HAL
open science

From black and white to full colour: extending redescription mining outside the Boolean world

Esther Galbrun, Pauli Miettinen

To cite this version:

Esther Galbrun, Pauli Miettinen. From black and white to full colour: extending redescription mining outside the Boolean world. Proceedings of the 11th SIAM International Conference on Data Mining, SDM'11, Apr 2011, Phoenix, AZ, United States. 10.1137/1.9781611972818.47 . hal-01399265

HAL Id: hal-01399265

https://hal.science/hal-01399265v1

Submitted on 25 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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.

But 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 real-valued data 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 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 May's mean temperature is between -3 and -7 degrees Celsius (indeed, this is one of the results our algorithm produces, see Section 5).

Until now, the redescription mining algorithms (see [4, 14, 16, 24]) 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, 2 and 3, present notation and definitions, and related work, respectively. We explain our algorithm in Section 4 and report about experimental evaluation in Section 5. Section 6 concludes the paper.

Contributions. In this paper we extend redescription mining to real-valued data with an algorithm that efficiently computes the optimal discretization on-the-fly. 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, like 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

*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.

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 two types of variables: Boolean and real-valued. (We omit categorical data for the sake of clarity and brevity, but handling it is straight forward using the approach presented here.) 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 $v \in [a, b]$. We will denote this truth value assignment 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.

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 $v \in V$. To avoid ambiguity, we consider only the *shortest* interval yielding some truth value assignment. An exception to this is the special case where the interval contains the lowest or highest value in the data set for the given variable. We then consider half-lines $(-\infty, b]$ and $[a, +\infty)$, 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 $\mathcal{D}$, $\mathrm{supp}_{\mathcal{D}}(q)$, is a set $\{e \in E : q \text{ is true for } e\}$. The support of a redescription $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$, $\mathrm{supp}_{\mathcal{D}}(q_{\mathbf{L}}, q_{\mathbf{R}})$, is the intersection of supports of $q_{\mathbf{L}}$ and $q_{\mathbf{R}}$, $\mathrm{supp}(q_{\mathbf{L}}, q_{\mathbf{R}}) = \mathrm{supp}(q_{\mathbf{L}}) \cap \mathrm{supp}(q_{\mathbf{R}})$. We will omit the subscripts when they are clear from the context.

A redescription $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$ is *exact* if $\mathrm{supp}(q_{\mathbf{L}}) = \mathrm{supp}(q_{\mathbf{R}})$. 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*

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

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 4.7.

3 Related Work

3.1 Rule discovery. Given a set of observations and a subset thereof, many methods try to find discriminative patterns for the given subset. Depending on the nature of the pattern and the type of variables observed, these techniques are called Logical Analysis of Data (LAD) [2], Emerging Pattern Mining (EPM), Contrast Set Mining (CSM) or Subgroup Discovery (SD), to name a few (see [12] for a unifying survey of the last three).

Attempts have been made to extend Subgroup Discovery from single binary class label to multiple output variables. In [21], this is done by first applying k-medoids clustering to input and output variables separately and then finding relationships between the clusters using χ^2 test of independence. In the Exceptional Model Mining framework [10, 22], the aim is more generally to identify a subset of the entities defined by a single binary feature or a pattern over several features such that a model fitted to that subset significantly differs from the same model fitted to the rest of the entities.

The approach presented in [5] somewhat similarly uses frequent itemsets on the binary attributes to define subsets of the entities and then tries to form a partition of the original data with the subsets that can be best modeled using the numerical attributes.

In Multi-label Classification [20] the goal is to learn classifiers for conjunctions of labels. It shares some similarties with redescription mining, but it can only handle conjunctions and it is a predictive approach while ours is a descriptive one.

Redescription mining differs from these 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. This problem was introduced in [16], and has since attained continuous research interest (e.g. [14, 24, 4, 9]). The algorithms proposed have been based on various ideas, including decision trees [16], co-clusters [14], and frequent itemsets [4].

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 cate-

gorical data using some type of bucketing.

When labels are available on the original data, as it is the case for Subgroup Discovery with a single output feature, a supervised discretisation method can be devised for the problem at hand. In [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 cut 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 [19], who presented a machinery that solves most of the problems automatically. Their method is still based on a priori bucketing. Moreover, it is very specific to association rule mining making it hard (or impossible) to apply to redescription mining.

Using redescription mining algorithms with non-Boolean data is not a new idea. Already in [16], 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. The problem of finding species' bioclimatic envelope is a rather new one (see e.g. [17] 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 [17]. 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 [18]). But to the best of the authors' knowledge, none of these methods allows automatically finding both the set of species and their envelope.

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

4 The Algorithm

In this section, we present our algorithm. We start with some motivation of and ideas behind our design choices. Then, we give a general outline of our algorithm, before moving to some of its building blocks. First we explain how to compute the accuracy of an extended Boolean query efficiently. While important in itself, this technique also forms the basis for bucketing non-Boolean variables on the fly. Next, we give a more detailed explanation of the algorithm, and end this section by discussing the constraints that can be applied in order to retain the most interesting redescriptions.

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 (e.g. [16] uses decision trees of fixed depth, [14] considers monotone CNF and DNF formulae, and [24] only (possibly negated) conjunctions). 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 at least from 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 in [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 expres-

sive 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 $(a \wedge c, \neg b)$, $(a \vee c, \neg b)$, $(a \wedge \neg c, \neg b)$, 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 similaries 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 [24], although without the monotonicity property.

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, $\mathcal{R}$.

$\mathcal{R} \leftarrow \emptyset$
$\mathcal{I} \leftarrow \{k_p$ best initial singleton redescriptions $\}$
for $S \in \mathcal{I}$ **do**
  $\mathcal{C} \leftarrow \{S\}$
  if $F_{\mathbf{L}}(S) \neq \emptyset$ or $F_{\mathbf{R}}(S) \neq \emptyset$ **then**
    $\mathcal{E} \leftarrow \{S\}$
  while $\mathcal{E} \neq \emptyset$ **do**
    for each $R \in \mathcal{E}$ **do**
      for side $s \in \{\mathbf{L}, \mathbf{R}\}$ and operator $\circ \in \{\vee, \wedge\}$ **do**
        if R can be extended on side s with operator $\circ$ and literal $l \in F_s(R)$ **then**
          $\mathcal{C} \leftarrow \mathcal{C} \cup \{$best such extension of R admitting constraints $C\}$
    $\mathcal{C} \leftarrow \{k_i$ best redescriptions from $\mathcal{C}\}$
    $\mathcal{E} \leftarrow \{R \in \mathcal{C} : F_{\mathbf{L}}(R) \neq \emptyset$ or $F_{\mathbf{R}}(R) \neq \emptyset\}$
  $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{C}$
Filter $\mathcal{R}$ with constraints C
return $\mathcal{R}$

Figure 1: ReReMi: Mine data sets for redescriptions

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 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 $\mathrm{J}(q_A \vee \neg v, q_B)$. Should we do this in a straight forward way, we would have to compute three supports over the data for each accuracy. 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 $E_{1,0}([\lambda \leq v \leq \rho])$.

It is well known that the Jaccard coefficient $\mathrm{J}(q_A, q_B)$ can be expressed as $\mathrm{J}(q_A, q_B) = |E_{1,1}| / (|E_{1,0}| + |E_{0,1}| + |E_{1,1}|)$. 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).

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

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

$$\mathrm{J}(q_A \vee v, q_B) = \frac{|E_{1,1}| + |E_{0,1}(v)|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}| + |E_{0,0}(v)|}$$

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

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

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 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 $q_{\mathbf{L}}$ of redescription $(q_{\mathbf{L}}, q_{\mathbf{R}})$ with a real-valued variable v. The algorithm considers the extended query $q_{\mathbf{L}} \wedge [\lambda \leq v \leq \rho]$ 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.

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

Detailed explanation is provided only for non-negated conjunctions; the other cases are analogous. Given a redescription (q_A, q_B) and a variable v, our aim is to find bounds λ and ρ that maximize

$$j(\lambda, \rho) = \frac{|E_{1,1}([\lambda \leq v \leq \rho])|}{|E_{1,0}([\lambda \leq v \leq \rho])| + |E_{0,1}| + |E_{1,1}|}. \tag{4.1}$$

In order to do this efficiently, we will classify the values of λ and ρ that can maximize $j(\lambda, \rho)$. Clearly, only the values of v occuring in entities that belong either to $E_{1,0}$ or to $E_{1,1}$ need to be considered. Furthermore, acceptable bounds for the interval are values that separate entities in $E_{1,0}$ from entities in $E_{1,1}$. Let $(v_1, v_2 \ldots)$ be all the values that appear in $E_{1,0}$ or in $E_{1,1}$, some of them appearing in both, sorted in increasing order (see Figure 3 for an example). A *lower cut point* is a value v_i such that $v_i \in D(E_{1,1}, v)$ and $v_{i-1} \in D(E_{1,0}, v)$. An *upper cut point* is a value v_j such that $v_j \in D(E_{1,1}, v)$ and $v_{j+1} \in D(E_{1,0}, v)$. In the example, v_2, v_3, v_7, v_{10} and v_{11} are lower cut points while v_1, v_2, v_5 and v_7 are upper cut points. We now have

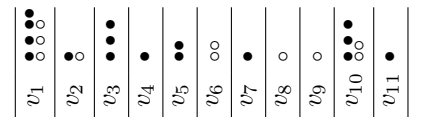

Figure 3: Example of repartition of the entities for one variable. Each bin represents a value taken by the variable. Black circles stand for entities belonging to $E_{1,1}$, white circles for entities from $E_{1,0}$.

Proposition 4.1. *The optimal value for λ is one of the lower cut points or $-\infty$; the optimal value for ρ is one of the upper cut points or $+\infty$.*

Proof. The proof will be for the lower bound λ. The other part is analogous. 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 $\lambda = -\infty$ can be optimal.

Consider the example in Figure 3. 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.

To search for an optimal bucket for variable v, we only need to consider the points defined in Proposition 4.1. 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.

4.5 Interval approximation. In cases where the search space of possible intervals is still too large, we use a faster search to find an interval whose accurracy 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 possible optimal values for λ and for ρ (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 $E_{1,1}$ are not clearly separated, saving heavy computations when encountering variables that are intuitively poor extensions.

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. Still, for sparse data or data with only a limited number of possible values, using the approximate search when necessary, this can be done in reasonable time (cf. Subsection 5.6).

Each of the initial pairs is extended in turn (lines 3–14), selecting at each step the k_i most promising candidates (line 12). A value of 4 for k_i, for example, enables to keep the first step candidates for both operators and both sides. 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.

For this purpose, the algorithm maintains a list of the redescriptions generated so far. The variables leading from R to some already generated one-step extension are not free for R; this includes the variables that appear in R. 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 13). The loop ends when $\mathcal{E}$ is empty, that is, when there is no extendible redescription left.

4.7 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 $|\text{supp}(q_\mathbf{L}, q_\mathbf{R})|$. This probability uses the binomial distribution and is equal to

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

where $p_R = |\text{supp}(q_{\mathbf{L}})|\,|\text{supp}(q_{\mathbf{R}})|\,/\,|E|^2$. The higher the p-value, the more likely it is to observe such a support for independent queries, 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 $\text{pvalE}(q_s, \wedge l) = \text{pvalM}(q_s, l)$ and $\text{pvalE}(q_s, \vee l) = 1 - \text{pvalM}(q_s, l)$.

Also important are the size of the support of the redescription 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 candidates 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 Experimental Evaluation

We now turn to the experimental evaluation of our algorithm. We will start by explaining a method for assessing the significance of the results based on randomizations. Then we will report the actual experiments. The first two sets of experiments (one with synthetic and one with real-world data) are designed to evaluate the capabilities of our algorithm in covering the search space and returning the best possible redescriptions.

Next, we will compare our algorithm against the Greedy algorithm by Gallo et al. [4], to study the effects of our more exhaustive coverage of the search space on the results. Most of these experiments are conducted with fully Boolean data, as performing well with Boolean data is a requisite for performing well with non-Boolean data. We then compare our algorithm against CARTwheels, beginning with a small comparison on fully Boolean data, after which we will compare the bucketing done as pre-processing to our method.

We will conclude the experiments with some real-world examples of niche finding results. The algorithm and the synthetic data generator are available online[3].

5.1 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 use also 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 (w.r.t. the property preserved by the randomization method).

For these experiments, we used two randomization methods. The first method is to permute the matrix – this preserves the values of the matrix. In addition, we used a variation to preserve symmetric matrices: 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, it can preserve symmetrical matrices.

5.2 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

[3] http://www.cs.helsinki.fi/u/galbrun/redescriptors/

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.

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 accuracy.

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 was assumed.

5.3 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 the 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.

The data used for this experiment is called $\texttt{DBLP}_\texttt{B}$. Obtained from the DBLP data base[4], its entities are authors, with one matrix defining the conferences in which each of them has published, and the other defining other authors with whom each of them has published. This version contains 19 hand-picked conferences, 2345 authors, and is Boolean, i.e. it does not contain information on the number of times an author has published in some conference or with some co-author. The same data was used by Gallo et al. [4], where more details are provided about how it was generated.

We used the ECLAT frequent itemset miner [23]. 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 w.r.t. 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.

5.4 Comparison to the work of Gallo et al. As the same data set, $\texttt{DBLP}_\texttt{B}$, is used also by Gallo et al. [4], we compared the results obtained with REREMI to theirs. As Gallo et al. did not allow negation, neither did we. Some results are presented in Table 1. In all tables, J stands for the Jaccard, supp is the support of the redescription, and the p-value is computed as in Section 4.7. The results obtained by REREMI had higher Jaccard similarity than those obtained by Gallo et al.: the highest similarity they report is 0.35, while REREMI returns a redescription with similarity 0.5, and 9 redescriptions have similarity above 0.35.

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 1).

We also tried another version of the DBLP data set, which we denote $\texttt{DBLP}_\texttt{N}$. This is otherwise like $\texttt{DBLP}_\texttt{B}$, but contains the information about the actual number of publications an author has had in a conference and with a co-author. Hence, instead of being Boolean, the matrices contain non-negative integers.

[4] http://www.informatik.uni-trier.de/~ley/db

Table 1: Example results of ReReMi with $\texttt{DBLP}_\texttt{B}$.

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

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

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

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

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

As can be seen from Table 2, this version of the data contains more information, allowing ReReMi to find more accurate redescriptions (the best Jaccard is 0.625). But the rules are also more specific, with small support and often requiring multiple publications in the same conference, making them possibly harder to interpret than those of Table 1. The algorithm works as assumed: it performs better when given more information; rather, the results follow from the data. The (senior) researchers who have published many times in the same conferences are simply the easiest to distinguish. Furthermore, the supports of the redescriptions in Table 1 are mostly small groups of well-known computer scientists. Hence, while the redescriptions are complex, they describe easily understandable groups.

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. On the real-valued data ReReMi's running time was naturally longer but remained very acceptable (less than 8 min).

5.5 Comparison to CARTwheels and pre-bucketing numerical data. We now turn to another algorithm for mining redescriptions, CARTwheels [16]. 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 3.

As can be seen from Table 3, the results have high similarity, 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 similarity (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 vary-

ing 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]).

We used a different data set for these experiments, called `Bio`. In this data, the entities are spatial areas, that is, approximately 50 km squares over Europe[5]. The data itself is composed from two publicly available data sets: 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. This data has 2575 rows (atlas squares), 194 mammals, and 48 climate variables. Notice that the mammals data is Boolean while the climate data is continuous.

Unfortunately, CARTwheels was unable to handle this big data set. To make comparison possible, we concentrated only on Northern Europe (specifically, areas between 50 and 71 degrees North). Also, we removed monthly maximum and minimum temperatures, leaving only monthly average temperature and average precipitation. This left us with 1271 rows, 119 mammals, and 24 climatic variables. We call this data $\texttt{Bio}_{\texttt{small}}$.

The results are reported in Table 4. The aim of this experiment is to study how good accuracies can be obtained with pre-processed buckets compared to ReReMi. Hence, and due to the lack of space, we do not report the actual redescriptions.

The first results are the five best (w.r.t. the accuracy) from CARTwheels using 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 maximum support (above 250) or too high p-value. CARTwheels can still return rather accurate redescriptions, but the quality drops quickly after the first ones.

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

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 (results omitted). 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.

5.6 Real-world application: 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 see 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 w.r.t. these null hypotheses. The algorithm processed the full data in about 13 min.

Some of the results are in Table 5. 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. 4(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. 4(b)), the area occupied by the European Elk. The remaining redescriptions are more complex. The fifth redescription (Fig. 4(c)) covers Northern and Central Europe, while the last covers only Central Europe (Fig. 4(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.

Table 4: Five best accuracies of CARTwheels and ReReMiwith $\texttt{Bio}_{\texttt{small}}$.

CARTwheels			CARTwheels (pruned)			ReReMi		
J	supp	p-value	J	supp	p-value	J	supp	p-value
0.980	1244	0.007	0.896	937	0.000	0.948	931	0.000
0.974	1237	0.158	0.832	944	0.000	0.947	931	0.000
0.974	1237	0.074	0.824	940	0.000	0.934	911	0.000
0.974	1235	0.000	0.674	225	0.000	0.926	888	0.000
0.964	1224	0.337	0.522	175	0.000	0.924	823	0.000

Table 5: 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	$[-7.0727 \leq t_{\text{May}}^{\text{avg}} \leq -3.375]$	0.973	36	0.000
(2)	Polar Bear	$[-16.694 \leq t_{\text{Mar}}^{\text{avg}} \leq -11.462]$	0.973	36	0.000
(3)	Bank Vole $\vee$ Northern Red-backed Vole $\vee$ Steppe Mouse $\vee$ Harbor Seal	$(([11.20 \leq t_{\text{Jul}}^{\max} \leq 15.40] \vee [13.10 \leq t_{\text{Aug}}^{\max} \leq 27.40]) \wedge [42.5 \leq p_{\text{Jul}}^{\text{avg}}]) \vee [17.10 \leq t_{\text{Apr}}^{\max} \leq 17.50]$	0.818	1679	0.000
(4)	European Elk	$([-9.80 \leq t_{\text{Fev}}^{\max} \leq 0.40] \wedge [12.20 \leq t_{\text{Jul}}^{\max} \leq 24.60] \wedge [56.852 \leq p_{\text{Aug}}^{\text{avg}} \leq 136.46]) \vee [183.27 \leq p_{\text{Sep}}^{\text{avg}} \leq 238.78]$	0.814	582	0.000
(5)	Arctic Fox $\vee$ Stoat	$(([2.60 \leq t_{\text{Jun}}^{\max} \leq 8.50] \vee [7.20 \leq t_{\text{Sep}}^{\max} \leq 22.20]) \wedge [36.667 \leq p_{\text{Aug}}^{\text{avg}}]) \vee [21.133 \leq t_{\text{Jul}}^{\text{avg}} \leq 21.20]$	0.813	1477	0.000
(6)	Greater White-toothed Shrew $\wedge$ Egyptian Mongoose	$([15.60 \leq t_{\text{Aug}}^{\min} \leq 19.00] \wedge [1.625 \leq p_{\text{Aug}}^{\text{avg}} \leq 7.4444] \wedge [66.222 \leq p_{\text{Dec}}^{\text{avg}} \leq 137.27]) \vee [19.083 \leq t_{\text{Oct}}^{\text{avg}} \leq 19.10]$	0.790	49	0.000
(7)	House Mouse $\vee$ Caucasian Squirrel $\vee$ Marbled Polecat	$(([3.50 \leq t_{\text{Jan}}^{\max}] \wedge [4.40 \leq t_{\text{Fev}}^{\max}]) \vee [3.5071 \leq t_{\text{Mar}}^{\text{avg}} \leq 4.1727]) \wedge [3.30 \leq t_{\text{Dec}}^{\max}]$	0.765	1034	0.000
(8)	Southwestern Water Vole $\vee$ Azores Noctule $\vee$ Common Noctule $\vee$ Blind Mole	$([17.10 \leq t_{\text{Mar}}^{\max}] \vee [19.30 \leq t_{\text{Aug}}^{\max} \leq 26.90] \vee [12.40 \leq t_{\text{Nov}}^{\max} \leq 14.50]) \wedge [14.60 \leq t_{\text{Sep}}^{\max}]$	0.697	1072	0.000
(9)	Brown Long-eared Bat	$([13.70 \leq t_{\text{Sep}}^{\max} \leq 22.70] \vee [8.4111 \leq t_{\text{Nov}}^{\text{avg}} \leq 8.6444]) \wedge [17.30 \leq t_{\text{Jul}}^{\max} \leq 28.40] \wedge [-8.15 \leq t_{\text{Jan}}^{\text{avg}} \leq 6.0083]$	0.693	963	0.000
(10)	Harvest Mouse $\wedge$ European Mole	$[-0.30 \leq t_{\text{Apr}}^{\min} \leq 8.70] \wedge [19.40 \leq t_{\text{Aug}}^{\max} \leq 27.20] \wedge [45.417 \leq p_{\text{Jun}}^{\text{avg}}] \wedge [48.75 \leq p_{\text{Aug}}^{\text{avg}} \leq 126.33]$	0.677	774	0.000
(11)	(Serotine Bat $\vee$ Lesser Mole Rat) $\wedge$ European Mole	$[19.70 \leq t_{\text{Jul}}^{\max}] \wedge [16.90 \leq t_{\text{Sep}}^{\max} \leq 23.70] \wedge [43.111 \leq p_{\text{Jul}}^{\text{avg}} \leq 149.5] \wedge [31.875 \leq p_{\text{Oct}}^{\text{avg}} \leq 119.5]$	0.634	664	0.000
(12)	Wood Mouse $\wedge$ Natterer's Bat $\wedge$ Eurasian Pygmy Shrew	$([3.20 \leq t_{\text{Mar}}^{\max} \leq 14.50] \wedge [17.30 \leq t_{\text{Aug}}^{\max} \leq 25.20] \wedge [14.90 \leq t_{\text{Sep}}^{\max} \leq 22.80]) \vee [19.60 \leq t_{\text{Jul}}^{\text{avg}} \leq 19.956]$	0.623	681	0.000

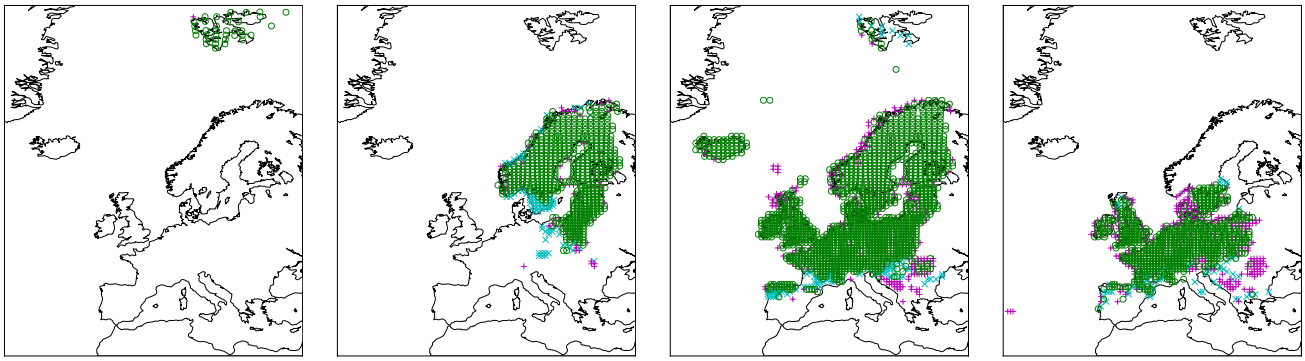

Figure 4: Support of redescriptions when mining Bio data. From left to right, rows 1, 4, 5 and 12 in Table 5. Green circles, cyan plus signs and magenta crosses respectively indicate areas where both queries hold, only the left query holds and only the right query holds.

6 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 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 work together 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 (1961), p. 284.

[2] E. Boros et al., *An implementation of logical analysis of data*, IEEE Trans. Knowl. Data Eng., 12 (2000), pp. 292–306.

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

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

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

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

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

[8] R. J. Hijmans et al., *Very high resolution interpolated climate surfaces for global land areas*, Int. J. Climatol., 25 (2005), pp. 1965–1978. `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, 2008, pp. 1–16.

[11] A. J. Mitchell-Jones et al., *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 (2009), pp. 377–403.

[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 (2009), pp. 209–230.

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

[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 (2003), pp. 361–371.

[16] N. Ramakrishnan et al., *Turning CARTwheels: An alternating algorithm for mining redescriptions*, in KDD, 2004, pp. 266–275.

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

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

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

[20] G. Tsoumakas, I. Katakis, and I. Vlahavas, *Mining multi-label data*, in Data Mining and Knowledge Discovery Handbook, O. Maimon and L. Rokach, eds., Springer, 2010, pp. 667–685.

[21] L. Umek et al., *Subgroup discovery in data sets with multi-dimensional responses: A method and a case study in traumatology*, in AIME, 2009, pp. 265–274.

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

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

[24] M. J. Zaki and N. Ramakrishnan, *Reasoning about sets using redescription mining*, in KDD, 2005, pp. 364–373.