From Black and White to Full Colour: Extending Redescription Mining Outside the Boolean World[*]

Esther Galbrun[†] Pauli Miettinen[‡]

Abstract

Redescription mining is a powerful data analysis tool that is used to find multiple descriptions of the same entities. Consider geographical regions as an example. They can be characterized by the fauna that inhabits them on one hand and by their meteorological conditions on the other hand. Finding such redescriptors, a task known as niche-finding, is of much importance in biology.

Current redescription mining methods cannot handle other than Boolean data. This restricts the range of possible applications or makes discretization a prerequisite, entailing a possibly harmful loss of information. In niche-finding, while the fauna can be naturally represented using a Boolean presence/absence data, the weather cannot.

In this paper, we extend redescription mining to categorical and real-valued data with possibly missing values using a surprisingly simple and efficient approach. We provide extensive experimental evaluation to study the behaviour of the proposed algorithm. Furthermore, we show the statistical significance of our results using recent innovations on randomization methods.

1 Introduction

Finding multiple ways to characterize the same entities is a problem that appears in many areas of science. In medical sciences, for example, one typically wants to find a subset of patients sharing similar symptoms and similar genes. In biology, the bioclimatic constraints that must be met for a certain species to survive constitute that species' bioclimatic envelope (or niche[1]), and finding such envelopes can help, e.g. to predict the results of global warming [15].

But this process is only semi-automatic. For instance, to find the bioclimatic envelopes, an expert first selects a species and then uses some method to find the envelope for this particular species. More complex combinations of species, or even any combinations at all, are rarely studied, as manually iterating over all possible combinations would be far too laborious.

It is here where *redescription mining* comes to help. In redescription mining the input contains entities with two sets of characterizing variables. The task is to find a pair of queries, one query

[*]A preliminary version of this paper appeared in SDM'11. This is the pre-print version of the original article to appear in *Statistical Analysis and Data Mining*, first published online 10 APR 2012, DOI: 10.1002/sam.11145, `http://onlinelibrary.wiley.com/doi/10.1002/sam.11145/abstract`

[†]Helsinki Institute for Information Technology (HIIT), Department of Computer Science, University of Helsinki, Finland, `esther.galbrun@cs.helsinki.fi`.

[‡]Max-Planck Institute for Informatics, Saarbrücken, Germany, `pmiettin@mpi-inf.mpg.de`. Part of this work was done when the author was with HIIT.

[1]The term *niche* is in this paper used in Grinnellian sense [6], considering only environmental variables, not inter-species competition or such.

for both sets of variables, such that both queries describe (almost) the same set of entities. In niche-finding, the entities would be spatial locations, one set of variables would be the fauna and the other set would contain the bioclimatic variables. A very simple example of a redescription in this setting could say that the area where polar bears live is the area where March's mean temperature is between -16 and -11 degrees Celsius and May's mean temperature is between -3 and -7 degrees Celsius.

Until now, redescription mining algorithms (see [4, 14, 17, 25]) have not been able to handle other than Boolean data. Hence they have not been able to help in the aforementioned cases, not at least without some pre-processing.

The rest of this paper is organized as follows. The next two sections, Sections 2 and 3, present notation and definitions, and related work, respectively. We explain the basic structure of our algorithm in Section 4. In Section 5 we present various extensions to the basic algorithm, including methods for trading some accuracy for speed and handling missing values. The experimental evaluation spans Sections 6–8, with focus on studying the properties of the algorithm and its extensions, comparing it to other algorithms, and a real-world example of niche finding, respectively. Section 9 concludes the paper.

Contributions. In this paper we extend redescription mining to categorical and real-valued data with an algorithm that efficiently computes the optimal discretization on-the-fly. The algorithm can handle missing data. We present experimental studies with synthetic and real-world data to verify that our algorithm scales and returns good results. We also assess the significance of our results by testing them against different null models. Our primary application for real-valued redescription mining is niche-finding, to which we present interesting and intuitive results. The proposed method is also applicable to other domains, e.g. medicine.

2 Notation and Definitions

This paper considers redescriptors over two sets of variables, $V_{\mathbf{L}}$ and $V_{\mathbf{R}}$. The set of entities is denoted by E. We will represent the data using two matrices, $D_{\mathbf{L}}$ and $D_{\mathbf{R}}$. Both matrices have $|E|$ rows and D_i has $|V_i|$ columns. The value of $D_{\mathbf{L}}(i,j)$ is the value of $v_j \in V_{\mathbf{L}}$ for $e_i \in E$. If I is a set of row indices (or a characterizing vector thereof), $D(I,j)$ is the column j of D restricted to the rows in I. The data is a 5-tuple $\mathcal{D} = (V_{\mathbf{L}}, V_{\mathbf{R}}, E, D_{\mathbf{L}}, D_{\mathbf{R}})$. We identify variables in $V_{\mathbf{L}}$ and $V_{\mathbf{R}}$ with the corresponding columns in $D_{\mathbf{L}}$ and $D_{\mathbf{R}}$ when there is no risk of ambiguity.

We consider three types of variables: Boolean, categorical, and numerical (real-valued). If $v \in V$ is Boolean, we interpret the column corresponding to it as a truth value assignment for $e \in E$ in a natural way. If $v \in V$ is real-valued, we consider an interval $[a,b]$, and the truth value assignment induced by the relation $v \in [a,b]$. A special case of this is when v is categorical. Then we consider the relation $v = c$, where c is some category. We will denote these truth value assignments using Iverson notation: $[a \leq v \leq b]$ is the Boolean (column) vector that has 1 in the rows where $v \in [a,b]$, and 0 elsewhere; $[v = c]$ is defined analogously.

These truth assignments and their negations constitute the set of *literals* for variables in V. Notice that there are infinitely many intervals yielding the same truth value assignment for some real-valued $v \in V$. To avoid ambiguity, we consider only the *shortest* interval yielding some truth value assignment. An exception to this is when leaving one side of the interval unbounded is equivalent. We then consider half-lines $(-\infty, b]$ or $[a, +\infty)$, respectively, but for the sake of brevity they are also called intervals. Notice that we can always reconstruct the interval given the data and the truth value assignment corresponding to the interval.

Literals can be combined with Boolean operators $\wedge$ (and) and $\vee$ (or). A *Boolean formula* is made by combining literals with Boolean operators. A *query over* V is a Boolean formula with literals of V. A *redescription* R of $\mathcal{D} = (V_{\mathbf{L}}, V_{\mathbf{R}}, E, D_{\mathbf{L}}, D_{\mathbf{R}})$ is a pair of queries $(q_{\mathbf{L}}, q_{\mathbf{R}})$ over $V_{\mathbf{L}}$ and $V_{\mathbf{R}}$, respectively. For a redescription $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$, we use $V_{\mathbf{L}}(R)$ to denote the variables of $q_{\mathbf{L}}$; $V_{\mathbf{R}}(R)$ is defined analogously.

The support of a query q on $\mathcal{D}$, $\operatorname{supp}_{\mathcal{D}}(q)$, is a set $\{e \in E : q \text{ is true for } e\}$. The support of a redescription $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$, $\operatorname{supp}_{\mathcal{D}}(q_{\mathbf{L}}, q_{\mathbf{R}})$, is the intersection of supports of $q_{\mathbf{L}}$ and $q_{\mathbf{R}}$, $\operatorname{supp}(q_{\mathbf{L}}, q_{\mathbf{R}}) = \operatorname{supp}(q_{\mathbf{L}}) \cap \operatorname{supp}(q_{\mathbf{R}})$. We will omit the subscripts when they are clear from the context.

A redescription $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$ is *exact* if and only if $\operatorname{supp}(q_{\mathbf{L}}) = \operatorname{supp}(q_{\mathbf{R}})$. If a redescription is not exact, it is approximate. The *accuracy* of a redescription $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$ is measured using the *Jaccard coefficient*

$$\mathrm{J}(R) = \mathrm{J}(q_{\mathbf{L}}, q_{\mathbf{R}}) = \frac{|\operatorname{supp}(q_{\mathbf{L}}, q_{\mathbf{R}})|}{|\operatorname{supp}(q_{\mathbf{L}}) \cup \operatorname{supp}(q_{\mathbf{R}})|}.$$

Formally, redescription mining is defined as follows:

PROBLEM 1. (REDESCRIPTION MINING) *Given data* $\mathcal{D} = (V_{\mathbf{L}}, V_{\mathbf{R}}, E, D_{\mathbf{L}}, D_{\mathbf{R}})$ *and a set of constraints* C*, find all redescriptions* $R_1, R_2, \ldots$ *of* $\mathcal{D}$ *that satisfy constraints in* C*.*

We leave open the exact constraints in C for a while and will turn back to it in Subsection 5.1.

The formulation of redescription mining above assumes that the describing variables are partitioned into two sets, $V_{\mathbf{L}}$ and $V_{\mathbf{R}}$, and looks for a pairs of queries over these two sets, respectively. Formulations of redescription mining exist that do not include this requirement. Typically, they consider a single set of describing variables and search for pairs of queries, with the constraint that the two subsets of variables appearing in the queries of any pair be disjoint. The methods presented in this paper can be naturally adapted to that alternative formulation.

Our proposed methods could also be adapted to handle multiple data sets, i.e. settings with more than two sets of variables: $V_A, V_B, \ldots, V_N$, where one looks for tuples of queries $(q_A, q_B, \ldots, q_N)$ over the different sets of variables, respectively.

3 Related Work

3.1 Rule discovery. A characterizing property of redescription mining is its 'many views' approach, that is, it deals with entities that can be explained using different sets of variables. This approach, however, is not unique to redescription mining.

One of the most traditional 'many views' approaches is classification, though it is not typically considered as such. There the data gives one characterization of the entities and the class another. Mining a single query can be considered as a classification task. Fixing one query at a time gives us binary class labels and we try to find a good classifier to it.

Logical Analysis of Data (LAD) [2] is a particular example among classification approaches in the presence of Boolean attributes and target. It aims at finding a perfect classifier of fixed form, e.g. a horn clause, a DNF, a CNF, or linear or quadratic Boolean formula.

Closer to the idea of redescription mining is Multi-label Classification [21], where the goal is to learn classifiers for conjunctions of labels. Perhaps the main difference to redescription mining is this restriction to conjunctions of classes. There is also a big difference in the goals: redescription mining is descriptive while Multi-label Classification is predictive.

The common aim of mining Emerging Patterns, Constrast Set Mining and Subgroup Discovery is to find queries whose support is distributed very unevenly with respect to the target attribute.

Emerging Patterns [12] is targeted at Boolean data and uses monotone conjunctive queries, i.e. itemsets. The purpose is to find itemsets whose presence is statistically dependent on the positive or negative labelling of the objects. In the extreme case, the itemset would be present only in the positive example and would form a perfect classifier for the data at hand. However, this is not generally the case.

Contrast Set Mining [12] can be used with a nominal target attribute to identify a monotone conjunctive query that best discriminates between the objects from one class and the rest of the objects.

Subgroup Discovery [22] aims in a more general sense at finding a query such that the objects in the defined subgroup have atypical values for a target attribute, possibly ordinal or numerical, compared to other objects. This is extended to several target attributes in Exceptional Model mining. In that framework, defined by Leman et al. [10], and in its recent instance [23], one considers a model defined over the target attributes and tries to identify a subgroup of objects where the fitted model differs significantly from the model fitted to the rest of the data.

A related approach is presented by Garriga, Heikinheimo and Seppänen [5]. It uses frequent itemsets on the binary attributes to form a partition of the original entities such that for each subset it is possible to construct a specific model that fits well on the numerical attributes. In other words, this approach tries to partition the original data into subgroups.

Redescription mining differs from the above techniques in that it aims at simultaneously finding multiple descriptions of a subset of entities which is not previously specified, selecting the few relevant among a potentially large set of variables. In contrast to these methods, it does not have a set of describing features and target attributes but rather several sets of describing variables. Yet, when there are two sets of describing attributes and a chosen query language, we can define a one-directional redescription problem. Queries can be built over one set of attributes, defining subgroups whose quality is measured in terms of how exactly and concisely they can be described by queries over the other set of attributes. In a sense, this can be loosely understood as a case of Exceptional Model Mining where the model is the chosen query language and fitting the model to a subgroup corresponds to finding as concise and exact a query for it over the target attributes as possible. The aim of redescription mining is then to solve this problem in both directions simultaneously.

Redescription mining was introduced in [17], and has since attained continuous research interest (e.g. [14, 25, 4, 9]). The approches proposed for redescription mining have been based on various ideas, including decision trees [17, 9], Karnaugh maps [25], co-clusters [14] and frequent itemsets [25, 4].

The CARTwheels algorithm [17] is an alternating method that uses decision trees. One side of the redescription is fixed, giving binary labels for the entities and a decision tree over the other variables, that is, a good classifier with respect to those labels, is constructed. At the next step, the labelling given by this decision tree is considered as the target and a new decision tree is built on the other set of variables. The branches of the two trees that correspond to positive labels form a pair of queries that can be considered as a redescription, as well as the pair of queries associated to the negative branches. This construction based on decision trees gives the redescriptions an atypical form. Consider the example of a tree of depth two, with the first level branching variable A and the second level branching variables B and C. Then, $(\textsc{a} \land \textsc{b}) \lor (\neg\textsc{a} \land \textsc{c})$ is a good example of query obtained by joining branches of such a tree. The fact that the same variable occurs multiple times with and without negation can make the query difficult to interpret.

Gallo et al. [4] propose two approaches to redescription mining. The first one is based on mining frequent itemsets from both data sets separately and combining them together. The second one is a greedy method that forms the basis of our work. We will therefore come back to it in more details.

A natural extension of redescription mining is *storytelling* [14], where the aim is to find consecutive redescriptors. That is, given data $\mathcal{D} = (V_{\mathbf{L}}, V_{\mathbf{R}}, E, D_{\mathbf{L}}, D_{\mathbf{R}})$, the goal is to find queries $q^1_{\mathbf{L}}, q^1_{\mathbf{R}}, q^2_{\mathbf{L}}, q^2_{\mathbf{R}}, \ldots$ such that consecutive pairs of queries $(q^1_{\mathbf{L}}, q^1_{\mathbf{R}}), (q^1_{\mathbf{R}}, q^2_{\mathbf{L}}), (q^2_{\mathbf{L}}, q^2_{\mathbf{R}}), \ldots$ all form a valid non-exact redescription. We will not cover storytelling in this paper.

3.2 Data discretization. Generalizing algorithms based on Boolean attributes to real-valued data has been a recurrent problem in data mining. Most solutions are based on some sort of pre-processing: typically categorical data is represented using one variable per category, and quantative data is turned into categorical data using some type of bucketing.

When labels are available on the original data, as is the case for Subgroup Discovery with a single output feature, a supervised discretisation method can be devised for the problem at hand. In the method proposed by Grosskreutz and Rüping [7], the discretization happens within the algorithm and relies on a property of the function measuring subgroup quality to merge basic intervals in a bottom-up fashion. Yet, the end points for the basic intervals are determined as a pre-processing step in a way that is not necessarily optimal with respect to their later use.

In most settings, though, no labelling of the data is available and one has to resort to unsupervised discretization. This approach raises several questions, from the choice of the number of buckets to the size of the resulting data. A more elegant approach was provided by Srikant and Agrawal [20], who presented a machinery that solves most of the problems automatically. Their method is still based on a priori bucketing, and moreover, it is very specific to association rule mining, making it hard (or impossible) to apply to redescription mining.

The problem of on-the-fly discretization during a classification task was studied by Fayyad and Irani [3]. Although the task is different, the results we obtained for dynamically choosing the most accurate extension shares similarities with their result.

Using redescription mining algorithms with non-Boolean data is not a new idea. Already in [17], the CARTwheels algorithm was used to mine bioinformatics data that was non-Boolean. As the algorithm requires Boolean input, the data had to be bucketed as a pre-processing step. But pre-processing typically requires considerable domain knowledge and might still be impossible or yield exponential growth in the number of variables. This is in contrast to our algorithm, where the optimal discretization is determined at each iteration within the algorithm, requiring no pre-processing. Nothing, of course, prevents users to pre-process their data, should that be needed.

3.3 Niche finding. In biology, the problem of finding species' bioclimatic envelope is a rather new one (see, e.g. [18] and references therein), but the idea of ecological niches dates back to the early 20th century [6]. There is also some level of ambiguity in what exactly is meant by the term niche [18]. In this paper we consider a bioclimatic envelope of a (group of) species to be a set of limits in climate variables (such as monthly mean temperature) that defines the region occupied by the species[2].

Despite the vague definition, the past ten years have seen a number of methods to model the bioclimatic envelopes. The methods are based, for example, on regression, neural networks, and genetic algorithms (see [19]). But to the best of the authors' knowledge, none of these methods allows automatically finding both the set of species and their envelope.

Other niche finding tasks have been formulated, for example, in a linguistic context [16]. In this paper we only consider the biological problem.

[2]i.e. we consider realized niches using correlative methods (see [15]).

4 The Basic Algorithm

In this section, we present the core of our algorithm. Various extensions to it, such as handling missing values, are presented in the next section.

4.1 Motivation and background. The redescription mining problem is defined for general Boolean queries, yet none of the proposed algorithms explores the full search space (using instead decision trees of fixed depth [17], monotone CNF and DNF formulae [14], or only (possibly negated) conjunctions [25]). Such restrictions are easy to understand, given the huge search space formed by all Boolean formulae (2^{2^n} distinct formulae can be defined over n variables). With non-Boolean data the search space is even more overwhelming. It is therefore evident that when devising an algorithm usable with real-world data, the space of all Boolean functions cannot be considered in its entireness.

Type of Boolean queries mined. How to restrict the search space? This question can be considered from at least three different perspectives: the expressive power of the resulting queries, the ease of finding them, and their interpretability. For example, monotone conjunctive queries (i.e. frequent item sets) are easy to interpret and relatively easy to find, given the monotonicity of the search space, but they lack expressive power. On the other hand, general Boolean queries have a high expressivity, but are hard to find. Furthermore, deeply nested structures and variables appearing multiple times can lead to difficulties in interpreting them.

Our aim is to restrict the search to queries that provide a good compromise between the three aforementioned properties. For this purpose, we follow the approach taken by Gallo, Miettinen and Mannila [4]. First, we evaluate the queries from left to right irrelevant of the operator precedence. In other words, we only consider queries that can be parsed in linear order, without trees. For example, $(a \vee b) \wedge \neg c$ is such a query, but $(a \wedge b) \vee (c \wedge d)$ is not. Second, we allow every variable to appear only once. Queries of this type are strict generalizations of purely conjunctive or disjunctive queries, save the tautological cases $a \wedge \neg a$ and $a \vee \neg a$. We consider such queries to be relatively easy to interpret while still having a satisfying expressive power. While becoming smaller, the search space still remains exponential. Therefore, we also employ a heuristic pruning, as will be explained later.

On-the-fly bucketing versus pre-processing. Binning the variables into buckets is a standard pre-processing technique to make non-Boolean data Boolean (see Section 3.2). But it has its drawbacks. For example, the resulting data has a special structure, with all variables corresponding to different buckets of a given non-Boolean variable being mutually disjoint. The algorithms are typically not adjusted to this property.

Moreover, the bucketing must be made in a pre-processing step, and cannot be modified by the algorithm later on. If the quality of the bucketing was poor, so will be the results. But the user typically does not know whether a certain bucketing yields good results before running the algorithm, so repeated trials and errors are needed to achieve satisfactory results.

Our approach of doing the bucketing on-the-fly avoids these problems. The algorithm will select the optimal bucket for each case when necessary. This removes the need of pre-processing and repeated trials. Furthermore, our algorithm can use different buckets for the same variable in different redescriptions, should that yield better results.

4.2 Outline of the algorithm. We use a strategy similar to beam-search to explore the solution space. The basic idea is to construct queries bottom-up, starting from singleton redescriptions (i.e. both queries contain only one literal) and progressively extending them by appending operators and literals. For example, we could start with a pair $(a, \neg b)$, and try to extend it to $(a \wedge c, \neg b)$,

```
    Input: Data $\mathcal{D} = (V_{\mathbf{L}}, V_{\mathbf{R}}, E, D_{\mathbf{L}}, D_{\mathbf{R}})$, nonnegative integers $k_p$ and $k_i$, constraints $C$.
    Output: A set of redescriptions, $\mathcal{R}$.
 1: $\mathcal{R} \leftarrow \emptyset$
 2: $\mathcal{I} \leftarrow \{k_p$ best initial singleton redescriptions $\}$
 3: for $S \in \mathcal{I}$ do
 4:     $\mathcal{K} \leftarrow \{S\}$
 5:     $F_{\mathbf{L}}(S), F_{\mathbf{L}}(S) \leftarrow$ free variables for $S$
 6:     if $F_{\mathbf{L}}(S) \neq \emptyset$ or $F_{\mathbf{R}}(S) \neq \emptyset$ then
 7:         $\mathcal{E} \leftarrow \{S\}$
 8:     while $\mathcal{E} \neq \emptyset$ do
 9:         for each $R \in \mathcal{E}$ do
10:             for side $s \in \{\mathbf{L}, \mathbf{R}\}$ and operator $\circ \in \{\vee, \wedge\}$ do
11:                 if $R$ can be extended on side $s$ with operator $\circ$ and literal $l \in F_s(R)$ then
12:                     $\mathcal{K} \leftarrow \mathcal{K} \cup \{$best such extension of $R$ admitting constraints $C\}$
13:         $\mathcal{K} \leftarrow \{k_i$ best redescriptions from $\mathcal{K}$, with updated free variables.$\}$
14:         $\mathcal{E} \leftarrow \{R \in \mathcal{K} : F_{\mathbf{L}}(R) \neq \emptyset$ or $F_{\mathbf{R}}(R) \neq \emptyset\}$
15:     $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{K}$
16: Filter $\mathcal{R}$ with constraints $C$
17: return $\mathcal{R}$
```

Figure 1: ReReMi: Mine data sets for redescriptions

$(a \vee c, \neg b)$, $(a \wedge \neg c, \neg b)$, etc. After evaluating all possible one-step extensions, we select the best candidates and extend them in turn. This process requires a book-keeping procedure to avoid repeatedly generating the same queries, as we will explain below. When no new redescription can be generated, we move to the next initial pair. The outline of the algorithm, called ReReMi, is given in Figure 1.

Our algorithm shares similarites with the Greedy algorithm presented by Gallo et al. [4]. But unlike Gallo et al., and following the idea of beam-search, we allow several extensions to be generated from a given redescription in each step. In this way we can explore the search space more extensively. Notice also our algorithm's resemblance to bottom-up frequent itemset mining algorithms. Indeed, finding redescriptions can be seen as a generalization of association rule mining [25], although without the monotonicity property.

4.3 Efficient computation of the accuracy for Boolean variables. Given two queries, q_A and q_B, to decide which of the possible extensions of q_A yields the best redescription, we need to compute the accuracy (i.e. Jaccard coefficient) for four different types of extensions for each Boolean variable v: $\mathrm{J}(q_A \wedge v, q_B)$, $\mathrm{J}(q_A \wedge \neg v, q_B)$, $\mathrm{J}(q_A \vee v, q_B)$ and $\mathrm{J}(q_A \vee \neg v, q_B)$. Doing this in a straight forward way, determining a single Jaccard coefficient requires the computation of three distinct supports over the data. But this is not necessary. To compute $\mathrm{J}(q_A \wedge v, q_B)$, we need consider only the rows in $\mathrm{supp}(q_A)$ – others will never be in $\mathrm{supp}(q_A \wedge v)$. On the other hand, rows in $\mathrm{supp}(q_A)$ will be in $\mathrm{supp}(q_A \vee v)$ in any case and can be omitted when computing $\mathrm{J}(q_A \vee v, q_B)$. Let us formalize this intuition.

Let $E_{1,0}$ be the set of entities for which only the first query holds (i.e. $E_{1,0} = \mathrm{supp}(q_A) - \mathrm{supp}(q_B)$), $E_{0,1}$ those for which only the second query holds, $E_{1,1}$ those for which both queries hold, and $E_{0,0}$ those for which neither of the queries hold. Finally, these sets restricted to $\mathrm{supp}(v)$ are denoted as $E_{x,y}(v)$ (e.g. $E_{1,0}(v) = E_{1,0} \cap \mathrm{supp}(v)$). The same notation is also used with real-valued variables and Iverson notation, as in $E_{1,0}([\lambda \leq v \leq \rho])$.

$$\begin{aligned}
\mathrm{J}(q_A \wedge v, q_B) &= \frac{|E_{1,1}(v)|}{|E_{1,0}(v)| + |E_{0,1}| + |E_{1,1}|} \\
\mathrm{J}(q_A \wedge \neg v, q_B) &= \frac{|E_{1,1}| - |E_{1,1}(v)|}{|E_{1,0}| - |E_{1,0}(v)| + |E_{0,1}| + |E_{1,1}|} \\
\mathrm{J}(q_A \vee v, q_B) &= \frac{|E_{1,1}| + |E_{0,1}(v)|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}| + |E_{0,0}(v)|} \\
\mathrm{J}(q_A \vee \neg v, q_B) &= \frac{|E_{1,1}| + |E_{0,1}| - |E_{0,1}(v)|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}| + |E_{0,0}| - |E_{0,0}(v)|}
\end{aligned}$$

Figure 2: Formulae for computing the Jaccard coefficient for different extensions.

It is well known that the Jaccard coefficient $\mathrm{J}(q_A, q_B)$ can be expressed as

$$\mathrm{J}(q_A, q_B) = \frac{|E_{1,1}|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}|}. \tag{4.1}$$

Similarly, we can write $\mathrm{J}(q_A \wedge v, q_B) = |E_{1,1}(v)| / (|E_{1,0}(v)| + |E_{0,1}| + |E_{1,1}|)$. Analogous formulae can be derived for all different extensions (see Figure 2).

Notice that $E_{1,0}$, $E_{0,1}$, and $E_{1,1}$ can be computed once for a given redescription. Then, for each candidate variable, it is enough to perform three intersection operations to obtain $E_{1,0}(v)$, $E_{0,1}(v)$, and $E_{1,1}(v)$. Furthermore, $E_{0,0}$ and $E_{0,0}(v)$ can be deduced from $\mathrm{supp}(v)$ and E, and we do not have to consider the rows in which neither q_A nor q_B hold. This observation can significantly speed up the algorithm.

4.4 Adding categorical variables. Handling the categorical variables is rather straight forward. We consider only the relation $v = c$, where c is the label. The above computation of Jaccard applies, as we can write, for example,

$$\mathrm{J}(q_A \wedge [v = c], q_B) = \frac{|E_{1,1}([v = c])|}{|E_{1,0}([v = c])| + |E_{0,1}| + |E_{1,1}|}.$$

What is different to the Boolean case is that we must select the class label c. But this we can do easily by trying all class labels and selecting the one that improves the Jaccard most. Naturally we can use the aforementioned speedup techniques for Jaccard when selecting the label, as each label just defines different Boolean vector.

4.5 Extension to real-valued variables. With the real-valued data, our approach is to do bucketing on-the-fly, finding the optimal bucket to add in every step. Assume that our algorithm tries to extend, say, query q_A of redescription (q_A, q_B) with a real-valued variable v. The algorithm considers the extended query $q_A \wedge [\lambda \leq v \leq \rho]$ for different thresholds λ and ρ and selects those that maximize the accuracy of the extension. Naturally, the optimal λ and ρ are different for different extensions. The two thresholds are set simultaneously since setting one bound first and possibly the other later would prevent the greedy search from finding some of the most specific intervals.

How can we find λ and ρ efficiently? To tackle this question, we adapt our approach from the previous section, using a result similar to that of Fayyad and Irani [3].

We consider only the *shortest* interval yielding any truth value assignment: only values in $D(E, v)$, i.e. values taken by the variable v, can be interval bounds. We could try all possibilities, but if the data contains n entities, this can require n^2 time, quickly becoming infeasible since we have to compute the accuracies for each *candidate* extension. However, as for the Boolean case,

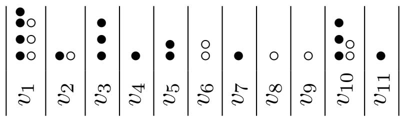


Figure 3: Example of repartition of the entities for one variable. Each bin represents a value taken by the variable. Black circles stand for entities belonging to $E_{1,1}$, white circles for entities from $E_{1,0}$.

only two subsets of entities for each type of extension can impact the Jaccard coefficient: those in $E_{1,1}$ and $E_{1,0}$ for conjunctions and those in $E_{0,1}$ and $E_{0,0}$ for disjunctions. Furthermore, only values separating entities from the two sets need to be considered.

Definition 4.1. *When extending a redescription with a numerical variable v, we say that value u is a* lower cut point *if for any fixed value w larger than u, $[u, w]$ is the shortest interval yielding a locally optimal support for the extension. Equivalently, we say that w is an* upper cut point *if for any fixed value u smaller than w, $[u, w]$ is the shortest interval yielding a locally optimal support for the extension. In addition, $-\infty$ and $+\infty$ can be lower and upper cut points, respectively.*

Example. Consider the example of extending a redescription (q_A, q_B) by appending a variable v with a non-negated conjunction. The bins in Figure 3 represent the values taken by the variable, sorted in increasing order. Black circles stand for entities belonging to $E_{1,1}$, white circles for entities from $E_{1,0}$. Our aim is to find bounds λ and ρ that maximize

$$j(\lambda, \rho) = \frac{|E_{1,1}([\lambda \le v \le \rho])|}{|E_{1,0}([\lambda \le v \le \rho])| + |E_{0,1}| + |E_{1,1}|}. \tag{4.2}$$

In this example, there is one entity in $E_{1,1}$ with value v_4, but none in $E_{1,0}$ with value v_3. Therefore v_4 cannot be an optimal choice for λ since choosing v_3 instead would always increase the accuracy. For the same reason, $-\infty$, v_2, v_3, v_7, v_{10}, and v_{11} are lower cut points, as only these values can be an optimal choice for λ here. Similarily, v_1, v_2, v_5 and v_7 are upper cut points.

To improve the speed of computing optimal extensions, we need to identify lower and upper cut points efficiently. To that end, we present succinct characterisations of the cut points for different types of extensions. We use $(v_1, v_2, \ldots, v_k)$ to denote the values taken by the variable v for entities in $E_{1,1}$ or $E_{1,0}$, that is, values in $D(E_{1,0} \cup E_{1,0}, v)$, sorted in increasing order. Similarly, the values in $D(E_{0,0} \cup E_{0,1}, v)$ ordered increasingly are denoted as $(v'_1, v'_2, \ldots, v'_l)$.

Proposition 4.1. *For a* non-negated conjunction*, a lower cut point is a value v_i such that $v_i \in D(E_{1,1}, v)$ and $v_{i-1} \in D(E_{1,0}, v)$, or $-\infty$ if $i = 1$ and $v_i \in D(E_{1,1}, v)$. An upper cut point is a value v_j such that $v_j \in D(E_{1,1}, v)$ and $v_{j+1} \in D(E_{1,0}, v)$, or $+\infty$ if $j = k$ and $v_j \in D(E_{1,1}, v)$.*

For a negated conjunction*, a lower cut point is a value v_i such that $v_i \in D(E_{1,0}, v)$ and $v_{i-1} \in D(E_{1,1}, v)$, or $-\infty$ if $i = 1$ and $v_i \in D(E_{1,0}, v)$. An upper cut point is a value v_j such that $v_j \in D(E_{1,0}, v)$ and $v_{j+1} \in D(E_{1,1}, v)$, or $+\infty$ if $j = k$ and $v_j \in D(E_{1,0}, v)$.*

For a non-negated disjunction*, a lower cut point is a value v'_i such that $v'_i \in D(E_{0,1}, v)$ and $v'_{i-1} \in D(E_{0,0}, v)$, or $-\infty$ if $i = 1$ and $v'_i \in D(E_{0,1}, v)$. An upper cut point is a value v'_j such that $v'_j \in D(E_{0,1}, v)$ and $v'_{j+1} \in D(E_{0,0}, v)$, or $+\infty$ if $j = l$ and $v'_j \in D(E_{0,1}, v)$.*

For a negated disjunction*, a lower cut point is a value v'_i such that $v'_i \in D(E_{0,0}, v)$ and $v'_{i-1} \in D(E_{0,1}, v)$, or $-\infty$ if $i = 1$ and $v'_i \in D(E_{0,0}, v)$. An upper cut point is a value v'_j such that $v'_j \in D(E_{0,0}, v)$ and $v'_{j+1} \in D(E_{0,1}, v)$, or $+\infty$ if $j = l$ and $v'_j \in D(E_{0,0}, v)$.*

Proof. We concentrate on the case of non-negated conjunction; other cases are similar. The aim is to maximize $j(\lambda, \rho)$ (cf. Equation 4.2). We start with the lower bound λ. First, suppose $v_i \notin D(E_{1,1}, v)$. Then v_i must occur in $E_{1,0}$ and we have $|E_{1,0}([v_i \leq v \leq \rho])| > |E_{1,0}([v_{i+1} \leq v \leq \rho])|$ while $|E_{1,1}([v_i \leq v \leq \rho])| = |E_{1,1}([v_{i+1} \leq v \leq \rho])|$. Hence, $j(v_i, \rho) < j(v_{i+1}, \rho)$. So v_i is not an optimal value for λ. Second, if $v_{i-1} \notin D(E_{1,0}, v)$, following a similar reasoning, we notice that $j(v_{i-1}, \rho) > j(v_i, \rho)$ and v_i is not an optimal value for λ. Finally, in case $v_1 \in D(E_{1,1}, v)$, setting $\lambda = v_1$ can be optimal and we simply leave the lower bound undefined, i.e. we use the half-line $(-\infty, \rho]$ that yields the same support.

The case of the upper bound ρ is analogous. If $v_i \notin D(E_{1,1}, v)$, then $v_i \in D(E_{1,0}, v)$ and we have $|E_{1,0}([\lambda \leq v \leq v_{i-1}])| < |E_{1,0}([\lambda \leq v \leq v_i])|$ while $|E_{1,1}([\lambda \leq v \leq v_{i-1}])| = |E_{1,1}([\lambda \leq v \leq v_i])|$. Hence, $j(\lambda, v_{i-1}) > j(\lambda, v_i)$ and v_i is not an optimal value for ρ. On the other hand, if $v_{i+1} \notin D(E_{1,0}, v)$ then $j(\lambda, v_{i-1}) > j(\lambda, v_i)$ and v_i is not an optimal value for ρ. Finally, when v_k occurs in $E_{1,1}$ for the last index k, setting $\rho = v_k$ can be optimal and we use the half-line $[\lambda, +\infty)$ which is equivalent with respect to the support.

The case of negated conjunctions is the reverse of this. Again, we only need to consider entities in $E_{1,0}$ or in $E_{1,1}$, but this time we will try to find an interval $[\lambda \leq v \leq \rho]$ such that the set $|E_{1,0}([\lambda \leq v \leq \rho])|$ is large while $|E_{1,1}([\lambda \leq v \leq \rho])|$ is small, so as to maximize

$$j(\lambda, \rho) = \frac{|E_{1,1}| - |E_{1,1}([\lambda \leq v \leq \rho])|}{|E_{1,0}| - |E_{1,0}([\lambda \leq v \leq \rho])| + |E_{0,1}| + |E_{1,1}|}. \tag{4.3}$$

The case of disjunctions is very similar to conjunctions, but focusing on entities in $E_{0,1}$ and $E_{0,0}$ instead of $E_{1,1}$ and $E_{1,0}$, respectively.

For non-negated disjunctions the aim is to maximize

$$j(\lambda, \rho) = \frac{|E_{1,1}| + |E_{0,1}([\lambda \leq v \leq \rho])|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}| + |E_{0,0}([\lambda \leq v \leq \rho])|}, \tag{4.4}$$

that is, we try to maximize the number of entities in $E_{0,1}$ while minimizing that in $E_{0,0}$.

The aim for negated disjunctions is to maximize

$$j(\lambda, \rho) = \frac{|E_{1,1}| + |E_{0,1}| - |E_{0,1}([\lambda \leq v \leq \rho])|}{|E| - |E_{0,0}([\lambda \leq v \leq \rho])|}, \tag{4.5}$$

the reverse of the non-negated case, similarly to conjunctions. □

To search for an optimal bucket for variable v, we only need to consider upper and lower cut points. Denoting by n_λ the number of lower cut points and by n_ρ the number of upper cut points, the size of the search space is $(n_\lambda + 1)(n_\rho + 1)$. In many cases, this is considerably smaller than the naïve n^2 (but see Section 5.2 for a method to deal with large $(n_\lambda + 1)(n_\rho + 1)$).

4.6 Putting it all together: The ReReMi algorithm. As we mentioned previously, the algorithm starts by evaluating all possible pairs of singleton redescriptions (i.e. literals) and keeps only the k_p best pairs (line 2). Alternatively, it is possible to extend all pairs with accuracy higher than some threshold or exhaust all the pairs in order to discover redescriptions with low first level accuracy. But after some number of initial pairs, a drop in the accuracy of the generated redescriptions can typically be observed. Limiting k_p is therefore reasonable.

Generating the initial pairs from real-valued data requires some extra work. There are two options. First, if one of the matrices (say $D_\mathbf{L}$) is Boolean while the other is real-valued, we create the initial pairs by considering redescriptions $R = (v_{\mathbf{L}}, \emptyset)$ for each $v_{\mathbf{L}} \in V_{\mathbf{L}}$, and extending their right-hand side using the standard on-the-fly bucketing approach. Second, if both sides are real-valued, an exhaustive search of all possible intervals needs to be performed. This might be computationally very expensive, and in Section 5.3 we present a method to find the initial pairs faster with possible loss in accuracy.

Each of the initial pairs is extended in turn (lines 3–15), selecting at each step the k_i most promising candidates (line 13). A value of 4 for k_i, for example, enables to keep the candidates for both operators and both sides on the first step. Two sets of variables, $F_{\mathbf{L}}(R) \subset V_{\mathbf{L}}$ and $F_{\mathbf{R}}(R) \subset V_{\mathbf{R}}$, are associated to each redescription R. They contain the variables that can be used to expand that redescription, which we call the free variables of R. The free variables are determined so as to avoid generating several times the same redescription. The variables leading from R to some already generated one-step extension, i.e. redescriptions obtained by appending one literal to it, are not free for R; this includes the variables that appear in R.

For the purpose of determining free variables, the algorithm maintains a list of the redescriptions generated so far. We assign a set of keys to every redescription, one for each literal involved. The redescriptions can be indexed using these keys in space bounded by the number of explored redescriptions multiplied by a factor quadratic in the maximal allowed query length. Then, given any redescription containing l literals, we can retrieve the sets of redescriptions associated to each of its keys in l accesses to the index. Its one-step extensions are the redescriptions of length $l + 1$ in the intersection of these sets.

In addition, when the query on either side of the redescription has reached the maximum number of variables, all remaining free variables for that side are removed. Among the selected candidates, those that have some free variables are put into the set $\mathcal{E}$ of redescriptions to be extended during the next iteration (lines 14). The loop ends when $\mathcal{E}$ is empty, that is, when there is no extendible redescription left.

5 Extensions to the Algorithm

With the basic algorithm presented in the previous section, we now turn to study some extensions to it. We first start by studying the possible constraints one can apply for the redescriptions, then present two methods that can be used to speed up the algorithm (possibly trading off some accuracy) with real-valued data. Finally, we explain how to extend the algorithm to handle missing values.

5.1 Constraints on the redescriptions. In this section, we discuss the different constraints one can apply to redescriptions. The accuracy is a simple constraint: leave out all redescriptions with accuracy lower than some threshold. But in addition to being accurate, we would like the redescriptions to be statistically significant. That is, the support of a redescription $(q_{\mathbf{L}}, q_{\mathbf{R}})$ should carry some new information, given the support of the queries. To measure this, we test against the null-model representing the case in which the two queries would be independent. We compute a p-value that represents the probability that two random queries with marginal probabilities (i.e. the fraction of entities supporting them) equal to those of $q_{\mathbf{L}}$ and $q_{\mathbf{R}}$ have an intersection equal to or larger than $|\mathrm{supp}(q_{\mathbf{L}}, q_{\mathbf{R}})|$. This probability uses the binomial distribution and is given by

$$\mathrm{pvalM}(q_{\mathbf{L}}, q_{\mathbf{R}}) = \sum_{s=|\mathrm{supp}(q_{\mathbf{L}}, q_{\mathbf{R}})|}^{|E|} \binom{|E|}{s} (p_R)^s (1 - p_R)^{|E|-s},$$

where $p_R = |\mathrm{supp}(q_{\mathbf{L}})|\,|\mathrm{supp}(q_{\mathbf{R}})| \,/\, |E|^2$. The higher the p-value, the more likely it is to observe such a support for independent queries, and the less significant the query.

Similarly, when appending a literal l to a redescription, we would like it to be as informative as possible. On one hand, if q_A and l have very similar supports, $q_A \vee l$ will not carry much more information than the original q_A, so we want q_A and l to be as uncorrelated as possible and intersect less than would occur randomly. On the other hand, when extending q_A to $q_A \wedge l$ we expect q_A and l to be correlated and intersect more than would occur randomly. Hence we define $\mathrm{pvalE}(q_s, \wedge l) = \mathrm{pvalM}(q_s, l)$ and $\mathrm{pvalE}(q_s, \vee l) = 1 - \mathrm{pvalM}(q_s, l)$.

Also important are the size of the support of the redescriptions and the number of entities by which each variable contributes to it, since redescriptions characterizing too few entities or almost all of them are of no interest.

These constraints on p-value and support can be applied more or less strictly during the beam search, using thresholds to penalize or simply disqualify candidate redescriptions that do not comply with them. The redescription p-value and support size can also be used to filter out uninteresting results a posteriori. Of course, the stricter the constraints applied within the candidate selection, the faster the search, but the more likely it becomes to miss candidates that would expand to acceptable redescriptions.

The type of query can also be selected, for example to disallow negations or use only disjunctions. Most of the constraints need not be tuned for good results, but they can be used to incorporate domain knowledge or to guide the algorithm to search for special redescriptors.

5.2 Interval approximation. In cases where the search space of possible intervals is still too large (i.e. $(n_\lambda + 1)(n_\rho + 1)$ is too big, cf. Section 4.5), we use a faster search to find an interval whose accuracy is a good lower bound to the optimal one.

Let $(t_0, t_1, \ldots, t_{n_\lambda + n_\rho + 1})$ be the ordered list of cut points (as per Proposition 4.1), with the special cases $t_0 = -\infty$ and $t_{n_\lambda + n_\rho + 1} = +\infty$. Let l, i, and u be any indices such that $S_1 = [t_l, t_i]$, $S_2 = [t_i, t_{i+1}]$, and $S_3 = [t_{i+1}, t_u]$ are valid intervals. Note that we can have $t_i = t_{i+1} = w$, when the value w is both a *lower* and an *upper cut point*.

On one hand, if the accuracy obtained by merging intervals S_1 and S_2 is lower than that of S_2 alone, then merging S_1, S_2, and S_3 yields lower accuracy than S_2 alone or merging S_2 and S_3, for any interval S_3. That is, if $j(t_l, t_{i+1}) < j(t_i, t_{i+1})$, then

$$j(t_l, t_u) < \max(j(t_i, t_{i+1}), j(t_i, t_u)).$$

We use this property to find the best interval by upward aggregation. Starting with the first interval, we construct at each iteration an interval of the form $I_i = [\lambda_i, t_i]$. We go through the possible optimal values in ascending order, while keeping track of the best accuracy encountered: $\lambda_{i+1} = t_i$ if $j(\lambda_i, t_{i+1}) < j(t_i, t_{i+1})$, and $\lambda_{i+1} = \lambda_i$ otherwise

On the other hand, if the accuracy obtained by merging intervals S_1 and S_2 is greater than that of S_2 alone, there might still be an interval S_3 such that merging S_2 and S_3 yields a higher accuracy than S_1, S_2 and S_3 together. That is, even if $j(t_i, t_{i+1}) < j(t_l, t_{i+1})$, it does not necessarily follow that $j(t_i, t_u) < j(t_l, t_u)$. Therefore, we also compute the best interval using downward aggregation, starting with the last interval and iterating over the possible optimal values in reverse order. Then we combine the two best intervals to eliminate possible undesirable values on either ends. Let I_u and I_d denote the best intervals found using upward and downward aggregations, respectively. The final interval returned is either I_u, I_d, or $I_u \cap I_d$, depending on which maximizes $j()$. Using this method, we can compute an interval that approximates the optimal accuracy in $O(n_\lambda + n_\rho)$. This is especially useful in cases where the rows in $E_{1,0}$ and $E_{1,1}$ (for conjunctions) or $E_{0,1}$ and

$E_{0,0}$ (for disjunctions) are not clearly separated, saving heavy computations when encountering variables that are intuitively poor extensions.

5.3 Approximating the initial pairs for real-valued data. The approximate search presented in Section 5.2 might not be sufficient to find the initial pairs in reasonable time. This is especially the case when the data contain dense variables with many different values on both sides. We then resort to unsupervised bucketing to reduce the number of intervals tested and make the computation tractable.

During the initial pair generation, when the number of intervals to test for a given pair of variables exceeds a predefined threshold, different values of the variables are aggregated together to construct non-overlapping contiguous buckets. Only the intervals corresponding to the bounds of the buckets are tested. The agressiveness of the method can be adapted by tuning the threshold on the original number of intervals and the number of entities that can be grouped.

Unlike with usual pre-bucketing approaches, the static buckets are only used for this special cases of initial pair generation; during the later extension steps, the algorithm determines the intervals dynamically as described previously.

5.4 Handling missing values. Real-world data often contains missing values, i.e. cases where not all values are known for all entities. In order to handle such data, we consider the extended truth value set $\{\mathrm{True}, \mathrm{False}, \mathrm{Missing}\}$. Any truth value assignment for a variable with missing values will lead to Missing as the value for those entites. When evaluating Boolean queries, the following new cases emerge: the negation of Missing is Missing; the value of $X \land \mathrm{Missing}$ is False if X is False and Missing otherwise; the value of $X \lor \mathrm{Missing}$ is True if X is True and Missing otherwise.

With missing values, there are five new groups of entities given a redescription (q_A, q_B): either q_A or q_B can be Missing while the other is True or False, or they are both Missing. Following the notation of already-defined sets $E_{1,0}$, $E_{0,1}$, $E_{1,1}$ and $E_{0,0}$, we denote these sets by $E_{1,?}$, $E_{0,?}$, $E_{?,1}$, $E_{?,0}$, and $E_{?,?}$.

The presence of missing values requires us to re-define how we compute the quality of the redescription, i.e. the Jaccard coefficient. A common approach is to ignore those entities that have missing values. With relatively few missing values that are evenly distributed, this should give a reasonably good estimate of the true accuracy. We call this method *rejective Jaccard*, denoted J_R, to distinguish from the cases with no missing values.

The rejective Jaccard has its problems, though. If the missing values are not evenly distributed, we may have to reject too much data to be able to obtain reasonable estimates. Therefore, we also consider two other estimates: the *optimistic Jaccard* (J_O) and the *pessimistic Jaccard* (J_P), defined as follows:

$$\mathrm{J}_\mathrm{O}(q_A, q_B) = \frac{|E_{1,1}| + |E_{1,?}| + |E_{?,1}| + |E_{?,?}|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}| + |E_{1,?}| + |E_{?,1}| + |E_{?,?}|} \tag{5.6}$$

$$\mathrm{J}_\mathrm{P}(q_A, q_B) = \frac{|E_{1,1}|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}| + |E_{0,?}| + |E_{?,0}| + |E_{?,?}|}\ . \tag{5.7}$$

The optimistic Jaccard gives the upper bound and the pessimistic the lower bound of the Jaccard coefficient when truth values are assigned to the missing values. In other words, the optimistic Jaccard corresponds to the accuracy obtained if one could assign truth values to the missing values in the most favorable way while the pessimistic Jaccard is the accuracy that would result from an adversarial assignment of the missing values.

These three accuracy measures change the way our algorithm behaves by changing the objective function. Optimizing $\mathrm{J_O}$, for example, means that we try to find a maximum upper bound, while optimizing $\mathrm{J_P}$ means we try to find a maximum lower bound. We study these effects in Section 6.4.

Other than the different accuracy measures, the missing values are rather straight forward to implement. We need to keep track of the new sets, $E_{1,?}$, $E_{0,?}$, $E_{?,1}$, $E_{?,0}$, and $E_{?,?}$, for efficient computation of the different versions of Jaccard (cf. Section 4.3), but this does not alter the main principle. Similarly with real-valued data, having missing values only adds notational complexity – the concepts remain the same.

6 Experimental Evaluation of Algorithm's Properties

We now turn to the experimental evaluation of our algorithm. We divide this in three parts. The first part (this section) studies the various properties of our algorithm with both synthetic and real-world data. The next two sections compare our algorithm to other methods and present a real-world application of our algorithm, namely biological niche finding.

But before starting with the experiments, we explain a method for assessing the significance of the results based on randomizations and explain the data sets used.

An implementation of the ReReMi algorithm and the synthetic data generator are available online[3].

6.1 Assessing the significance with randomization methods. When mining the redescriptions, we compute various p-values in order to prune uninteresting rules. But these p-values are based on assumptions about the distribution of 0s and 1s in the (bucketed) data. Given the generality of our algorithm, we cannot assume the distributions to model exactly the underlying distribution of values. Hence, we also use property-preserving randomization methods. Such methods sample random matrices that share some property with the original matrix. The algorithm is then re-run using a random matrix as an input, and this process is repeated multiple times. If the results with random matrices contain multiple redescriptions that have same or higher accuracy than some redescription found from the original data, that redescription is deemed insignificant; otherwise it is significant (with respect to the property preserved by the randomization method).

For these experiments, we used two randomization methods. The first method permutes the matrix, preserving the values of the matrix. On symmetric matrices, we used a variation that preserves their symmetry: only the upper-right triangle was permuted, and the lower-left triangle was copied from there. The property of a matrix having the same values is not a very strong one. Hence, we also used a method to preserve the distribution of values in columns and rows of the matrix [13]. This method is called swap-randomization. While swap-randomization is in many ways stronger than permutation, the latter is used for two reasons. First, its use is suggested in [13]. Second, unlike swap-randomization, permutation can preserve symmetrical matrices.

6.2 The real-world data sets. For the real-world data, we used two basic data sets: `DBLP` and `Bio`. The former is obtained from the DBLP data base[4], and its entities are authors. The first matrix defines the conferences in which each of them has published, while the second defines other authors with whom each of them has published. The entities of the latter data set are spatial areas,

[3] `http://www.cs.helsinki.fi/u/galbrun/redescriptors/`
[4] `http://www.informatik.uni-trier.de/~ley/db`

Table 1: Statistics of the real-world data sets used in the experiments.

Data set	Description	Dimensions	Type	Density
$\texttt{DBLP}_\texttt{F}$	authors × conferences	6455 × 304	integer values	0.033
	authors × authors	6455 × 6455	integer values	0.002
$\texttt{DBLP}_\texttt{N}$	authors × conferences	2345 × 19	integer values	0.194
	authors × authors	2345 × 2345	integer values	0.005
$\texttt{DBLP}_\texttt{B}$	authors × conferences	2345 × 19	Boolean indicators	0.194
	authors × authors	2345 × 2345	Boolean indicators	0.005
Bio	locations × mammals	2575 × 194	Boolean indicators	0.166
	locations × climate	2575 × 48	real values	—
$\texttt{Bio}_\texttt{small}$	locations × mammals	1271 × 119	Boolean indicators	0.259
	locations × climate	1271 × 24	real values	—

that is, approximately 50 km squares over Europe[5]. The data itself is composed from two publicly available data bases: European mammal atlas [11] and Worldclim climate data [8]. The mammals data contains presence/absence information of mammal species in Europe, and the climate data contains minimum, average, and maximum monthly temperatures as well as average monthly precipitation. Notice that the mammals data is Boolean while the climate data is continuous.

We created variations of these two basic data sets for specific purposes. From the DBLP data we created three variations: $\texttt{DBLP}_\texttt{F}$, $\texttt{DBLP}_\texttt{N}$, and $\texttt{DBLP}_\texttt{B}$. The first, $\texttt{DBLP}_\texttt{F}$, is a big data set with 6455 authors and 304 conferences containing information on how many times each author has published in each conference and with each other author. The second, $\texttt{DBLP}_\texttt{N}$, also contains numerical information but is restricted to 19 hand-picked conferences[6] and 2345 authors. The third, $\texttt{DBLP}_\texttt{B}$, is like $\texttt{DBLP}_\texttt{N}$, but Boolean: every positive value of $\texttt{DBLP}_\texttt{N}$ is replaced with 1. This data is identical to the one used by Gallo et al. [4].

For the Bio data, we constructed two variations. The basic Bio data contains the whole data while smaller $\texttt{Bio}_\texttt{small}$ concentrates only on Northern Europe (specifically, areas between 50 and 71 degrees North). Also, it does not include monthly maximum or minimum temperatures, leaving only monthly average temperature and average precipitation.

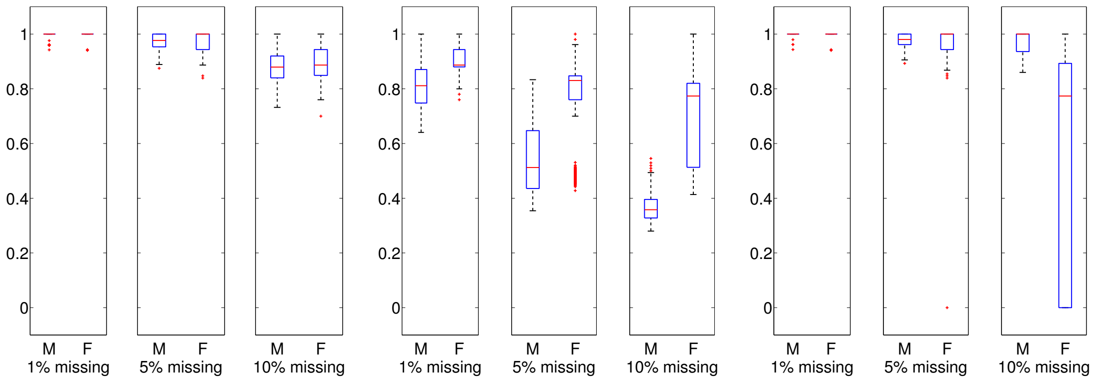
The statistics of the real-world data sets used in the experiments are presented in Table 1.

6.3 Finding planted redescriptions. To study the behaviour of our algorithm we first apply it to synthetic data. The idea is to generate data with planted redescriptions and check whether our algorithm is able to recover the redescriptions. Generating matrices such that, for example, no subset of the query forms an exact redescription, is not trivial.

To generate a pair of synthetic Boolean 500 × 10 matrices, we plant on both matrices one Boolean formula, either conjunction or disjunction over 3 variables with 50 supporting rows, such that the resulting redescription is exact. Then we add random noise of density between 0.01 and 0.1. The noise can either be conservative or destructive, leaving the redescription exact or not. A synthetic real-valued data matrix is then obtained replacing ones and zeros by values uniformly sampled from the intervals $[0.75, 1]$ and $[0, 0.25]$, respectively.

[5]Details of the grid can be found in www.fmnh.helsinki.fi/english/botany/afe/index.html.

[6]Namely WWW, SIGMOD, VLDB, ICDE, KDD, SDM, PKDD, ICDM, EDBT, PODS, SODA, FOCS, STOC, STACS, ICML, ECML, COLT, UAI and ICDT.



(a) Rejective Jaccard (b) Pessimistic Jaccard (c) Optimistic Jaccard

Figure 4: Distribution of redescription accuracies with missing values, conservative noise, and different accuracy measures. The three panels of subplots correspond to, from left to right, 1%, 5%, and 10% of missing values. In each panel, the left-hand box is the distribution of accuracies with missing data and using the corresponding accuracy measure (M) and the right-hand box is the distribution of accuracies of the same redescriptions computed over full data and using normal Jaccard (F).

Applied to some two hundred synthetic Boolean matrix pairs with conservative noise the algorithm managed to find all planted queries. In the case of destructive noise and fully Boolean data, the algorithm managed to find the planted queries in only about 40% of the data sets, but always found queries with higher accuracy than that of the planted redescription. The algorithm cannot be considered faulty is these cases: It behaved as assumed, finding the redescriptions with the highest accuracies.

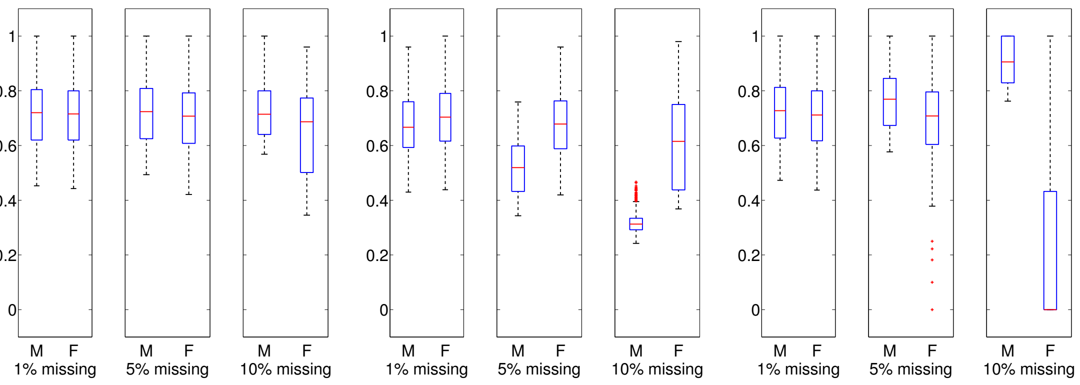
Applied to synthetic data sets where one matrix is Boolean and the other real-valued, both with conservative noise, the algorithm managed to find the planted redescriptions or equivalent in 76 cases out of 80. The planted redescriptions that were not found had contributions below the acceptance threshold. Thus, the algorithm again worked as assumed.

6.4 Experiments with missing values. As for the full data, our first experiment with missing values is on synthetic data. We removed values from synthetic Boolean matrices with planted redescriptions (cf. Section 6.3) uniformly at random. We removed 1%, 5%, and 10% of the values (zeroes or ones) and replaced them with markers for missing values. Then we mined them with our algorithm and checked how well the planted redescriptions were discovered.

For the planted redescriptions with conservative noise, the planted redescription was found in 97% of the cases. With increasing ratio of missing values the algorithm more often did not found the complete planted redescription but rather one or several fragments of high accuracies.

We report the distributions of the accuracies in Figures 4 (conservative noise) and 5 (destructive noise). The results compare the different variations of the Jaccard (rejective, pessimistic, and optimistic) as well as their estimates to the true Jaccard computed over the full data (i.e. with no missing values).

In both figures, it is clear that the rejective Jaccard is the best. With conservative noise (Figure 4(a)) missing values have hardly any effect, and while the destructive noise reduces the quality (Figure 5(a)), different levels of missing values have very small effect and the estimated



(a) Rejective Jaccard (b) Pessimistic Jaccard (c) Optimistic Jaccard

Figure 5: Distribution of redescription accuracies with missing values, destructive noise, and different accuracy measures. The three panels of subplots correspond to, from left to right, 1%, 5%, and 10% of missing values. In each panel, the left-hand box is the distribution of accuracies with missing data and using the corresponding accuracy measure (M) and the right-hand box is the distribution of accuracies of the same redescriptions computed over full data and using normal Jaccard (F).

Jaccards are close to the true Jaccards. Pessimistic Jaccard is the second-best option. While its estimates look bad (Figures 4(b) and 5(b)), the true Jaccards are almost as good as those with rejective Jaccard. One has to remember that the estimate of the pessimistic Jaccard is only the lower bound, and therefore having very low estimate is not bad in itself. The last option, optimistic Jaccard (Figures 4(c) and 5(c)) has the worst behaviour with higher level of missing values (although with 1% and 5% of missing values the true Jaccards are slightly better than those of pessimistic Jaccard).

As the missing values were evenly scattered over the data, the good behaviour of rejective Jaccard was no surprise.

Next, we experimented with the $\texttt{Bio}_{\texttt{small}}$data set, generating copies with 1% or 5% of randomly chosen values removed. For each setting, we generated 10 copies and mined them. We compared the queries found in the data with missing values to thoses found in the full data. We only used rejective Jaccard due to its good performance with synthetic data.

The results are reported in Figure 6. With 1% of missing values the results are almost as good as the results with full data, but with 5% of missing values the true Jaccards are somewhat worse, even if the estimates are still high.

6.5 From Boolean to Numerical: the DBLP data. The purpose of this experiment is to study the behaviour of our algorithm with two versions of the same data, one Boolean and one numerical. For this we used the two variants of the `DBLP` data: $\texttt{DBLP}_{\texttt{B}}$ and $\texttt{DBLP}_{\texttt{N}}$.

Some results with $\texttt{DBLP}_{\texttt{N}}$ are presented in Table 2, while some example results with $\texttt{DBLP}_{\texttt{B}}$ are in Table 3. In all tables, J stands for the Jaccard, supp is the support of the redescription, and the *p*-value is computed as in Section 5.1.

For these experiments, we disabled negations to allow later comparison (see Section 7.2). Comparing the two tables, it is obvious that $\texttt{DBLP}_{\texttt{N}}$ contains more information, allowing ReReMi to find more accurate redescriptions (the best Jaccard in Table 2 is 0.625). But the rules are also

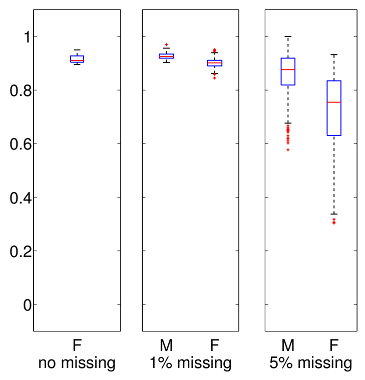


Figure 6: Results with $\texttt{Bio}_{\texttt{small}}$ data set having missing values. Left: distribution of similarities with all values known. Middle: distribution of similarities with 1% of values missing. Right: distribution of similarities with 5% of values missing. In middle and right panels, left-hand box is rejective Jaccard with missing values (M), and the right-hand box is the normal Jaccard of the same redescriptions computed over full data (F).

Table 2: Example results of ReReMi with $\texttt{DBLP}_{\texttt{N}}$.

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	J	supp	p-value
(1) $[1.0 \leq$ STOC $\leq 6.0] \wedge [8.0 \leq$ COLT$]$	$[1.0 \leq$ D.P. Helmbold$] \vee [1.0 \leq$ M. Frazier$]$ $\vee [2.0 \leq$ N. Cesa-Bianchi $\leq 2.0]$	0.625	15	0.000
(2) $[1.0 \leq$ VLDB $\leq 18.0] \wedge [2.0 \leq$ ICDM$]$ $\wedge [1.0 \leq$ SDM $\leq 5.0]$ $\wedge [3.0 \leq$ ICDE$]$	$\big(([5.0 \leq$ W. Wang$] \vee [1.0 \leq$ J. Pei$])$ $\wedge [2.0 \leq$ P.S. Yu$]\big) \vee [6.0 \leq$ G. Das $\leq 6.0]$	0.600	12	0.000
(3) $[5.0 \leq$ COLT$]$	$[2.0 \leq$ P.L. Bartlett$] \vee [1.0 \leq$ M.K. Warmuth$]$ $\vee [1.0 \leq$ E.B. Kinber$] \vee [1.0 \leq$ S.A. Goldman$]$	0.472	42	0.000

Table 3: Example results of ReReMi with $\texttt{DBLP}_{\texttt{B}}$.

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	J	supp	p-value
(1) STOC $\wedge$ COLT $\wedge$ ICML	Y. Freund $\vee$ N. Littlestone $\vee$ P.M. Long $\vee$ S. Kwek	0.500	21	0.000
(2) VLDB $\wedge$ ICDM $\wedge$ SDM $\wedge$ SIGMOD	(J. Han $\wedge$ P.S. Yu)$\vee$ C.-R. Lin $\vee$ S. Lonardi	0.444	16	0.000
(3) ICDM $\wedge$ SDM $\wedge$ KDD	J. Lin $\vee$ I.S. Dhillon $\vee$ P.S. Yu $\vee$ V. Kumar	0.338	44	0.000
(4) FOCS $\wedge$ SODA $\wedge$ STOC	B. Awerbuch $\vee$ S. Khanna $\vee$ R.E. Tarjan $\vee$ N. Alon	0.324	158	0.000

more specific than those in Table 3, with small support and often requiring multiple publications in the same conference, making them possibly harder to interpret than those of Table 3. Yet, in both cases the algorithm correctly identifies subfields of computer science and well-known authors from those fields.

6.6 Approximating the initial pairs. To study the effects of approximating the initial pairs, we took the climate data of the $\texttt{Bio}_{\texttt{small}}$ data set and divided it into two: the monthly average temperatures and monthly average rainfall. This gave us two real-valued data sets with 12 variables each. We call this data `Climate`.

This seemingly small data set is already hard for exhaustive initial pairs selection due to the nature of the data: almost each data point contains different values. Consequently, the exhaustive search was not able to finish in two days.

Instead of exhaustive search we used the binning-based initial-pairs searching method explained in Section 5.3. We tried different bin sizes, ranging from 10 to 100. We compared this with fully pre-bucketing approach, bin size again ranging from 10 to 100.

The results were clear. There was very little variation between different bin sizes for initial pairs in terms of accuracy: the smaller bins gave overall slightly better results, but all sizes had top-20 redescriptions with accuracy clearly above 0.8 with bests above 0.9. Pre-bucketing approaches, on the other hand, were clearly inferior, with the best results (obtained with bin size 10) being below 0.7.

In terms of speed, the pre-bucketed approaches were the fastest, with even the slowest (bin size 10) taking less than 4 minutes. The fastest method with initial bins (bin size 100) took roughly 10 times longer, 38 minutes. The slowest method (initial bin size 10) took one day and 24 minutes. In all the methods with initial bins, the time differences are explained by the time spent on constructing the initial pairs; after that, they all took roughly the same time.

The results were as expected. Using binning only for initial pairs (as opposed to working with pre-bucketed data) gave much better results, but also took more time. It is, however, noteworthy that increasing the bin size used for selecting the initial pairs did not have considerable effect on the quality. Apparently, the algorithm was able to overcome the possibly weaker initial pairs in later phases.

6.7 Running times. Does our algorithm scale? The data sets we used were not extremely large (although $\texttt{DBLP}_{\texttt{F}}$ is already of considerable size), but we feel confident to say that our algorithm is scaling reasonably well. All experiments were conducted in a single core of an 8 core Intel Xeon 2.8 GHz processor and with 32 GB of memory.

The statistics of the running times on the different data sets are presented in Table 4. There we can see that original data takes much more time to run than randomized data. This is because with randomized data, there are less potential redescriptions, and the algorithm can prune the search space much faster. The longest time required, 1 hour with $\texttt{DBLP}_{\texttt{F}}$, is still very good, especially when one remembers that the algorithm needs to be run only once for the original data.

To evaluate more precisely the evolution of our algorithm's running time with respect to the size of the data, we generated synthetic matrices of increasing sizes. As we just mentionned, the running times on random data is not representative. Therefore, we need to insure that some redescriptions are present in the synthetic data. Thus, we constructed diagonal block matrices, where the blocks on the diagonal are matrices obtained as described in Section 6.3 while off-diagonal blocks are filled with randon noise. We used original matrices of size 500×10 for the fully Boolean setting and of size 250×10 for the setting where one side is Boolean and the other real-valued. We

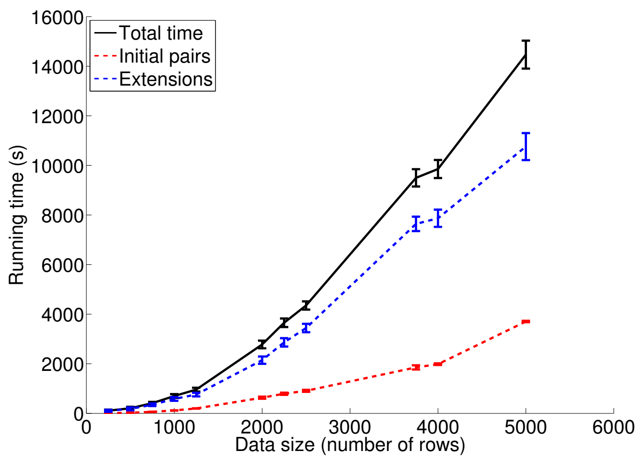


Figure 7: Running times for mining redescriptions on synthetic data of increasing sizes with one side Boolean and the other real-valued.

Table 4: Running times for the different data sets. For mean time, '—' denotes that there was only one data set.

Data Set	mean	max
$\mathtt{DBLP}_\mathtt{F}$	—	60min
$\mathtt{DBLP}_\mathtt{F}$ permuted	5min 16s	6min 32s
$\mathtt{DBLP}_\mathtt{F}$ swap	16min	18min
$\mathtt{DBLP}_\mathtt{N}$	—	7min
$\mathtt{DBLP}_\mathtt{B}$	—	1min
`Bio`	—	10min
`Bio` permuted and swap	6min 48s	10min 30s

let our algorithm run on ten such synthetic data sets for each studied size. The algorithm consists of two major steps: firstly, computing singleton redescriptions by trying out all pairs of variables and secondly extending the best pairs in turn. In the case of fully Boolean data, the computation of initial pairs clearly dominates as the data grows larger, since it is quadratic in the number of columns while finding extension is linear. The running times remain reasonable even for very large data sets.

When real-valued data is involved, the running times are much greater, as expected. Therefore we had to restrict the experiments to smaller data sets. The running times for that experiment are displayed in Figure 7. The increase in running times is due to the fact that extending a redescription is no longer linear and the search for initial pairs also has increased complexity.

7 Experimental Comparison Against Other Methods

With the previous sections results showing that our algorithm has solid performance, we can now turn to comparing our algorithm to other methods.

7.1 Comparison to association rule mining. The first experiment with real-world data mirrors the experiments with synthetic data: the task is to study how well our algorithm finds redescriptions from the data. But as we cannot know all redescriptions present in the real-world data, we narrow our scope to monotone conjunctive redescriptions from Boolean data. These redescriptions are simply bi-directional association rules, and hence can be found by mining all frequent itemsets from both data matrices and using the itemset pairs as redescriptions. This gives us the ground truth, against which we can compare our algorithm.

For these experiments we used the $\texttt{DBLP}_\texttt{B}$ data (see Section 6.2). We used the Eclat frequent itemset miner [24]. The redescriptions were generated as follows: first, all closed frequent itemsets with support greater than 5 were mined for both data sets. The itemsets were then combined into redescriptions. Only those with accuracy greater than 0.1, support above 10 but below 100 (inclusive), and p-value higher than 0.01 were retained. The same parameters were set to ReReMi, and it was only allowed to find monotone conjunctive redescriptions.

The best four redescriptions found using Eclat had a Jaccard similarity between 0.366 and 0.333. ReReMi found exactly the same redescriptions. After these four redescriptions, the Eclat approach found several redundant redescriptions: they were minor variations of the first four redescriptions. ReReMi did not report these redundant redescriptions, which we consider a positive feature – the user should not be overwhelmed by the quantity of results.

The next non-redundant redescription found by Eclat approach was also found by ReReMi. The same trend continued throughout the results: Eclat approach found several thousands of redescriptions, but most of them were redundant.

Applying Eclat on swapped and permuted randomized copies of the original data (500 pairs of matrices for each method) following the same approach did not return any redescription. Therefore all original redescriptions are considered significant with respect to these null hypotheses.

While these experiments cannot guarantee that ReReMi will always find the best redescriptions, they suggest that it is able to find most of the important ones.

7.2 Comparison to the work of Gallo et al. As we used the same data set as Gallo et al. [4], $\texttt{DBLP}_\texttt{B}$, we compared the results obtained with ReReMi to theirs. Following [4], we did not allow negations. The example results were already presented in Table 3.

The results obtained by ReReMi had higher Jaccard similarity than those obtained by Gallo et al.: the highest accuracy they report is 0.35, while ReReMi returns a redescription with accuracy 0.5, and 9 redescriptions have accuracy above 0.35.

An effect of the beam search is illustrated by the following for example. The results presented by Gallo et al. contain the following redescription of accuracy 0.30:

$$\text{SDM} \wedge \text{ICDE} \qquad \text{P. S. Yu} \vee \text{J. Lin} \vee \text{M. Schubert} \vee \text{Y. Ma.}$$

Our results do not contain this redescription, but

$$\text{SDM} \wedge \text{ICDM} \wedge \text{KDD} \qquad \text{P. S. Yu} \vee \text{J. Lin} \vee \text{I. S. Dhillon} \vee \text{V. Kumar,}$$

instead, with accuracy 0.34. Indeed, when starting from the intial pair (sdm, p. s. Yu), keeping at each step only the best candidate does not allow to find this redescription as it is obtained from a candidate ranked lower during the extension process.

While allowing for a better exploration of the search space, our algorithm required only a third of Greedy's running time (3 min) with the same data. This shows that the proposed algorithm is faster yet more exhaustive on Boolean redescription mining than the Greedy algorithm.

Table 5: Example results of CARTwheels with $\texttt{DBLP}_\texttt{B}$.

$q_\mathbf{L}$	$q_\mathbf{R}$	J	supp	p-value
(1) (STOC $\wedge \neg$ FOCS) $\vee \neg$ STOC	B. Dageville $\vee$ ($\neg$ B. Dageville $\wedge \neg$ A. Wigderson)	0.736	1673	0.011
(2) (STOC $\wedge \neg$ FOCS) $\vee \neg$ STOC	T. Grust $\vee$ ($\neg$ T. Grust $\wedge \neg$ A. Wigderson)	0.736	1673	0.011
(3) EDBT $\vee$ ($\neg$ EDBT $\wedge \neg$ STOC)	(P. Datta $\wedge$ P. Langley) $\vee$ ($\neg$ P. Datta $\wedge \neg$ A. Wigderson)	0.693	1577	0.021
(4) ICDM $\vee$ ($\neg$ ICDM $\wedge \neg$ STOC)	(C. Olston $\wedge \neg$ C. Chekuri) $\vee$ ($\neg$ C. Olston $\wedge \neg$ A. Wigderson)	0.691	1570	0.017
(5) PKDD $\vee$ ($\neg$ PKDD $\wedge \neg$ STOC)	T. Grust $\vee$ ($\neg$ T. Grust $\wedge \neg$ A. Wigderson)	0.689	1567	0.019

Otherwise the results are similar; both algorithms identify sets of conferences from different fields of computer science together with well-known authors from those fields. Both algorithms are also able to identify interdisciplinary researchers, say, theoretical machine learners who publish in both machine learning and theoretical conferences (row 1 of Table 3).

7.3 Comparison to CARTwheels and pre-bucketing numerical data. We now turn to another algorithm for mining redescriptions, CARTwheels [17]. We used the implementation of CARTwheels provided by the authors. Because of this, we do not have any control over the results reported by CARTwheels (e.g. minimum support, type of queries, p-values).

Boolean data. First, we tried CARTwheels with $\texttt{DBLP}_\texttt{B}$. The algorithm returned in total 35 redescriptions before running out of the available 32 GB of memory. Of these, only 5 were retained after removing rules with p-value higher than 0.05. All of the remaining redescriptions had p-values between 0.0111 and 0.02, making them insignificant on the highest significance level (99%). The rules also covered almost the whole data, having at least 1567 (of 2345) rows in the support. This high support is a consequence of using mostly negated variables. Results are reported in Table 5.

As can be seen from Table 5, the results have high accuracy, which is not surprising, given their high support. Results also have many negations, making them less interesting. We let ReReMi find results with negations, too, and while we were not able to find results with as high accuracy (results omitted), they all had p-values essentially 0. We conclude that while CARTwheels finds more accurate redescriptions, they are somewhat insignificant, and less interesting than the ones found by ReReMi.

Pre-bucketing numerical data. In this section, we compare our on-the-fly bucketing approach to CARTwheels with bucketing as a pre-processing step.

To bucket the data, one has to select the bucketing method. For a fair comparison, we used three different methods with a number of buckets per variable varying between 10 and 150, ran CARTwheels for all of these configurations, and selected the best results. The three methods were (1) buckets of equal width, where the range of the values in a column was divided into equally long buckets; (2) buckets of equal height, where each bucket contained approximately equally many entities; and (3) segmentation, where the entities were separated into segments (i.e. buckets) minimizing the sum-of-square distances to the segment's mean (the segmentation was obtained using Bellman's algorithm [1]).

The CARTwheels algorithm was unable to handle the full `Bio` data, so we used $\texttt{Bio}_\texttt{small}$ instead.

The results are reported in Table 6. The aim of this experiment is to study how good accuracies can be obtained with pre-processed buckets compared to ReReMi.

The first results are the five best (with respect to the accuracy) from CARTwheels using

Table 6: Example redescriptions from $\texttt{Bio}_{\texttt{small}}$ data: $t_X^{\min}, t_X^{\max}$, and t_X^{avg} stand for minimum, maximum, and average temperature of month X in degrees Celsius, and p_X^{avg} stands for average precipitation of month X in millimeters.

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	J	supp	p-value
	CARTwheels			
(1) (European Pine Vole ∧ European Pine Marten) ∨ (¬European Pine Vole)	$([57.5 \le p_{\text{Nov}}^{\text{avg}} \le 62.706] \wedge \neg[75.03 \le p_{\text{Jun}}^{\text{avg}} \le 82.6]) \vee (\neg[57.5 \le p_{\text{Nov}}^{\text{avg}} \le 62.706])$	0.980	1244	0.007
(2) (Argali ∧ Hazel Dormouse) ∨ (¬Argali)	$([78.291 \le p_{\text{Aug}}^{\text{avg}} \le 82.282] \wedge \neg[0.8245 \le t_{\text{Nov}}^{\text{avg}} \le 2.2357]) \vee (\neg[78.291 \le p_{\text{Aug}}^{\text{avg}} \le 82.282])$	0.974	1237	0.158
(3) (Brown Bear ∧¬Arctic Fox) ∨ (¬Brown Bear)	$([43.163 \le p_{\text{Dec}}^{\text{avg}} \le 46.92] \wedge \neg[t_{\text{Jul}}^{\text{avg}} \le 10.427]) \vee (\neg[43.163 \le p_{\text{Dec}}^{\text{avg}} \le 46.92])$	0.974	1237	0.074
(4) (Nathusius' Pipistrelle ∧ ¬Mediterranean Water Shrew) ∨ (¬Nathusius' Pipistrelle)	$([40.896 \le p_{\text{Nov}}^{\text{avg}} \le 46.743] \wedge \neg[16.32 \le t_{\text{Jul}}^{\text{avg}} \le 16.751]) \vee (\neg[40.896 \le p_{\text{Nov}}^{\text{avg}} \le 46.743])$	0.974	1235	0.000
(5) (Brown Bear ∧¬Arctic Fox) ∨ (¬Brown Bear)	$([60.13 \le p_{\text{Apr}}^{\text{avg}} \le 74.305] \wedge \neg[82.6 \le p_{\text{Jun}}^{\text{avg}}]) \vee (\neg[60.13 \le p_{\text{Apr}}^{\text{avg}} \le 74.305])$	0.964	1224	0.337
	CARTwheels pruned			
(1) (Eurasian Least Shrew) ∨ (¬Eurasian Least Shrew ∧¬House mouse)	$([75.154 \le p_{\text{Aug}}^{\text{avg}} \le 78.291] \wedge \neg[1.622 \le t_{\text{Feb}}^{\text{avg}} \le 3.44]) \vee (\neg[75.154 \le p_{\text{Aug}}^{\text{avg}} \le 78.291] \wedge \neg[5.488 \le t_{\text{Mar}}^{\text{avg}}])$	0.832	944	0.000
(2) (Wisent) ∨ (¬Wisent ∧¬House mouse)	$([34.467 \le p_{\text{Feb}}^{\text{avg}} \le 41.402] \wedge \neg[5.488 \le t_{\text{Mar}}^{\text{avg}}]) \vee (\neg[34.467 \le p_{\text{Feb}}^{\text{avg}} \le 41.402] \wedge \neg[3.15 \le t_{\text{Jan}}^{\text{avg}}])$	0.824	940	0.000
(3) (Red-necked Wallaby) ∨ (¬Red-necked Wallaby ∧ House mouse)	$([6.12 \le t_{\text{Nov}}^{\text{avg}}] \wedge \neg[14.028 \le t_{\text{Sep}}^{\text{avg}}]) \vee (\neg[6.12 \le t_{\text{Nov}}^{\text{avg}}] \wedge [1.622 \le t_{\text{Feb}}^{\text{avg}} \le 3.44])$	0.522	175	0.000
(4) (European Mole ∧¬House mouse) ∨ (¬European Mole ∧ House mouse)	$([5.0439 \le t_{\text{Nov}}^{\text{avg}} \le 6.12] \wedge \neg[1.691 \le t_{\text{Mar}}^{\text{avg}} \le 2.9013]) \vee (\neg[5.0439 \le t_{\text{Nov}}^{\text{avg}} \le 6.12] \wedge [6.12 \le t_{\text{Nov}}^{\text{avg}}])$	0.674	225	0.000
(5) (European Mole ∧ House mouse) ∨ (¬European Mole ∧¬House mouse)	$([5.0439 \le t_{\text{Nov}}^{\text{avg}} \le 6.12] \wedge [1.691 \le t_{\text{Mar}}^{\text{avg}} \le 2.9013]) \vee (\neg[5.0439 \le t_{\text{Nov}}^{\text{avg}} \le 6.12] \wedge \neg[6.12 \le t_{\text{Nov}}^{\text{avg}}])$	0.896	937	0.000
	ReReMi			
(1) ¬ House mouse	$(\neg[3.4133 \le t_{\text{Mar}}^{\text{avg}}] \vee [1.790 \le t_{\text{Aug}}^{\text{avg}}] \vee [6.5917 \le p_{\text{May}}^{\text{avg}} \le 6.6083]) \wedge \neg[2.3667 \le t_{\text{Jan}}^{\text{avg}} \le 3.0667]$	0.948065	931	0.000
(2) ¬ House mouse	$([t_{\text{Feb}}^{\text{avg}} \le 2.20] \wedge [t_{\text{Mar}}^{\text{avg}} \le 3.41]) \vee [1.790 \le t_{\text{Aug}}^{\text{avg}}] \vee [6.5917 \le p_{\text{May}}^{\text{avg}} \le 6.6083]$	0.947101	931	0.000
(3) ¬ House mouse ∧ ¬ Eastern gray squirrel	$[1.8375 \le t_{\text{Jan}}^{\text{avg}} \le 1.90] \vee \neg[0.25 \le t_{\text{Feb}}^{\text{avg}}]$	0.934359	911	0.000
(4) ¬ House mouse ∧¬ Raccoon ∧ ¬ Eastern gray squirrel	$([5.37 \le t_{\text{Apr}}^{\text{avg}} \le 6.1357] \vee [5.04 \le t_{\text{Nov}}^{\text{avg}} \le 5.10] \vee \neg[1.25 \le t_{\text{Dec}}^{\text{avg}}]) \wedge [3.775 \le t_{\text{Mar}}^{\text{avg}}]$	0.925965	888	0.000
(5) ¬ Grey Red-Backed Vole ∧ ¬ Wolverine ∧¬ Brown Bear ∧ ¬ Siberian Flying Squirrel	$(\neg[t_{\text{Jan}}^{\text{avg}} \le -6.4615] \wedge [-5.70 \le t_{\text{Feb}}^{\text{avg}}]) \vee [t_{\text{Jul}}^{\text{avg}} \le 4.6267] \vee [-4.9133 \le p_{\text{Dec}}^{\text{avg}} \le -4.871]$	0.923681	823	0.000
	ReReMi bucketed			
(1) (Striped Field Mouse ∨ House mouse)∧ Wood mouse	$[5.925 \le t_{\text{Apr}}^{\text{avg}} \le 7.0] \vee [7.0 \le t_{\text{Apr}}^{\text{avg}} \le 7.9077] \vee [7.9077 \le t_{\text{Apr}}^{\text{avg}} \le 8.46] \vee [8.46 \le t_{\text{Apr}}^{\text{avg}}]$	0.807	442	0.000
(2) House mouse	$[-0.21534 \le t_{\text{Feb}}^{\text{avg}} \le 1.622] \vee [1.622 \le t_{\text{Feb}}^{\text{avg}} \le 3.44] \vee [3.44 \le t_{\text{Feb}}^{\text{avg}}]$	0.778	308	0.000
(3) European Rabbit	$[1.691 \le t_{\text{Mar}}^{\text{avg}} \le 2.9013] \vee [2.9013 \le t_{\text{Mar}}^{\text{avg}} \le 3.9685] \vee [3.9685 \le t_{\text{Mar}}^{\text{avg}} \le 5.488] \vee [5.488 \le t_{\text{Mar}}^{\text{avg}}]$	0.774	441	0.000
(4) House mouse	$[2.9013 \le t_{\text{Mar}}^{\text{avg}} \le 3.9685] \vee [3.9685 \le t_{\text{Mar}}^{\text{avg}} \le 5.488] \vee [5.488 \le t_{\text{Mar}}^{\text{avg}}]$	0.773	307	0.000
(5) North American Beaver ∧ Reindeer	$[t_{\text{Jan}}^{\text{avg}} \le -11.9] \wedge [14.994 \le t_{\text{Jul}}^{\text{avg}} \le 15.785] \wedge [7.1073 \le t_{\text{Sep}}^{\text{avg}} \le 8.5667] \wedge [70.333 \le p_{\text{Jul}}^{\text{avg}} \le 73.385]$	0.700	7	0.000

10 buckets of approximately equal number of entities (this method produced the best overall results, although 10 segments gave very similar results). The redescriptions have again very high accuracy, and, again, they cover almost all of the studied area. Furthermore, two of them are clearly insignificant and one is not very significant, according to the p-values. Results of this type are rarely of any interest for users, as they do not convey any interesting information.

We pruned out results that had too high support (leaving less than 250 entities uncovered) or too high p-value. CARTwheels can still return rather accurate redescriptions, but the quality drops quickly after the first ones.

The last results in this experiment are from ReReMi. As CARTwheels obtained almost all of its results using negations, we allowed also ReReMi to use negations. As can be seen, ReReMi positions itself between non-pruned and pruned CARTwheels. But unlike the non-pruned CARTwheels, ReReMi does not return insignificant results.

We also experimented with bucketed data and Boolean ReReMi. The results were similar to those with pruned CARTwheels, but without the quick drop in accuracy. Also, they were considerably worse than the results with ReReMi using on-the-fly bucketing.

We conclude that with similar constraints, the on-the-fly bucketing of ReReMi gives the best results, although in this application, not allowing negations could be considered more reasonable an option.

8 Real-world application: Finding bioclimatic envelopes.

Our final experiment is a real-world application of finding bioclimatic envelopes.

ReReMi was run on full `Bio` data for a maximum of 100 initial pairs, minimum score for the initial pairs 0.2, minimum support 15, minimum number of uncovered entities 500, and minimum contribution 3, disallowing negated variables. This was done because the negated variables can lead to counterintuitive redescriptions in this type of application. The algorithm found 69 redescriptions within these constraints. Again, we regard it positively that our algorithm returns only a reasonable amount of results.

The data was randomized using swap-randomization and permutation. With both methods, 500 random matrices were generated, and in both cases the best redescription had lower accuracy than the lowest original accuracy. Hence, all redescriptions were considered significant with respect to these null hypotheses. The algorithm processed the full data in about 13 minutes.

Some of the results are displayed in Table 7. The redescriptions have varying support sizes: some cover only a small part of the data, while others cover almost the whole data. Yet, they all have very high accuracy. The first two redescriptions cover exactly the same area. They represent the Svalbard archipelago (see Fig. 8(a)). The climate in Svalbard is so different from other areas that it allows multiple ways to define it, causing multiple redescriptions. The fourth redescription has only European Elk on the left hand side, but the right hand side is more complex, characterizing very accurately the environment in Scandinavia and Baltia (Fig. 8(b)), the area occupied by the European Elk. The remaining redescriptions are more complex. The fifth redescription (Fig. 8(c)) covers Northern and Central Europe, while the last covers only Central Europe (Fig. 8(d)).

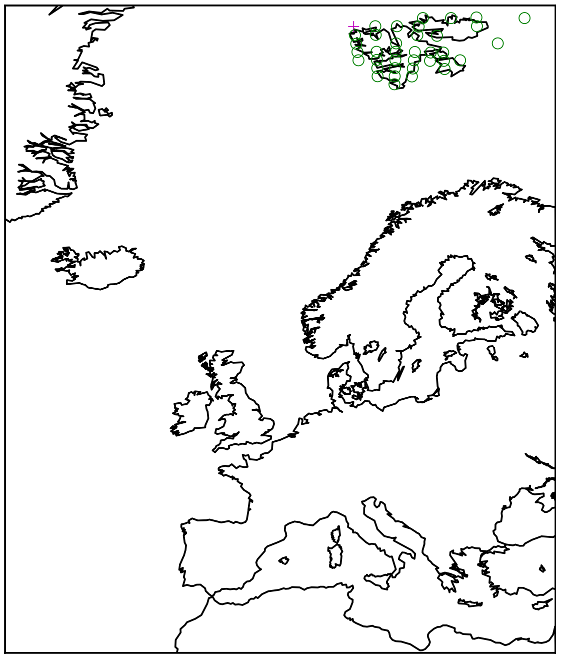
We point out that while the results of [5] are superficially similar, the differences in the methods used and the goals pursued make the results incomparable.

9 Conclusions

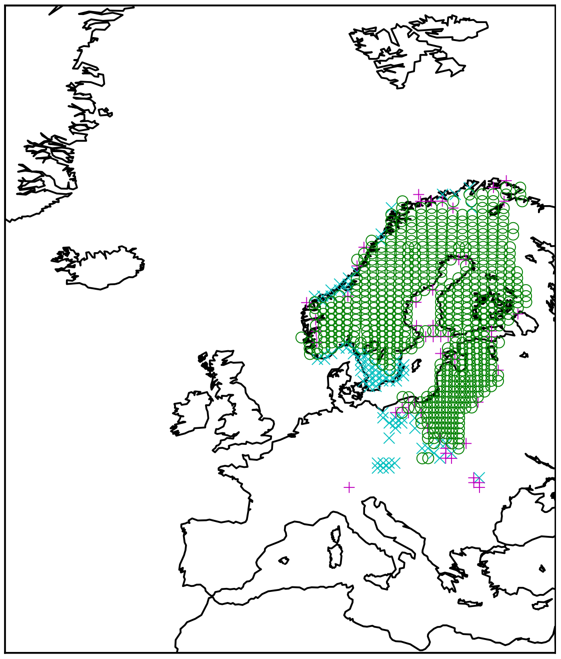
We have presented a new algorithm to mine redescriptions from real-valued data. Unlike previous algorithms, ours does not require any pre-processing when used with non-Boolean data. It is based on a beam-search type of method. We have shown with both synthetic and real-world data sets

Table 7: Example redescriptions from `Bio` data: $t_X^{\min}$, $t_X^{\max}$, and t_X^{avg} stand for minimum, maximum, and average temperature of month X in degrees Celsius, and p_X^{avg} stands for average precipitation of month X in millimeters.

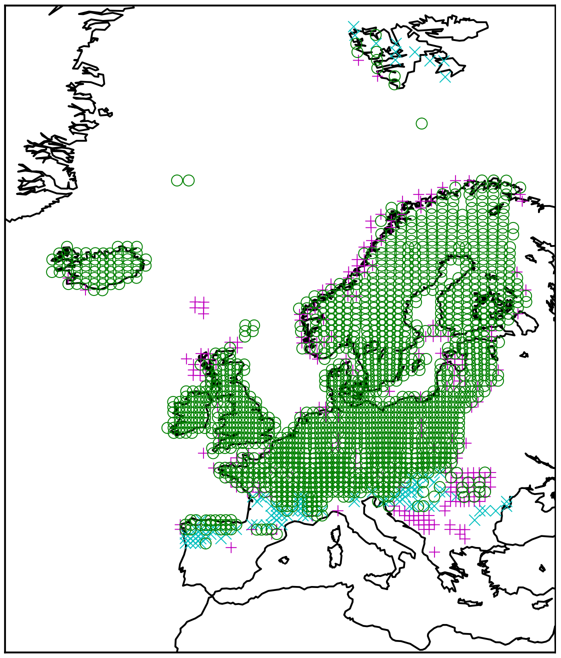
	$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	J	supp	p-value
(1)	Polar Bear	$[-7.0727 \le t^{\text{avg}}_{\text{May}} \le -3.375]$	0.973	36	0.000
(2)	Polar Bear	$[-16.694 \le t^{\text{avg}}_{\text{Mar}} \le -11.462]$	0.973	36	0.000
(3)	Bank Vole $\lor$ Northern Red-backed Vole $\lor$ Steppe Mouse $\lor$ Harbor Seal	$\big(([11.20 \le t^{\max}_{\text{Jul}} \le 15.40] \lor [13.10 \le t^{\max}_{\text{Aug}} \le 27.40]) \land [42.5 \le p^{\text{avg}}_{\text{Jul}}]\big) \lor [17.10 \le t^{\max}_{\text{Apr}} \le 17.50]$	0.818	1679	0.000
(4)	European Elk	$([-9.80 \le t^{\max}_{\text{Feb}} \le 0.40] \land [12.20 \le t^{\max}_{\text{Jul}} \le 24.60] \land [56.852 \le p^{\text{avg}}_{\text{Aug}} \le 136.46]) \lor [183.27 \le p^{\text{avg}}_{\text{Sep}} \le 238.78]$	0.814	582	0.000
(5)	Arctic Fox $\lor$ Stoat	$\big(([2.60 \le t^{\max}_{\text{Jun}} \le 8.50] \lor [7.20 \le t^{\max}_{\text{Sep}} \le 22.20]) \land [36.667 \le p^{\text{avg}}_{\text{Aug}}]\big) \lor [21.133 \le t^{\text{avg}}_{\text{Jul}} \le 21.20]$	0.813	1477	0.000
(6)	Greater White-toothed Shrew $\land$ Egyptian Mongoose	$([15.60 \le t^{\min}_{\text{Aug}} \le 19.00] \land [1.625 \le p^{\text{avg}}_{\text{Aug}} \le 7.4444] \land [66.222 \le p^{\text{avg}}_{\text{Dec}} \le 137.27]) \lor [19.083 \le t^{\text{avg}}_{\text{Oct}} \le 19.10]$	0.790	49	0.000
(7)	House Mouse $\lor$ Caucasian Squirrel $\lor$ Marbled Polecat	$\big(([3.50 \le t^{\max}_{\text{Jan}}] \land [4.40 \le t^{\max}_{\text{Feb}}]) \lor [3.5071 \le t^{\text{avg}}_{\text{Mar}} \le 4.1727]\big) \land [3.30 \le t^{\max}_{\text{Dec}}]$	0.765	1034	0.000
(8)	Southwestern Water Vole $\lor$ Azores Noctule $\lor$ Common Noctule $\lor$ Blind Mole	$([17.10 \le t^{\max}_{\text{Mar}}] \lor [19.30 \le t^{\max}_{\text{Aug}} \le 26.90] \lor [12.40 \le t^{\max}_{\text{Nov}} \le 14.50]) \land [14.60 \le t^{\max}_{\text{Sep}}]$	0.697	1072	0.000
(9)	Brown Long-eared Bat	$([13.70 \le t^{\max}_{\text{Sep}} \le 22.70] \lor [8.4111 \le t^{\text{avg}}_{\text{Nov}} \le 8.6444]) \land [17.30 \le t^{\max}_{\text{Jul}} \le 28.40] \land [-8.15 \le t^{\text{avg}}_{\text{Jan}} \le 6.0083]$	0.693	963	0.000
(10)	Harvest Mouse $\land$ European Mole	$[-0.30 \le t^{\min}_{\text{Apr}} \le 8.70] \land [19.40 \le t^{\max}_{\text{Aug}} \le 27.20] \land [45.417 \le p^{\text{avg}}_{\text{Jun}}] \land [48.75 \le p^{\text{avg}}_{\text{Aug}} \le 126.33]$	0.677	774	0.000
(11)	(Serotine Bat $\lor$ Lesser Mole Rat) $\land$ European Mole	$[19.70 \le t^{\max}_{\text{Jul}}] \land [16.90 \le t^{\max}_{\text{Sep}} \le 23.70] \land [43.111 \le p^{\text{avg}}_{\text{Jul}} \le 149.5] \land [31.875 \le p^{\text{avg}}_{\text{Oct}} \le 119.5]$	0.634	664	0.000
(12)	Wood Mouse $\land$ Natterer's Bat $\land$ Eurasian Pygmy Shrew	$([3.20 \le t^{\max}_{\text{Mar}} \le 14.50] \land [17.30 \le t^{\max}_{\text{Aug}} \le 25.20] \land [14.90 \le t^{\max}_{\text{Sep}} \le 22.80]) \lor [19.60 \le t^{\text{avg}}_{\text{Jul}} \le 19.956]$	0.623	681	0.000



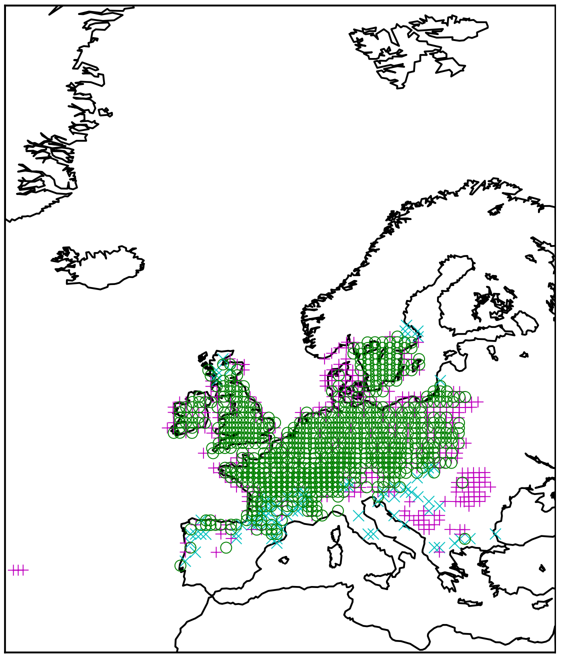
(a)



(b)



(c)



(d)

Figure 8: Support of redescriptions when mining `Bio` data. (a) Row 1, (b) row 4, (c) row 5, and (d) row 12 in Table 7. Green circles, cyan plus crosses and magenta plus signs respectively indicate areas where both queries hold, only the left query holds and only the right query holds.

that our algorithm performs better than its peers. In particular, the experiments show the benefits of on-the-fly bucketing against pre-processing.

The non-Boolean redescription mining has many applications in various fields of science, of which niche-finding is the one we have studied here. One of the most prominent future works would be to collaborate with biologists and ecologists and explore the true value of redescription mining in finding the bioclimate envelopes. Also other fields should be considered. Medical data describing patients where the matrices would contain genetic or physiological characteristics and symptoms, respectively, is one example.

While our algorithm seems to be working fine, it is by no means the final word. Building better algorithms is, as always, an important future direction. For a concrete example, the selection of initial pairs seems to have space for improvements.

Finally, having proofs of the behaviour of the algorithms is important. Is there, for example, an algorithm for real-valued redescription mining for which one can prove that it finds a redescription, provided that the redescription is sufficiently strong?

Acknowledgements

The authors are grateful to Dr. Jussi Eronen for helpful comments and suggestions.

References

[1] R. Bellman. On the approximation of curves by line segments using dynamic programmming. *Comm. ACM*, 4(6):284, 1961.

[2] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. B. Muchnik. An implementation of logical analysis of data. *IEEE Trans. Knowl. Data Eng.*, 12(2):292–306, 2000.

[3] U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. *Mach. Learn.*, pages 1022–1027, 1993.

[4] A. Gallo, P. Miettinen, and H. Mannila. Finding subgroups having several descriptions: Algorithms for redescription mining. In *SDM*, pages 334–345, 2008.

[5] G. C. Garriga, H. Heikinheimo, and J. K. Seppänen. Cross-mining binary and numerical attributes. In *ICDM*, pages 481–486, 2007.

[6] J. Grinnell. The niche-relationships of the California Thrasher. *The Auk*, 34(4):427–433, 1917.

[7] H. Grosskreutz and S. Rüping. On subgroup discovery in numerical domains. *Data Min. Knowl. Disc.*, 19(2):210–226, 2009.

[8] R. J. Hijmans, S. Cameron, L. Parra, P. Jones, and A. Jarvis. Very high resolution interpolated climate surfaces for global land areas. *Int. J. Climatol.*, 25:1965–1978, 2005. `www.worldclim.org`.

[9] D. Kumar. *Redescription mining: Algorithms and applications in bioinformatics.* PhD thesis, Department of Computer Science, Virginia Tech, 2007.

[10] D. Leman, A. Feelders, and A. J. Knobbe. Exceptional model mining. In *ECML/PKDD*, pages 1–16, 2008.

[11] A. J. Mitchell-Jones, G. Amori, W. Bogdanowicz, B. Krystufek, P. Reijnders, F. Spitzenberger, M. Stubbe, J. Thissen, V. Vohralik, and J. Zima. *The atlas of European mammals.* Academic Press, London, 1999. `www.european-mammals.org`.

[12] P. K. Novak, N. Lavrac, and G. I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *J. Mach. Learn. Res.*, 10:377–403, 2009.

[13] M. Ojala, N. Vuokko, A. Kallio, N. Haiminen, and H. Mannila. Randomization methods for assessing data analysis results on real-valued matrices. *Stat. Anal. Data Min.*, 2(4):209–230, 2009.

[14] L. Parida and N. Ramakrishnan. Redescription mining: Structure theory and algorithms. In *AAAI*, pages 837–844, 2005.

[15] R. G. Pearson and T. P. Dawson. Predicting the impacts of climate change on the distribution of species: Are bioclimate envelope models useful? *Global Ecol. Biogeogr.*, 12:361–371, 2003.

[16] V. Pericliev and R. E. Vladès-Pérez. Differentiating 451 languages in terms of their segment inventories. *Studia Linguistica*, 56(1):1–27, 2002.

[17] N. Ramakrishnan, D. Kumar, B. Mishra, M. Potts, and R. F. Helm. Turning CARTwheels: An alternating algorithm for mining redescriptions. In *KDD*, pages 266–275, 2004.

[18] J. Soberón and M. Nakamura. Niches and distributional areas: Concepts, methods, and assumptions. *PNAS*, 106(Supplement 2):19644, 2009.

[19] J. Soberón and A. T. Peterson. Interpretation of models of fundamental ecological niches and species distributional areas. *Biodiv. Inform.*, 2(0), 2005.

[20] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *SIGMOD*, pages 1–12, 1996.

[21] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Springer, 2010.

[22] L. Umek, B. Zupan, M. Toplak, A. Morin, J.-H. Chauchat, G. Makovec, and D. Smrke. Subgroup discovery in data sets with multi-dimensional responses: A method and a case study in traumatology. In *AIME*, pages 265–274, 2009.

[23] M. van Leeuwen. Maximal exceptions with minimal descriptions. *Data Min. Knowl. Disc.*, 21(2):1–18, 2010.

[24] M. J. Zaki. Scalable algorithms for association mining. *IEEE Trans. Knowl. Data Eng.*, 12(3):372–390, 2000.

[25] M. J. Zaki and N. Ramakrishnan. Reasoning about sets using redescription mining. In *KDD*, pages 364–373, 2005.