $\times$ conferences	2345	19	0.1938
	authors $\times$ authors	2345	2345	0.0049

Table 5: Number of redescription found from original and randomized Courses data;  $N = 200$ .

Algorithm	Data	$J_{\min}$					
		0.4	0.5	0.6	0.7	0.8	0.9
GREEDY	Orig.	200	200	200	200	200	56
	Rand.	200	200	200	200	167	0
MID	Orig.	150	119	53	19	0	0
	Rand.	0	0	0	0	0	0

Table 12. To improve the readability, we write formulae of type  $\neg(A = 1 \wedge B = 1)$  as  $A = 0 \vee B = 0$ ; thus the results contain disjunctions even if the algorithm did not use them. The redescription given in Table 12 describe a subgroup of theoretical computer scientists with the special emphasis on avoiding those with research also in the field of data mining.

## 5 Related Work

Redescription rule mining was introduced by Ramakrishnan et al. [16], and they also presented an algorithm for it. Their algorithm, CARTwheels, is based on learning decision trees and, like our algorithms, it does not put any restriction to the type of the formulae in redescription, but rather limits the number of variables in formulae. Since that, other algorithms have been proposed, most of which concentrate only on special type of Boolean formulae. For example, Zaki and Ramakrishnan [22] give an algorithm to find exact minimal conjunctive redescription while Parida and Ramakrishnan [15] give algorithms for finding exact and approximate monotone redescription in CNF or DNF. Parida and Ramakrishnan also study several theoretical properties of redescription.

Recently, Kumar et al. [11] extended redescription mining framework to the fascinating task of *storytelling*, where the goal is to find consecutive redescription that relate completely disjoint elements. Notice that this is different to our goal of finding  $m$ -tuples of formulae, as we require the mutual similarity of the formulae to be high, while in storytelling, the similarities between

descriptions next to each other is required to be high, but the similarity between first and last formulae must be zero.

Mining association rules and their variants has been, and still is, an active area of research. See [9] for a thorough survey. Redescription mining can be seen as a one generalization of association rule mining. Other generalizations include, for example, the negative association rules [17], i.e., rules of type  $A \rightarrow \neg B$ ,  $\neg A \rightarrow B$ , and  $\neg A \rightarrow \neg B$ .

An adaptation of the standard classification rule learning methods to subgroups discovery is proposed in [12]. The method proposed in [12] seeks for the best subgroups in terms of rule coverage and distributional usualness. A weighted relative accuracy heuristic is used to trade off generality and accuracy of a certain rule  $\alpha \rightarrow A$ , where  $A$  is a single target attribute. For other methods for subgroup discovery, see, e.g., [18].

Recently, Wu et al. [21] proposed an algorithm to mine positive and negative association rules. Their approach has some similarities with ours, as they define an interestingness measure and a confidence measure and use these and the frequency to prune the search space. However, excluding the frequency, their measures are different to ours. Also, they only consider itemsets and their negations, not general Boolean formulae, as we do. Negative association rules have also been applied with success for example to bioinformatics [1].

Another generalization to association rules are the disjunctive association rules that can contain disjunctions of itemsets. Nanavati et al. [13] propose an algorithm for mining them.

Both negative and disjunctive association rules still have restriction in the types of the Boolean expressions they consider, and they do not allow arbitrary Boolean expressions. Recently Zhao et al. [23] have proposed a framework to mine arbitrary Boolean expressions from binary data. While their algorithm is able to find all frequent expressions, and in theory could be used in redescription mining, its time complexity makes it difficult to apply for our purposes.

The idea of studying two parallel datasets is present also in co-training [2]. This technique has some similar-

Table 6: Results from the `Courses` dataset with GREEDY algorithm.

	Left formula	Right formula	$J$
1.	“Introduction to UNIX”=1 $\vee$ “Programming in Java”=0 $\vee$ “Introduction to Databases”=0 $\vee$ “Programming (Pascal)”=1	“Introduction to Programming”=0 $\vee$ “Introduction to Application Design”=0 $\vee$ “Programming Project”=0 $\vee$ “The Principles of ADP”=1	0.91
2.	“Database Systems I”=0 $\vee$ “Concurrent Systems (Fin)”=1 $\vee$ “Data Structures Project”=0 $\vee$ “Operating Systems I”=1	“Introduction to Application Design”=1 $\vee$ “Concurrent Systems (Eng)”=0 $\vee$ “Database Management”=1 $\vee$ “Data Structures”=1	0.91
3.	“Information Systems Project”=1 $\wedge$ “Data Structures Project”=1 $\wedge$ “Database Management”=0 $\wedge$ “Elementary Studies in Computer Science”=0	“Database Systems I”=1 $\wedge$ “Information Systems”=1 $\wedge$ “Database Application Project”=0 $\wedge$ “Operating Systems I”=0	0.90
4.	“Information Systems”=1 $\vee$ “Introduction to Databases”=0 $\vee$ “Managing Software Projects”=1 $\vee$ “Corba Architecture”=1	“Introduction to Application Design”=0 $\vee$ “Database Systems I”=1 $\vee$ “Languages for AI”=1 $\vee$ “Unix Softwareplatform”=1	0.90
5.	(“Programming in Java”=1 $\wedge$ “Programming (Pascal)”=0 $\vee$ “Introduction to Application Design”=1) $\wedge$ “Software Engineering”=0	(“Operating Systems I”=1 $\vee$ “Introduction to Programming”=1 $\vee$ “Introduction to Databases”=1) $\wedge$ “Corba Architecture”=0	0.90

Table 7: Number of redescription found from original and randomized `Web` data.

Algorithm	Data	$J_{\min}$					
		0.4	0.5	0.6	0.7	0.8	0.9
GREEDY	Orig.	52	50	13	5	0	0
	Rand.	0	0	0	0	0	0
MID	Orig.	56	56	56	56	55	0
	Rand.	0	0	0	0	0	0

ities with redescription mining. In the problem setting one is given two parallel datasets with some, but not all, transactions being labeled. The goal is to build two classifiers to classify the data. The building of these classifiers is done iteratively, using the labeling given by one of the classifier as a training data to the other, after which the roles are switched. This approach has some similarities with the CARTwheels algorithm by Ramakrishnan et al. [16] and indeed, in redescription mining, if we are given a formula  $\alpha \in \Gamma(D_1)$ , finding the formula  $\beta \in \Gamma(D_2)$  can be considered as a machine learning task.

The concept of cross-mining binary and numerical attributes was studied recently in [6]. In short, in the cross-mining problem we are given a set of transactions, each of which contains two types of attributes, Boolean and real-valued, and the goal is to segment the real-valued data so that the segments correspond to itemsets in Boolean data and segment the real-valued data well.

In a Boolean data table, a pattern is defined as a set of rows that share the same values in two or more columns. Similarly, it can be identified as a set of items occurring in the same transactions. Unfortunately, in real-world problems the number of all the possible patterns is prohibitively large. Quantifying the interestingness of patterns found is thus often mandatory in order to prune the search space. Several methods were defined in the last decade to define the “interestingness” of a pattern: conciseness, generality, reliability, peculiarity, diversity, novelty, surprisingness, utility, applicability. A survey on the most prominent interestingness measures is given in [7].

In [4] a scalable method to mine interesting and non-redundant patterns from a dataset is proposed. A new statistical measure of interestingness is devised in order to measure the “surprisingness” of a pattern under two different null models. An extension of the method for parallel datasets is given in [5].

## 6 Concluding remarks

We have considered a version of the redescription mining problem, i.e., finding Boolean formulae  $\alpha$  and  $\beta$  such that the rules  $\alpha \rightarrow \beta$  and  $\beta \rightarrow \alpha$  both hold with a reasonable accuracy. The problem is motivated by the general goal of data mining: finding unexpected viewpoints to the data.

We gave two simple heuristic algorithms for the task. The results are pruned by requiring that the subgroup is large enough, the Jaccard similarity between

Table 8: Results from the **Web** data with the GREEDY algorithm. Left formulae are over the terms in web pages and right formulae are over terms in links.

	Left formula	Right formula	$J$
1.	$(\text{autumn}=1 \vee \text{cse}=1) \wedge \text{seattl}=0 \wedge \text{intern}=0$	$(\text{cse}=1 \vee \text{autumn}=1) \wedge \text{other}=0 \vee \text{winter}=1$	0.75
2.	$(\text{grade}=1 \wedge \text{section}=1 \vee \text{wendt}=0) \wedge \text{hour}=1$	$(\text{section}=1 \vee \text{lectur}=1) \wedge \text{system}=0 \wedge \text{program}=0$	0.63
3.	$(\text{lectur}=1 \vee \text{instructor}=1) \wedge \text{fax}=0 \vee \text{exam}=1$	$\text{cs}=1 \vee \text{cse}=1 \vee \text{report}=1 \vee \text{introduc}=1$	0.55

Table 9: Results from the **Web** data with the MID algorithm. Left formulae are over the terms in web pages and right formulae are over terms in links.

	Left formula	Right formula	$J$
1.	$\text{dayton}=0 \vee \text{wi}=0 \vee \text{madison}=0$	$\text{back}=0 \vee \text{home}=0 \vee \text{page}=0 \vee \text{to}=0$	0.88
2.	$\text{texa}=0 \vee \text{austin}=0$	$\text{back}=0 \vee \text{home}=0 \vee \text{page}=0 \vee \text{to}=0$	0.83
3.	$\text{instructor}=1 \wedge \text{assign}=1 \wedge \text{hour}=1$	$\text{section}=1 \wedge \text{cs}=1 \vee \text{cse}=1$	0.34

Table 10: Number of descriptions found from original and randomized DBLP data.

Algorithm	Data	$J_{\min}$						
		0.1	0.15	0.2	0.25	0.3	0.35	0.4
GREEDY	Orig.	8	8	8	8	5	1	0
	Rand.	0	0	0	0	0	0	0
MID	Orig.	2440	0	0	0	0	0	0
	Rand.	0	0	0	0	0	0	0

Table 11: Results from the DBLP data with the GREEDY algorithm. Left formulae are over the conferences and right formulae are over the co-authors.

	Left formula	Right formula	$J$
1.	$\text{COLT}=1 \wedge \text{ICML}=1$	$\text{"Michael J. Kearns"}=1 \vee \text{"Peter L. Bartlett"}=1 \vee \text{"Robert E. Schapire"}=1 \vee \text{"Phillip M. Long"}=1$	0.35
2.	$\text{SODA}=1 \wedge \text{FOCS}=1 \wedge \text{STOC}=1$	$\text{"Noga Alon"}=1 \vee \text{"Frank Thomson Leighton"}=1 \vee \text{"Leonidas J. Guibas"}=1 \vee \text{"Sampath Kannan"}=1$	0.32
3.	$\text{SDM}=1 \wedge \text{ICDE}=1$	$\text{"Philip S. Yu"}=1 \vee \text{"Matthias Schubert"}=1 \vee \text{"Jessica Lin"}=1 \vee \text{"Yiming Ma"}=1$	0.30

Table 12: Results from the DBLP data with the MID algorithm. Left formulae are over the conferences and right formulae are over the co-authors.

	Left formula	Right formula	$J$
1.	$\text{SODA}=1 \wedge \text{FOCS}=1 \wedge \text{STOC}=1 \wedge (\text{KDD}=0 \vee \text{PODS}=0)$	$\text{"Noga Alon"}=1 \wedge \text{"Hector Garcia-Molina"}=0$	0.13
2.	$\text{SODA}=1 \wedge \text{FOCS}=1 \wedge \text{STOC}=1 \wedge (\text{PKDD}=0 \vee \text{ICML}=0)$	$\text{"Sanjeev Khanna"}=1 \wedge \text{"Rakesh Agrawal"}=0$	0.09

the formulae is large enough, and the  $p$ -value of the formulae being approximately equivalent is small enough. The significance of the results was evaluated by using swap randomization.

We tested the algorithms on synthetic data, showing that the methods can find planted subgroup descriptions, and that on completely random data the methods do not find many descriptions. On real datasets the algorithms find small sets of results, and the results are easy to interpret. Swap randomization shows that on randomized versions of the datasets the algorithms do not find anything: this implies that the results are not due to chance.

There are obviously several open problems. Our algorithms are quite simple; it would be interesting to know whether more complex methods would be useful. Also, evaluation of the methods on, say, genetic marker data is being planned.

Generalizing the approach to other types of queries is – at least in principle – straightforward. For example, if the data is real-valued, we can ask whether there are subgroups that have at least two different definitions by sets of inequalities. Algorithmically, such variants seem challenging.

## Acknowledgments

The authors are grateful to Gemma Garriga, Aristides Gionis, and Evimaria Terzi for their helpful comments.

## References

- [1] I. I. Artamonova, G. Frishman, and D. Frishman. Applying negative rule mining to improve genome annotation. *BMC Bioinformatics*, 8(261), 2007. <http://www.biomedcentral.com/1471-2105/8/261>.
- [2] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, pages 92–100, 1998.
- [3] G. W. Cobb and Y.-P. Chen. An application of Markov chain Monte Carlo to community ecology. *American Mathematical Monthly*, 110:264–288, 2003.
- [4] A. Gallo, T. De Bie, and N. Cristianini. MINI: Mining informative non-redundant itemsets. In *PKDD*, pages 438–445, 2007.
- [5] A. Gallo, T. De Bie, and N. Cristianini. Mining informative patterns in large databases. Technical report, University of Bristol, UK, September 2007. <http://patterns.enm.bris.ac.uk/node/161>.
- [6] G. C. Garriga, H. Heikinheimo, and J. K. Seppänen. Cross-mining binary and numerical attributes. In *ICDM*, 2007. To appear.
- [7] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3), 2006. Article 9.
- [8] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. *ACM Transactions on Knowledge Discovery from Data*, 1(3), 2007.
- [9] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, second edition, 2006.
- [10] E. Keogh, S. Lonardi, and B. Chiu. Finding surprising patterns in a time series database in linear time and space. In *KDD*, pages 550–556, 2002.
- [11] D. Kumar et al. Algorithms for storytelling. In *KDD*, pages 604–610, 2006.
- [12] N. Lavrac, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *The Journal of Machine Learning Research*, 5:153–188, 2004.
- [13] A. A. Nanavati, K. P. Chitrapura, S. Joshi, and R. Krishnapuram. Mining generalised disjunctive association rules. In *CIKM*, pages 482–489, 2001.
- [14] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.
- [15] L. Parida and N. Ramakrishnan. Redescription mining: Structure theory and algorithms. In *AAAI*, pages 837–844, 2005.
- [16] N. Ramakrishnan et al. Turning CARTwheels: An alternating algorithm for mining redescription. In *KDD*, pages 266–275, 2004.
- [17] A. Savasere, E. Omiecinski, and S. Navathe. Mining for strong negative associations in a large database of customer transactions. In *ICDE*, pages 494–505, 1998.
- [18] M. Sholz. Sampling-based sequential subgroup mining. In *KDD*, pages 265–274, 2005.
- [19] G. Webb. Discovering significant rules. In *KDD*, pages 434–443, 2006.
- [20] G. Webb. Discovering significant patterns. *Machine Learning*, 68(1):1–33, 2007.
- [21] X. Wu, C. Zhang, and S. Zhang. Efficient mining of both positive and negative association rules. *ACM Transactions on Information Systems*, 22(3):381–405, 2004.
- [22] M. J. Zaki and N. Ramakrishnan. Reasoning about sets using redescription mining. In *KDD*, pages 364–373, 2005.
- [23] L. Zhao, M. Zaki, and N. Ramakrishnan. BLOSSOM: a framework for mining arbitrary Boolean expressions. In *KDD*, pages 827–832, 2006.