A

Mining Redescriptions with Siren

Esther Galbrun, Inria Nancy – Grand Est
Pauli Miettinen, Max Planck Institute for Informatics

In many areas of science, scientists need to find distinct common characterizations of the same objects and, vice versa, to identify sets of objects that admit multiple shared descriptions. For example, in biology, an important task is to identify the bioclimatic constraints that allow some species to survive, that is, to describe geographical regions both in terms of the fauna that inhabits them and of their bioclimatic conditions. In data analysis, the task of automatically generating such alternative characterizations is called redescription mining.

If a domain expert wants to use redescription mining in his research, merely being able to find redescriptions is not enough. He must also be able to understand the redescriptions found, adjust them to better match his domain knowledge, test alternative hypotheses with them, and guide the mining process towards results he considers interesting. To facilitate these goals, we introduce `Siren`, an interactive tool for mining and visualizing redescriptions.

`Siren` allows to obtain redescriptions in an anytime fashion through efficient, distributed mining, to examine the results in various linked visualizations, to interact with the results either directly or via the visualizations, and to guide the mining algorithm toward specific redescriptions. In this paper, we explain the features of `Siren` and why they are useful for redescription mining. We also propose two novel redescription mining algorithms that improve the generalizability of the results compared to the existing ones.

CCS Concepts: •**Information systems** → **Data mining;** •**Human-centered computing** → **Visualization systems and tools;** Graphical user interfaces;

Additional Key Words and Phrases: redescription mining; interactive data mining; visual data mining

ACM Reference Format:
Esther Galbrun and Pauli Miettinen. 2016. Mining Redescriptions with Siren. *ACM Trans. Knowl. Discov. Data.* V, N, Article A (January YYYY), 30 pages.
DOI: 0000001.0000001

1. INTRODUCTION

This paper presents `Siren`, a tool for exploratory, interactive, and visual redescription mining, as well as two new algorithms for mining the redescriptions.[1]

To support the interactive exploration of data using redescriptions, `Siren` provides multiple state-of-the-art techniques, including, but not limited to, anytime mining algorithms, multiple linked interactive visualizations, and significance testing. Redescriptions have not been visualized earlier, and hence we had to develop new visualizations for them. In particular, we based our visualizations on the parallel coordinates plots and

[1] `Siren` source code and packages for Linux, MS Windows, and Mac OS are available from http://siren.gforge.inria.fr/main/.

Authors' addresses: E. Galbrun, Inria Nancy – Grand Est, 615 Rue du Jardin botanique, FR-54600 Villers-lès-Nancy, France; email: esther.galbrun@inria.fr. P. Miettinen, Max Planck Institute for Informatics, Saarland Informatics Campus Building E1.4, DE-66123 Saarbrücken, Germany; email: pauli.miettinen@mpi-inf.mpg.de.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© YYYY Copyright held by the owner/author(s). Publication rights licensed to ACM. 1556-4681/YYYY/01-ARTA $15.00
DOI: 0000001.0000001

decision tree diagrams. We give a formal definition of redescription mining in Section 3 and describe the features of Siren and the different visualizations in-depth in Section 4.

The two new algorithms we present are both based on decision tree induction, and are presented in Section 5. Section 6 contains an experimental evaluation assessing the quality of the proposed tree-based algorithms, comparing our algorithms' capability of finding redescriptions that generalize to unseen data, and studying the ability of the mining processes to scale and to parallelize. Section 7 provides an outline of work on interactive pattern mining, both retrospective and prospective.

This work is partially based on, and significantly extends, our earlier publications [Galbrun and Miettinen 2012a; 2012c; 2014; Zinchenko et al. 2015]. In short, the main contributions of this paper are as follows: *i)* we present Siren in a consolidated and coherent way, including a thorough discussion about the interactive mining process, *ii)* we present a framework for parallel computation to support this process and study its efficiency, *iii)* we present new tree-based algorithms for redescription mining, extending the presentation of Zinchenko et al. [2015], and *iv)* we evaluate the algorithms implemented in Siren for both their scalability in parallel workloads and for the predictive power of the redescriptions returned by them.

But to begin, we would like to give the reader a first taste of what the matter of the discussion is. To do so, in the following section we introduce the main concepts of the data analysis task and present the workflow for carrying out exploratory redescription mining using Siren informally, by means of an example use case.

2. PROLOGUE: FINDING BIOCLIMATIC NICHES

Let us now walk through a typical use case scenario of Siren. Our example application is *bioclimatic niche-finding*: The bioclimatic niche, or envelope, of a species is defined by a set of constraints on the bioclimatic conditions that need to be satisfied for the species to survive [Grinnell 1917]. Naturally, knowing the species' niches is important, as it can, for example, help to predict the consequences of global warming [Pearson and Dawson 2003].

In this use case scenario, our task is to automatically find the niches of European mammal species (or groups thereof). Our data describes, on one hand, the presence and absence of various mammal species in a (rough) grid of fifty-by-fifty kilometer squares over Europe. On the other hand, we have data on bioclimatic variables (monthly minimum, maximum, and average temperatures and rainfall) over the same grid.

Note that these datasets characterize the same entities over two different sets of variables, that is, the spatial areas of Europe are characterized by Boolean variables representing mammal species on one hand, and by real-valued variables representing monthly climate statistics on the other hand. From now on, we refer to the species data as the *left-hand side* and the bioclimatic data as the *right-hand side* data.

Our aim when analysing this data is to characterize the habitat of certain species with a climatic profile. That is, we want to identify spatial areas that constitute the habitat of a species or group of species while also sharing a specific climatic profile. The species or group of species that inhabits the area of interest constitutes a query over the first set of variables, while the climatic pattern constitutes a query over the second set of variables. Thus, they provide alternative descriptions for the same set of entities. We call such a pair of queries a *redescription*.

In the optimal scenario, we would like to find perfect redescriptions: cases where the habitat of the group of species (the *support* of the first query) matches perfectly with the areas where the considered climate prevails (the support of the second query). However, due to a variety of reasons – such as missing sightings, erroneous climate records, or the impact of external factors on species distribution – mismatches might occur. Hence, rather than look for perfect matches, we would like to find pairs of queries such that

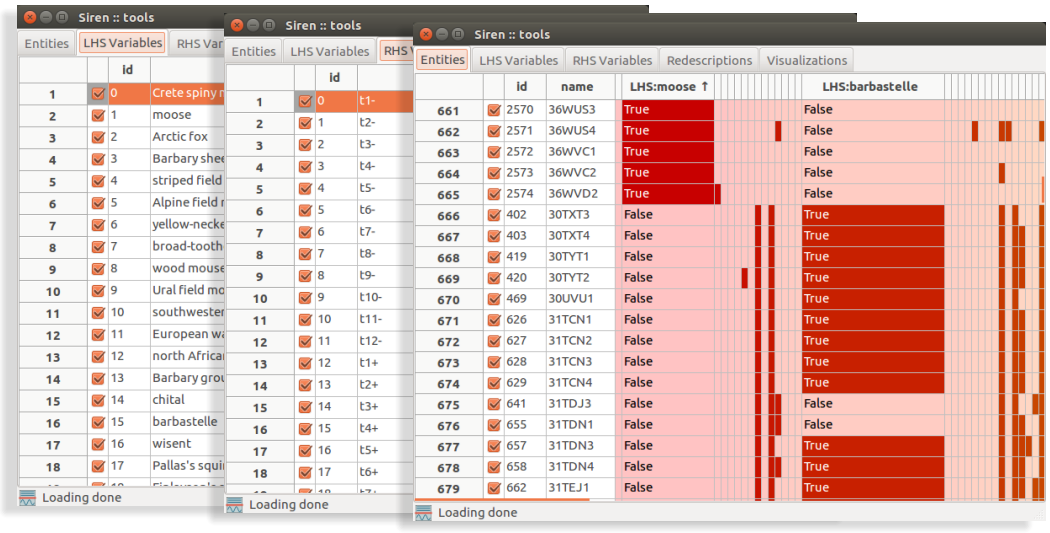


Fig. 1. The list of left-hand side variables, of right-hand side variables, and of entities, from left to right respectively.

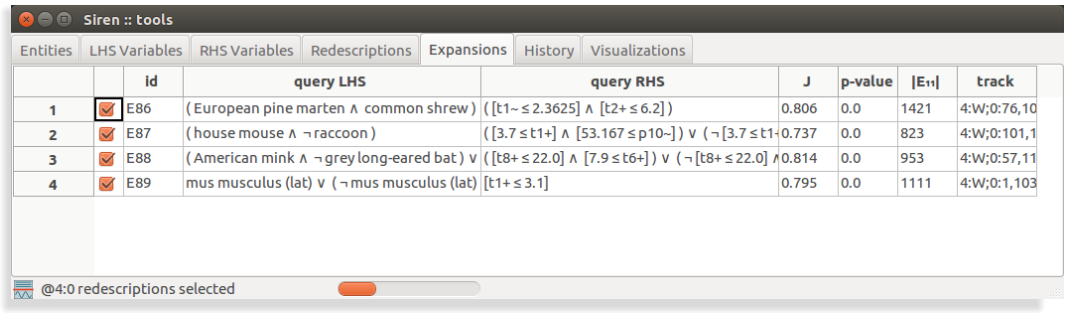


Fig. 2. The list of redescriptions where results appear as they are obtained by the algorithm.

their supports are as similar as possible and measure the similarity of the supports using the *Jaccard coefficient*. Given two sets, this coefficient takes a value between 0 and 1, with higher values meaning that the sets are more similar.

We start by loading the data into `Siren` from a pair of CSV-formatted files (one file for the species dataset and one for the bioclimatic dataset). Once the data is loaded into the interface, we can view the list of entities, as well as the lists of variables for each side respectively (see Figure 1).

The most natural next step is to start mining redescriptions. `Siren` provides a GUI for selecting the algorithm to be used and setting its parameters. The actual mining is conducted in the background, and works in an *anytime* [Dean and Boddy 1988] fashion: As soon as the first redescriptions are obtained, they start appearing in the list. More and more results continue to be added as they are found, until the mining process completes or is stopped.

While the mining process runs in the background, the user can start examining the redescriptions obtained so far (see Figure 2). In addition to the queries constituting the redescription, the listing shows a few measures including the Jaccard coefficient (J),

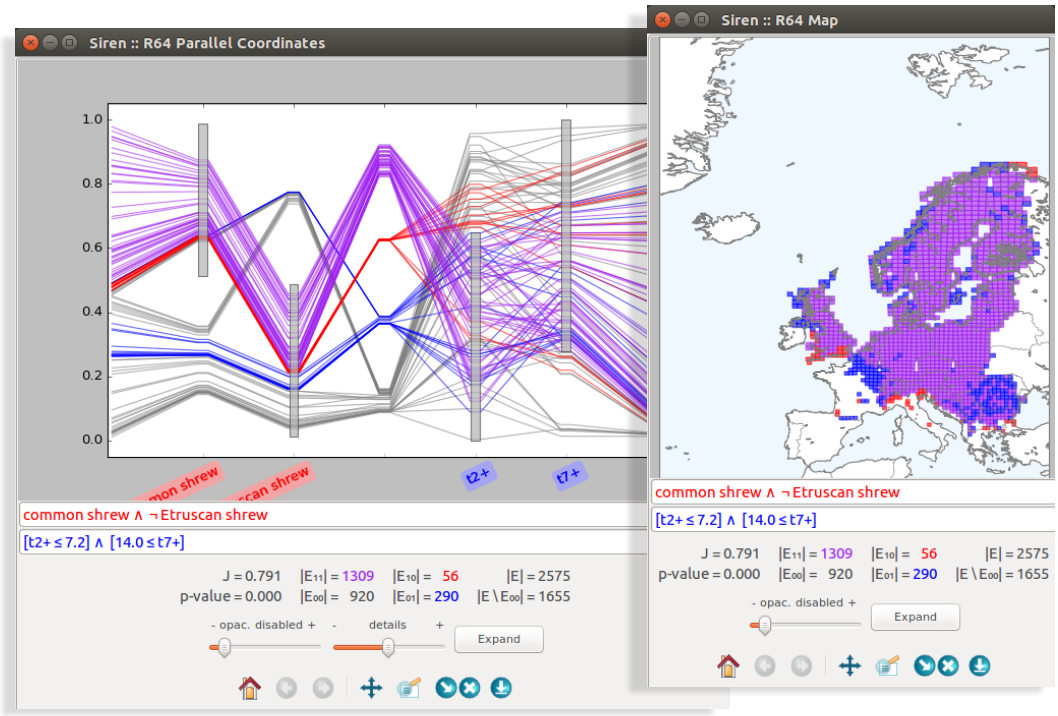


Fig. 3. A redescription visualized as a parallel coordinates plot and projected on a map.

indicating the accuracy of the redescription, and the number of entities for which both queries are true ($|E_{1,1}|$), i.e. the support of the redescription.

For instance, one of the redescriptions listed consists of the following two queries:

$$q_{\mathbf{L}} = \text{common shrew} \wedge \neg\, \text{Etruscan shrew}$$
$$q_{\mathbf{R}} = [t2+ \leq 7.2] \wedge [14.0 \leq t7+]$$

This redescription indicates that the areas inhabited by the common shrew but not by the Etruscan shrew are typically areas where the maximum temperature reaches at most $7.2\,^{\circ}\mathrm{C}$ in February and at least $14.0\,^{\circ}\mathrm{C}$ in July. Having a Jaccard coefficient of $\mathrm{J} = 0.791$, this is a rather accurate redescription.

But is this result intuitive? As the data points are associated with geographical areas, `Siren` can visualize the redescription over a map, as shown in Figure 3 (right). Areas where the left-hand side query $q_{\mathbf{L}}$ holds true but the right-hand side query $q_{\mathbf{R}}$ does not are those areas inhabited by the common shrew but not by the Etruscan shrew and where the maximum temperature in February or July is outside the specified range. They are denoted as $E_{1,0}$ and drawn in red. Similarly, areas where the left-hand side query $q_{\mathbf{L}}$ does not hold true but the right-hand side query $q_{\mathbf{R}}$ does are denoted as $E_{0,1}$ and drawn in blue. Areas where both queries hold, denoted as $E_{1,1}$, are drawn in purple.

This way, we easily notice that most of the mismatches consist of areas where the specified climate prevails but the records of species do not conform (areas belonging to $E_{0,1}$, in blue), located mainly across Romania and the north-east of France.

The map-based visualization is a great way to understand *where* the queries $q_{\mathbf{R}}$ and $q_{\mathbf{L}}$ hold (and do not hold). However, it does not tell us *why* they hold (or do not). To that

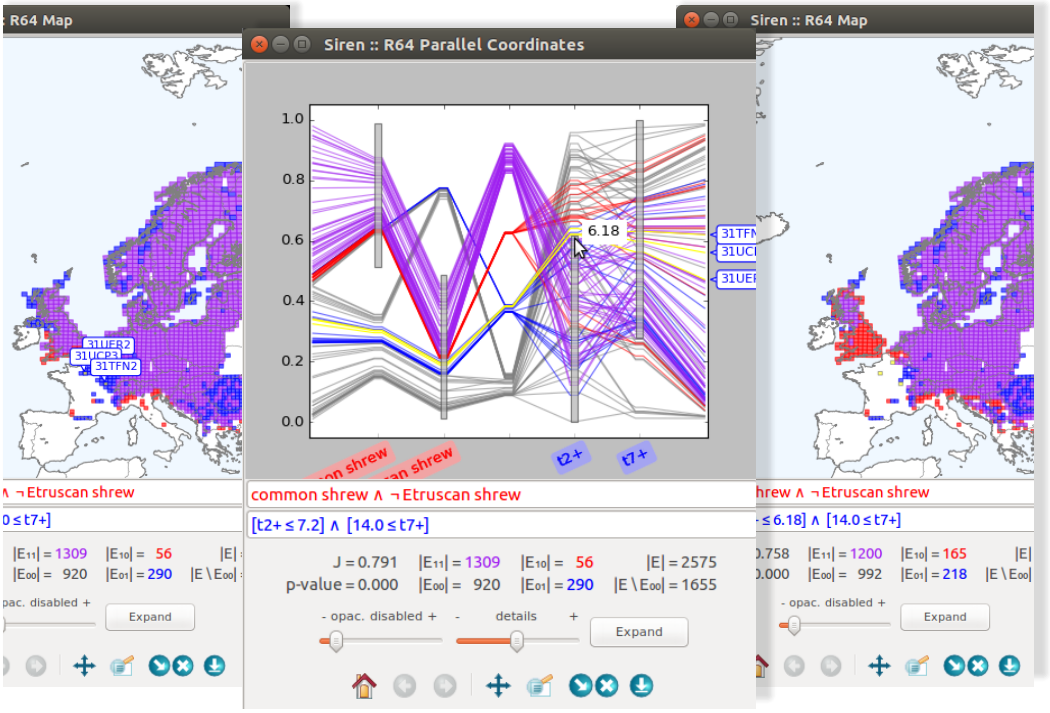


Fig. 4. Interactively editing a redescription through the visualizations. Areas selected on the map are highlighted across visualizations. Using the parallel coordinates plot (center), the bound of a variable is adjusted. The map for the recomputed redescription (right) can be compared to the original one (left).

end, we can use the *parallel coordinates* visualization, as shown in Figure 3 (left). In parallel coordinates, the data points (in this case grid areas) are represented by lines going through a series of parallel vertical axes, one for each condition appearing in the queries. The position where a line crosses an axis indicates the value of the associated variable for the corresponding area. On each axis, a grey box represents the range of values of the variable that satisfy the specified conditions. An extra axis separates the two sides and shows the support of the queries. The same color code is used as with the map. In addition, grey lines represent areas that do not support either query ($E_{0,0}$).

Merely seeing the same redescription visualized in two different ways – over a map and using parallel coordinates – is not all that helpful, however. To benefit from the different visualizations, we need a way to connect the displayed information across visualizations. To do this, `Siren` supports a *brush-and-link* workflow [Heer and Shneiderman 2012]: By selecting an area on the map, we can highlight the corresponding line in the parallel coordinates plot and inspect the values taken by the variables at play in this particular location. For instance, upon selecting a few blue areas in eastern France from the map shown on the left-hand side of Figure 4, the corresponding lines get highlighted in yellow in the parallel coordinates plot shown in the middle of Figure 4. We can then see that the common shrew does not inhabit this area, which is why the left-hand side does not hold. On the other hand, the line crosses the axes associated to the temperatures within their respective specified ranges, so that the right-hand side query does hold.

We can also see from the parallel coordinates that February's maximum temperature in this area is close to the top of the specified range. Hence, we could change the query

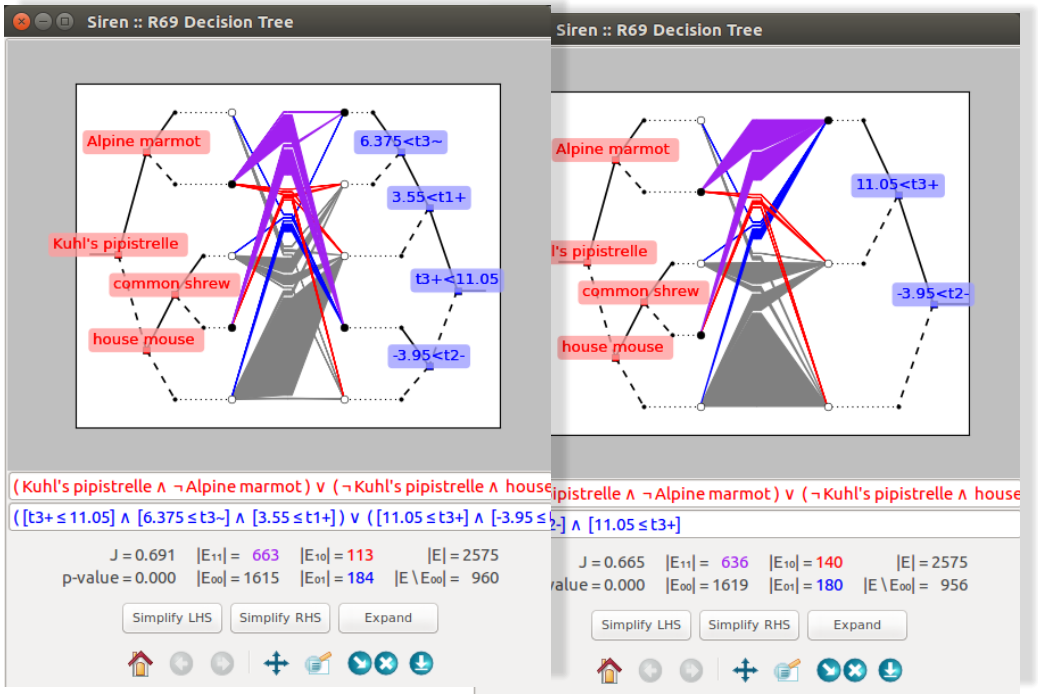


Fig. 5. A redescription visualized as tree diagram, and a simplified variant obtained by removing a branch from the right-hand side query.

by lowering the upper bound for February's maximum temperature, although this would also affect many areas where the query currently holds. Still, we can edit the query simply by dragging the top of the grey box for $t2+$ down to, say, 6.18 degrees Celsius. The queries, plots, and statistics are immediately updated to reflect the change, the resulting map is shown on the right-hand side of Figure 4. We notice that, as a result of the edit, most of southern England no longer match the climate profile, so that the number of areas supporting the queries decreases from 1301 to 1200 and the accuracy drops from 0.791 to 0.758. So, it is best that we revert to the original redescription.

Let us now consider a more complex redescription:

$$\begin{aligned} q_{\mathbf{L}} &= (\text{Kuhl's pipistrelle} \wedge \neg\,\text{Alpine marmot}) \\ &\quad \vee (\neg\,\text{Kuhl's pipistrelle} \wedge \text{house mouse} \wedge \neg\,\text{common shrew}) \\ q_{\mathbf{R}} &= ([t3+ \leq 11.05] \wedge [6.375 \leq t3\sim\,] \wedge [3.55 \leq t1+\,]) \\ &\quad \vee ([11.05 \leq t3+\,] \wedge [-3.95 \leq t2-\,]) \end{aligned}$$

We can use the parallel coordinates to visualize this redescription as well, but the alternating conjunctions and disjunctions make the parallel coordinates harder to interpret. Instead, we can plot it as a pair of decision trees as shown on the left-hand side of Figure 5. Looking at the query $q_{\mathbf{R}}$, the root splits the entities into two sets, depending on whether March's maximum temperature is below or above $11.05\,^{\circ}\mathrm{C}$. But when it is below (top right branch in the figure), only very few areas contribute to $E_{1,1}$. Indeed, there are only few purple lines coming out of that leaf. Consequently, we could consider removing the top branch, leaving only

$$q_{\mathbf{R}} = [11.05 \leq t3+\,] \wedge [-3.95 \leq t2-\,]$$

as the right-hand side query. Indeed, if we do so by clicking on the branch, the accuracy drops from 0.691 to 0.665 but the query is much simpler. The simplified tree diagram is shown on the right-hand side of Figure 5.

Now, we would be interested in finding redescriptions involving a particular species, namely the southwestern water vole. We can do so simply by selecting the chosen variable and letting the algorithm automatically extend it into redescriptions. For instance, the best extension we obtain has an accuracy of $\mathrm{J} = 0.647$:

$$\begin{aligned} q_{\mathbf{L}} &= \text{southwestern water vole} \vee \text{Savi's pine vole} \\ &\quad \vee \text{Mediterranean monk seal} \\ q_{\mathbf{R}} &= [11.2 \leq t3+\,] \wedge [0.51 \leq t1\sim\ \leq 11.333] \\ &\quad \wedge [50.556 \leq p11\sim\ \leq 176.75] \end{aligned}$$

We can also expand the redescription only on one side, say, expanding only the query over the bioclimatic variables, or forbid disjunctions from being used in the query over species, for instance. Further, `Siren` lets us disable some variables, to prevent them from appearing in the queries.

After the mining process has returned a number of redescriptions, and possibly completed, some of these results might cover approximately the same areas, even if they have somewhat different sets of variables. It can be useful to filter redundant redescriptions, to remove redescriptions that do not convey substantially new information. We can either select a redescription and ask `Siren` to filter out all redescriptions that are redundant with respect to it, or we can let the algorithm go through the whole list of redescriptions to filter out all redescriptions that are redundant with respect to some earlier-encountered (i.e. better) redescription.

For instance, the results returned during the extensions mentioned previously may contain many redundant redescriptions found at different steps. We can easily sort them, e.g. by accuracy, select one of interest and filter out all the following results that are redundant with respect to it.

Having filtered, visualized, edited, and refined the redescriptions, we can save those we short-listed, together with the data and parameters into a dedicated file, making it handy to continue the analysis later on. Alternatively, we can also export the list of redescriptions and save our favorite plots in publication-ready formats.

3. REDESCRIPTION MINING

Our aim here is to present the `Siren` interface and discuss its interactive and visual features. However, since `Siren` is a tool dedicated to redescription mining, some background about this data mining task is in order, before delving into a more systematic and detailed discussion of the features of our system.

3.1. Previous Work on Redescription Mining and Related Methods

The problem of mining redescriptions was introduced by Ramakrishnan et al. [2004]. They proposed an algorithm called `CARTwheels` based on decision trees, with an alternating approach to grow decision trees from which redescriptions are then extracted (see also Kumar [2007]).

Apart from decision trees, algorithms have been proposed for Boolean redescription mining, based on approaches including co-clusters [Parida and Ramakrishnan 2005], and frequent itemsets [Gallo et al. 2008].

Algorithms employing heuristics to produce pairs of queries that are almost equivalent on the given dataset were presented by Gallo et al. [2008]. Theses algorithms rely

on different strategies to prune the search space. By means of an efficient on-the-fly discretization, such heuristics were extended to real-valued data, resulting in the `ReReMi` algorithm [Galbrun and Miettinen 2012b].

The main feature of redescriptions is their ability to describe data from different points of view, i.e. their "multi-view" aspect. A similar approach is taken by some other methods such as multi-label classification [Tsoumakas et al. 2010], emerging patterns [Novak et al. 2009], and subgroup discovery [Umek et al. 2009] to name a few (see Galbrun and Miettinen [2012b] for more details). The main differences between redescription mining and these methods are that redescription mining aims to find multiple descriptions simultaneously for a subset of entities which is not specified *a priori*, that it selects only relevant variables from a potentially large number, and that it is symmetric, in the sense that it considers both sides of the data similarly.

3.2. Concepts and Definitions

The input of redescription mining consists of entities with two sets of characterizing variables, thus forming a dataset with two sides.

Throughout this paper, we refer to the two sides as the left- and right-hand sides. We represent the data using two matrices $D_{\mathbf{L}}$ and $D_{\mathbf{R}}$ over two sets of variables, $V_{\mathbf{L}}$ and $V_{\mathbf{R}}$, respectively. The set of entities characterized by the two sides is denoted by E, hence both matrices have $|E|$ rows. The value of $D_{\mathbf{L}}(i,j)$ is the value of variable $v_j \in V_{\mathbf{L}}$ for entity $e_i \in E$.

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. For categorical and real-valued variables, truth value assignments are induced by relations $[v = c]$ and $[a \leq v \leq b]$, respectively, where c is some category and $[a, b]$ is an interval. These truth assignments and their negations constitute *literals* which can be combined using the Boolean operators $\wedge$ (and) and $\vee$ (or) to form *queries*. Then, a redescription is simply a pair of queries over variables from the two sets.

The support of a query q is the subset of entities for which the query holds true, that is, $\mathrm{supp}(q) = \{e \in E : q \text{ is true for } e\}$. We refer to the two sets of variables informally as left- and right-hand side data, and the queries over them as left- and right-hand side queries, denoted as $q_{\mathbf{L}}$ and $q_{\mathbf{R}}$, respectively. Then, a redescription is simply a pair of queries over variables from the two sets, $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$. We denote as $E_{1,1}$ the set of entities for which both queries hold, which we also call the support of the redescription. We further denote as $E_{1,0}$ the set of entities for which only the left-hand side query holds (i.e. $E_{1,0} = \mathrm{supp}(q_{\mathbf{L}}) - \mathrm{supp}(q_{\mathbf{R}})$), $E_{0,1}$ those for which only the right-hand side query holds, and $E_{0,0}$ those for which neither of the queries hold.

The main quality of a redescription is the similarity of the supports of its two queries, also called its *accuracy*. The Jaccard similarity coefficient is the measure of choice to evaluate the accuracy of redescriptions, being at once simple, symmetric, and intuitive. It is defined as follows:

$$\mathrm{J}(R) = \mathrm{J}(q_{\mathbf{L}}, q_{\mathbf{R}}) = \frac{|\mathrm{supp}(q_{\mathbf{L}}) \cap \mathrm{supp}(q_{\mathbf{R}})|}{|\mathrm{supp}(q_{\mathbf{L}}) \cup \mathrm{supp}(q_{\mathbf{R}})|} = \frac{|E_{1,1}|}{|E_{1,1}| + |E_{1,0}| + |E_{0,1}|} \ .$$

Beside accuracy, several other factors impact the quality of a redescription. For instance, we are not interested in redescriptions which are supported by only a handful of entities or conversely by almost all of them, i.e. redescriptions for which $|E_{1,1}|$ is too small or too large.

Furthermore, redescriptions should be statistically significant. To evaluate the significance of results, 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 $|E_{1,1}|$. This probability

uses the binomial distribution and is given by

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

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, and the less significant the query.

In short, given two data matrices, redescription mining is the task of searching for the best matching pairs of queries, with one query over each of the datasets.

The formulation of redescription mining presented here assumes that the describing variables are partitioned into two sets a priori, and looks for a pair of queries over these two sets. This can be naturally adapted to settings with a single set of describing variables. One might then search for pairs of queries, with the constraint that the two subsets of variables appearing in the queries of any redescription be disjoint. Otherwise, the user can be enabled to interactively determine the split between the variables.

The results of redescription mining, the redescriptions, can be approached from two points of view. On one hand, the variables and conditions appearing in the queries provide valuable information in themselves; on the other hand, the support set of the redescriptions, i.e. the subset of entities where both queries of a redescription hold, forms a particularly coherent group.

When the data is geospatial, that is, the entities are connected to geographical locations, the task is called *geospatial redescription mining*. A meaningful geospatial redescription should define coherent areas using expressive queries.

3.3. Application Domains

Finding multiple ways to characterize the same entities, the motivating principle of redescription mining, is a problem that appears in many areas of science.

The task of finding bioclimatic niches, as illustrated in Section 2, is one instance in the field of biology where the goal is to describe geographical regions in terms of both their bioclimatic conditions and the fauna that inhabits them. In bioinformatics, redescription mining has been used to find ways to describe the different isolates of *S. aureus* [Gaidar 2015] as well as to find patterns linking the expression levels of yeast genes during stress tests and the taxonomic categories to which they belong in the Gene Ontology [Ramakrishnan and Zaki 2009].

In political sciences, redescription mining can be used to study the links between people's socio-economical status and their answers to opinion polls [Galbrun and Miettinen 2016]. On the other hand, exact redescriptions (that is, redescriptions with exact match on the left and right support sets) can be used to reduce the search space in sequential equivalence checking of circuits [Goel et al. 2010].

4. THE FEATURES OF SIREN

`Siren` provides a complete environment for redescription mining, from loading the data to finally exporting the results into various formats, through mining, visualizing, and editing the redescriptions. `Siren` allows for a seamless interaction with both mining and visualization, enabling the user to interactively edit the redescriptions and to call the mining algorithm, for instance to extend the current results.

The analysis is — or at least should be — done by a domain expert, as only a domain expert can judge what kind of results are surprising or useful. But to allow the domain expert to perform the data analysis as effectively as possible, the process should be as seamless as possible. Typically, it is unreasonable to assume that the domain experts are proficient on using half a dozen different programs, often with arcane command-line

interfaces, to deploy the full analysis process. There are two common ways to achieve the desired consistency: either via a workflow integrated into a well-known general-purpose analysis framework, such as R or Matlab, or via a special-purpose tool specifically designed for the task at hand. `Siren` takes the latter approach, aiming to answer the data analyst's needs by providing means to interact with the mining algorithm and to visualize and edit the results in an intuitive way within an integrated framework.

To analyse the redescriptions, the ability to visualize their support and the variables involved in the queries is very helpful. That is, visualizing simultaneously the conditions and the support of the queries and how the former affects the latter, as enabled by parallel coordinates plots and tree diagrams, helps understand and interpret the redescriptions. Furthermore, data mining is generally an iterative process, with the results obtained at one step giving rise to hypotheses which will be tested at a further step, and redescription mining is no exception. Providing means to the user to easily interact with the mining process greatly improves the analysis. Therefore, a static display of the results is not enough: the user must also be able to interact with the program. This interaction can be conceptually divided into two sub-phases: interacting with the data mining algorithm and interacting with the resulting visualization. The analysis is an alternation of these two phases, with the user moving back-and-forth between issuing commands to find new results and examining those obtained. We argue that a good interactive data mining tool should support both types of interaction and facilitate the alternation between the different phases.

Below, we discuss the features of the `Siren` interactive and visual redescription mining tool in a systematic manner. We divide the discussion between visualizing, editing, and interacting, though we emphasize that these goals are not independent. Additional screenshots and videos are available on the tool's webpage.[2]

4.1. Visualizing Redescriptions

The most fundamental goal when designing a tool for visual data analysis is, of course, to have good visualizations. Indeed, visualization is the key to understanding the results of the mining process.

In all visualizations in `Siren`, colors encode whether an entity belongs to the support of the left-hand side query ($E_{1,0}$), the right-hand side query ($E_{0,1}$), or both ($E_{1,1}$). The colors can be chosen by the user. By default, we use red, blue, and purple, respectively.

2D projections. The simplest among available visualizations are the various projections of the data into the 2D space. Different types of projections have been studied intensively, and `Siren` provides a number of them, including Karhunen–Loève transform (i.e. PCA), multi-dimensional scaling, and various scatter plots. Figure 6 (left) shows an Isomap embedding [Tenenbaum et al. 2000] of a redescription.

With geospatial redescriptions, a map is the most natural 2D projection, and a very informative one, and `Siren` is capable of plotting redescriptions over entities with geospatial information on a map. The choice of the map projection might alter the visualization significantly. For example, in Figure 6, the same redescription is plotted on a map using both the familiar Mercator projection (middle) and the Lambert azimuthal equal area projection[3] (right). Notice how the Mercator projection makes the blue area in the Svalbard archipelago look huge compared to the Lambert projection. On the other hand, when the data is from a smaller geographical area, the Mercator projection can provide more natural-looking maps. To allow the user to select the best projection for

[2] http://siren.gforge.inria.fr/main/

[3] Lambert azimuthal equal area projection is the recommended map projection for statistical analysis and display in the EU [Annoni et al. 2004].

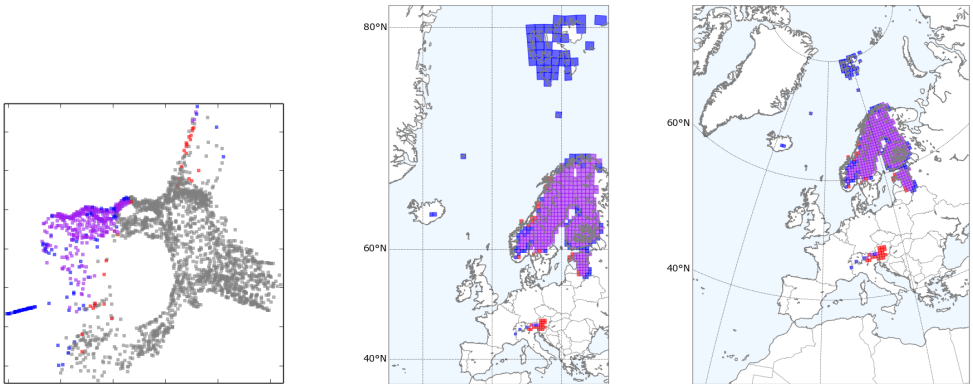


Fig. 6. A redescription visualized using an Isomap embedding (left), a Mercator projection (middle), and a Lambert azimuthal equal area projection (right).

the data, `Siren` supports multiple different projections. Obviously, the map projection is only available for geospatial data. Note that it is the only 2D projection, and more generally the only visualization, with this restriction.

Parallel coordinates plot. With minor enhancements, parallel coordinates plots [Inselberg 2009] are particularly suited for visualizing redescriptions. In such a plot, the entities are represented by lines going through a series of parallel vertical axes, one for each literal appearing in the queries. The position where a line crosses an axis indicates the value of the associated variable for the corresponding entity. For each literal, the range of values that make the truth assignment hold is represented by a grey interval box. An extra axis separates the two sides and registers the support of the queries. Figure 7 shows an example of a parallel coordinates plot. The same color code is used for the entities as with other visualizations. In addition, grey lines represent entities that do not support either query ($E_{0,0}$).

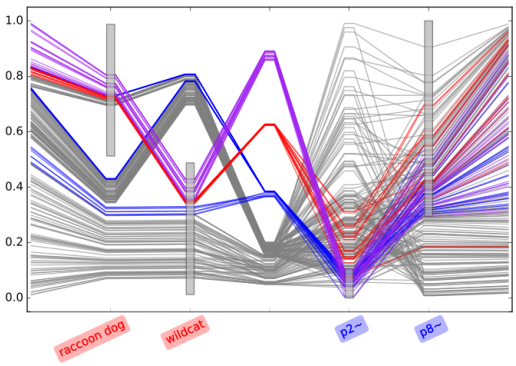


Fig. 7. A parallel coordinates plot.

Parallel coordinates plots are particularly suited to show queries that take the form of conjunctions as they best illustrate how constraints on value ranges can be combined to select a set of entities.

Recently, Palmas et al. [2014] proposed a method of bundling the edges in the parallel coordinates plots to reduce the clutter. Their technique could also be used in `Siren`, although the possibility to see a line corresponding to every entity is still important for interaction, as we explain below.

Trees diagram. When the queries contain nested conjunctions and disjunctions, parallel coordinates can become difficult to interpret. Queries that are (or can be transformed into) disjunctive normal form (DNF) can be expressed as a pair of decision forests. While the transformation to DNF can make the query itself look more complex and difficult to interpret, as is the case for example of the queries shown in Figure 8, the decision forests are easy to understand.

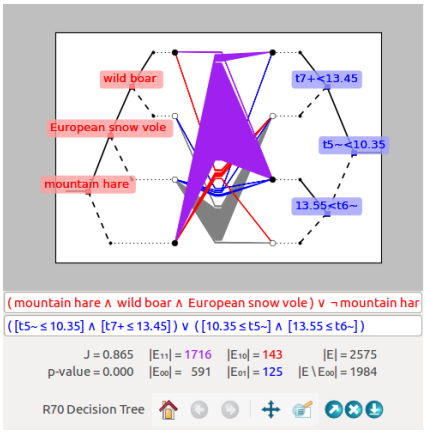


Fig. 8. The visualization of a redescription as a decision tree diagram.

Visualizing a decision tree is, of course, a well-known and not overly complicated problem. What makes our setting special, is that instead of one decision tree, we must visualize two decision trees (or forests) that are linked in their leaves. We visualize this linkage by drawing lines between the leaves: a line connects two leaves if the data has an entity for which the corresponding paths in the trees hold true. Not every path in the tree participates in the query – otherwise we would always cover the whole data – and the color of the line encodes this information, using the same color code as in other visualizations. Furthermore, we do not draw the lines to overlap each other, but next to each other. This allows the user to see the volume of entities connecting different leaves, and hence estimate the importance of the corresponding branch: if only very few entities travel to some leaf, the corresponding branch can often be removed from the query. For instance, the tree diagram displayed in Figure 8 helps interpret the corresponding queries, which appeared somewhat intricate at first glance.

Coordinated views. As, for example, Heer and Shneiderman [2012] argued, having multiple concurrent visualizations of the same data reinforces their explanatory power. `Siren` supports the brush-and-link workflow, where several views and the data are coordinated and modifications made to a redescription are reflected immediately on the different views

For instance, the user can highlight an entity by clicking or hovering over the corresponding line or dot in a view, it will then be highlighted in the other views, allowing to identify it across the different visualizations.

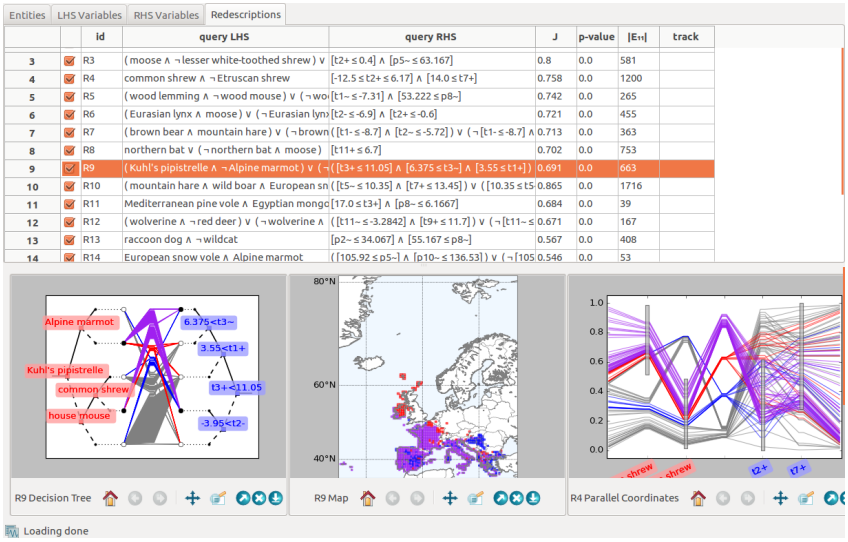


Fig. 9. The main window split into two frames showing the list of redescriptions and a grid of visualizations beneath.

Siren supports showing the visualizations either in their own windows separate from the main application window, or in a dedicated grid of illustrations in the main application window. The latter option facilitates the presentation of multiple visualizations without having to worry about one visualization hiding parts of other visualizations. As shown in Figure 9, the main window can also be split horizontally into two frames in order to examine at the same time a list of redescriptions and several visualizations.

4.2. Editing the Queries

A redescription can be edited from any visualization by using the text fields under the plot. Thus, the user might adjust the conditions in the queries, add or remove variables, as well as build entirely new redescriptions by hand. Upon editing a query, the plot and the statistics of the redescription are recomputed to account for the modifications. Siren allows for simple editing of the redescriptions thanks to flexible parsing of different representations.

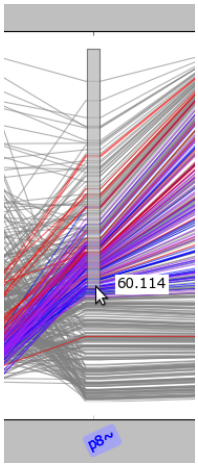


Fig. 10. Editing a redescription in a parallel coordinates plot.

Editing through the visualizations. It is also possible to edit the queries directly from the parallel coordinates plot by dragging the interval boxes to modify the bounds or categories for the variables. For example, in Figure 10, we can simply drag the bottom of the grey box up to exclude some of the blue areas from the support (and, incidentally, some purple areas as well). Of course, this will trigger the re-computation of the statistics of the redescription and the update of all the associated visualizations.

In a tree diagram, the user can modify the query by adding or removing a branch of the decision tree just by clicking on the corresponding leaf. Then, the user may also simplify a query, letting the tool automatically remove variables and branches that no longer affect the query, because they are only involved in tautological constraints, for instance.

4.3. Interacting with the Mining Process

Editing the queries is the most elementary form of redescription mining, as it basically allows to construct patterns manually, without any automation.

Fully automated mining. Obviously, the core of Siren is mining the redescriptions. This can be done fully automatically from the application, which also provides a graphical interface for selecting the mining algorithm and setting its various parameters. However, mining all the redescriptions from a dataset can be a time-consuming task. To avoid extensive waiting times, mining redescriptions in Siren is an asynchronous any-time process, meaning that the user will start seeing results from the mining algorithm as soon as the first (partial) results are ready. The user can start working with and editing these redescriptions while the mining algorithm continues in the background.

Partially automated mining. In between the two extremes of automation lies a mode of partial automation, where the mining algorithm and the analyst collaborate to the construction of results.

For instance, when exploring the data, the user often wants to extend existing redescriptions, either those returned by the algorithm or some queries he has constructed himself, e.g. to test a hypothesis about the data. Siren allows to use an existing redescription as a starting point for the mining process. The user can also decide to let only one side of the redescription be extended. In bioclimatic niche finding, for example, this

can be used to see how good a bioclimatic envelope the algorithm can find for a chosen combination of species.

Like with the fully automated mining, the extensions are computed in an anytime fashion so that the first results are returned almost instantly.

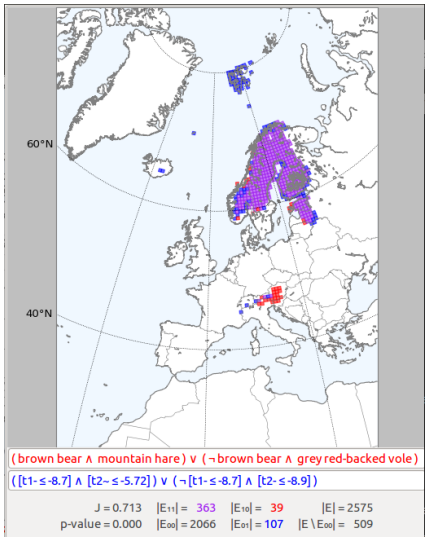


Fig. 11. The polygon selection tool allows to highlight contiguous areas, which can then be used to orient the mining process.

Furthermore, the user may specify a subset of entities that he wants to be emphasized during the mining process. For example, if the user wants to remove some entities (in this case, geographic areas of Svalbard) from the support of the redescription shown in Figure 11, he can highlight them using the polygon selection tool and ask `Siren` to extend the given redescription on the right-hand side with emphasis on excluding them from the support. In other words, the user can manipulate a redescription both through its queries and through its support.

The user can also disable variables and/or entities so that they will not be used in the mining process. For example, there can be known anomalous areas (e.g. coastal regions or valleys in mountain ranges) that the user might want to exclude, to prevent them from affecting the algorithm. The user can also disable the entities after the mining is done, in which case `Siren` will automatically update the support and accuracy of all redescriptions.

Post-processing: Redundancy filtering. `Siren` allows automatic filtering of redundant redescriptions. That is, redescriptions that cover approximately the same entities even if they have (somewhat) different sets of variables. The user can select a redescription and ask `Siren` either to filter out all redescriptions that are redundant with respect to the selected one, or to go through the whole list of redescriptions while filtering out all redescriptions that are redundant with respect to some redescription encountered earlier (and hence consider to be better). Naturally, the decisions made by `Siren` can be reverted whenever the user wishes to.

Significance testing and k-fold mining. `Siren` provides the user with a powerful set of tools to guide the mining process and to edit the resulting redescriptions. Powerful tools increase the risk that the user tailors the results to match his a priori expectations. Miettinen [2014] has argued that interactive data analysis tools should provide methods to warn the user about potential overfitting. To that end, `Siren` employs two standard techniques. The first is to consistently compute the p-value, as explained in Section 3.2.

The second is to study how well the redescriptions mined using a particular set of parameters generalize to unseen entities. In *k-fold mining*, the entities are partitioned into k sets, each of which is left out in turn while the mining algorithm is re-started. The redescriptions found on the training data are re-evaluated over the hold-out data, and their behaviour (e.g. the Jaccard coefficient) is studied. If the Jaccard coefficient is consistently significantly worse in the hold-out data than in the training data, the user can conclude that the parameters used led to overfitting. `Siren` facilitates such k-fold mining experiments by allowing to easily partition the data, run the algorithm on a selected subset of the data and compare accuracy and support of the results across training and hold-out subsets.

4.4. Implementation details

The main design goals of Siren were platform independence, easy extensibility to new methods (both for mining and visualization), and general responsiveness. The platform independence was obtained using Python and the wxPython[4] open source GUI toolkit together with the matplotlib[5] plotting library to plot the various visualizations. Some visualizations, such as the trees diagrams, maps, and parallel coordinates, use a mixture of off-the-shelf and custom-made parts, while the various 2D-projections use the implementations from the scikit-learn[6] package.

Using Python also allowed easy extensibility: The parent visualization, providing the main interactivity and layout primitives, can be inherited and enriched with refined visualization methods. New 2D-projections are even easier, as they only need to accept a numpy[7] matrix and return the corresponding 2D coordinates; Siren will take care of the plotting and linking. New redescription mining algorithms can also be added with relative ease, especially when they are implemented in Python.

Responsiveness is a crucial part of any interactive data mining system. Here, the use of Python has some drawbacks, as the interpreted Python code is not as fast as native code. Furthermore, the mining itself, and computation of some of the visualizations, especially some 2D-projections, is computationally heavy. Naturally, Siren performs these tasks in background threads, thereby exploiting modern multi-core processors to keep the GUI responsive even during heavy computations.

Siren implements a basic framework for parallel redescription mining. The greedy ReReMi algorithm (see Section 5) grows the redescriptions from initial pairs, which can be computed using multiple threads. For all algorithms currently supported, the initial candidates can also be distributed for growing. A master thread creates at most a chosen number of new threads, each one growing a candidate separately. These threads report back results to the master thread, that appends them to the collected results and filters away redundant ones. A new thread is then launched to grow the next candidate, and so on, until all initial candidates have been processed.

To improve the responsiveness even further, and to allow the mining and exploration of datasets too large to be handled on conventional desktop or laptop computers, Siren can also offload the computations onto external servers. The anytime behaviour of the algorithms in Siren allows the user to launch time-consuming mining operations in a computing server, and start analysing the results soon after, when the first redescriptions return. The server module in Siren is naturally multi-threaded, and utilizes the client–server architecture, enabling it to serve multiple clients simultaneously. The Siren server module can run both from a local computing server (e.g. university) or from a cloud server (e.g. Amazon's EC2).

Finally, Siren facilitates distributing the results and sharing information. It can handle any data provided in a compatible format. Redescriptions can be exported in easy-to-read format and their visualizations can be readily converted to publication-ready graphics. Data, settings and pre-mined redescriptions can be saved in dedicated archives that can be easily reloaded later on.

5. MINING ALGORITHMS

At the core of Siren, feeding the results to be visualized and interacted with, are the mining algorithms. Currently, Siren supports the greedy ReReMi algorithm and a number of algorithms based on classification and regression trees (CART). In what

[4]https://wxpython.org, accessed 18 October 2016.
[5]http://matplotlib.org, accessed 18 October 2016.
[6]http://scikit-learn.org, accessed 18 October 2016.
[7]http://www.numpy.org, accessed 18 October 2016.

follows, we will give a quick recap of the ReReMi algorithm (Galbrun and Miettinen [2012b] provide a full explanation), and proceed to explain our two novel CART-based algorithms in more detail in Section 5.2.

5.1. The Greedy Approach

Building on the work of Gallo et al. [2008], Galbrun and Miettinen [2012b] designed a greedy algorithm called ReReMi using efficient on-the-fly discretization to extend redescription mining to categorical and numerical variables. The queries ReReMi builds can be parsed in linear order, without trees, with every variable allowed to appear only once. They constitute a subset of Boolean formulae that provides a good compromise between expressive power, difficulty of search, and interpretability.

Yet, the search space remains exponential and we resort to heuristic pruning. 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. After evaluating all possible one-step extensions, we select the best candidates and extend them in turn. This process stops when no new redescription can be generated. Redescriptions with too high p-value can be filtered out during the search. We exploit some simple observations to make the computation of accuracy more efficient, allowing faster evaluation of the candidates, which is particularly important for an interactive tool.

Owing to this beam-search-like behavior, ReReMi is an any-time algorithm. The intermediate redescriptions explored during the search are returned at each step. This way, the user is able to see the candidates present in the beam and might stop the extension process if he wishes. The possibility to remove a candidate from the beam, cutting off a less promising branch from the search, remains to be implemented.

5.2. The Tree-based Approach

Decision tree induction [Quinlan 1986] presents an alternative approach for mining redescriptions. This idea was originally introduced by Ramakrishnan et al. [2004]. Our algorithms, however, differ from the existing ones both in their inner working and in their capabilities. Indeed, unlike existing tree-based methods, the family of algorithms based on decision tree induction that we propose can handle non-binary data and scale well. As evidenced by our experiments, they allow to find intuitive and accurate redescriptions that, importantly, generalize well to unseen data.

More specifically, our approach consists in growing two trees in opposite directions, gradually increasing their depth and matching their leaves. We use *classification and regression trees (CART)* [Breiman et al. 1984] for this purpose, but any other approach to induce decision trees can be exploited as well.

Growing Trees. Our input is a pair of data matrices and we use Boolean, categorical or sparse numerical variables to initialize our procedure. Specifically, the initialization of the algorithm requires a binary target vector for building the first tree. Boolean variable columns directly provide such binary vectors. Each categorical variable can be turned into as many binary target vectors as it has categories, by considering membership in each category separately. Finally, sparse numerical variables also provide binary target vectors by considering zero vs. non-zero entries. The vector of predictions output at one step is then used as the target vector when growing a tree over attributes from the other side during the next step, in alternating iterations. Without loss of generality and for simplicity of exposition, we assume that the initialization variable is from the left-hand side, so that the first tree will be induced over the right-hand side variables.

Input: A pair of data matrices $(D_{\mathbf{L}}, D_{\mathbf{R}})$,
an initialization variable $v_i \in V_{\mathbf{L}}$, and max depth κ.
Output: A redescription $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$.
1: $\tau_{\mathbf{L}} \leftarrow$ initialize the target with v_i
2: **for** $k \leftarrow \{1..\kappa\}$ **do**
3: $T_{\mathbf{R}} \leftarrow$ induce tree over $D_{\mathbf{R}}$ with target $\tau_{\mathbf{L}}$ and depth k
4: $\tau_{\mathbf{L}} \leftarrow$ extract classification vector from $T_{\mathbf{R}}$
5: $T_{\mathbf{L}} \leftarrow$ induce tree over $D_{\mathbf{L}}$ with target $\tau_{\mathbf{R}}$ and depth k
6: $\tau_{\mathbf{R}} \leftarrow$ extract classification vector from $T_{\mathbf{L}}$
7: **end for**
8: $q_{\mathbf{L}} \leftarrow$ extract query from positive branches of $T_{\mathbf{L}}$
9: $q_{\mathbf{R}} \leftarrow$ extract query from positive branches of $T_{\mathbf{R}}$
10: **return** $(q_{\mathbf{L}}, q_{\mathbf{R}})$

Fig. 12. The `SplitT` algorithm.

This procedure continues to alternate between growing trees over either side of the data, until one of the following three stopping conditions is met: *i)* no tree can be induced with the given parameters, *ii)* the maximal tree depth chosen by the user is reached, or *iii)* the prediction vector obtained from the new tree is identical to the one previously obtained on that side, in other words, the tree growing procedure has reached a fixed point.

A tunable parameter, $l_{\min}$, controls the minimum number of entities allowed in any leaf of the tree. It allows us to combat overfitting and terminate the tree induction earlier.

We present two algorithms which follow the process described above but differ in their strategy for growing trees.

The `SplitT` Algorithm. Our first algorithm grows a new classification tree at each iteration while progressively increasing their depth. An outline of the `SplitT` algorithm is shown in Figure 12. As explained above, a variable $v_i \in V_{\mathbf{L}}$ is initially used to provide a target vector to induce a tree over the variables on the other side, i.e. over the matrix $D_{\mathbf{R}}$. This first iteration produces a tree of depth one with two leaves, which are labelled according to the majority class of the entities they contain, one positive and one negative. The corresponding binary prediction vector for the entities is then used as a target vector to grow a new tree over $D_{\mathbf{R}}$, this time of depth two. In turn, the prediction vector is used as a target to learn a tree of depth two over $D_{\mathbf{R}}$ from scratch, and so on.

The `LayeredT` Algorithm. Our second algorithm grows trees layer by layer instead of building a new tree from scratch in each iteration. One layer is added to the current candidate tree by appending a new decision tree of depth one to each of its branches, each of which is learnt independently from the others. An outline of the `LayeredT` algorithm is shown in Figure 13.

Extracting Redescriptions. At the end of the alternating process, we obtain a pair of decision trees, over either sets of variables. The extraction of a redescription from such a pair of decision trees is illustrated in Figure 14. It works as follows. Each leaf corresponds to the set of entities that satisfy the conditions specified along the branch leading up to it (this relation is represented by grey arrows leading from the leaves, on both sides, to the entities, in the middle). Each leaf is labelled as belonging either to the positive or the negative class according to the majority label (from the last round of tree induction) among the entities associated to the leaf. This way, the trees are matched by their leaves through the common entities.

From either tree we obtain a query over the variables from that side charaterizing the positive class. One literal is constructed for each node of the decision tree using the associated splitting variable and threshold. Then, literals leading to the positive leaves

Input: A pair of data matrices $(D_{\mathbf{L}}, D_{\mathbf{R}})$,
an initialization variable $v_i \in V_{\mathbf{L}}$, and max depth κ.
Output: A redescription $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$.

$\tau_{\mathbf{L}} \leftarrow$ initialize the target with v_i
$T_{\mathbf{R}}^{(1,\emptyset)} \leftarrow$ induce tree over $D_{\mathbf{R}}$ with target $\tau_{\mathbf{L}}$ and depth 1
$\tau_{\mathbf{L}} \leftarrow$ extract classification vector from $T_{\mathbf{R}}$
$T_{\mathbf{L}}^{(1,\emptyset)} \leftarrow$ induce tree over $D_{\mathbf{L}}$ with target $\tau_{\mathbf{R}}$ and depth 1
$\tau_{\mathbf{R}} \leftarrow$ extract classification vector from $T_{\mathbf{L}}$
for $k \leftarrow \{2..\kappa\}$ **do**
  for each leaf ℓ of $T_{\mathbf{L}}$ **do**
    $T_{\mathbf{R}}^{(k,\ell)} \leftarrow$ induce tree over the subset of $D_{\mathbf{R}}$ contained in ℓ with target $\tau_{\mathbf{L}}$ and depth 1
  end for
  $\tau_{\mathbf{R}} \leftarrow$ extract classification vector from the right-hand side trees at level k, $T_{\mathbf{R}}^{(k,*)}$
  for each leaf ℓ of $T_{\mathbf{R}}$ **do**
    $T_{\mathbf{L}}^{(k,\ell)} \leftarrow$ induce tree over the subset of $D_{\mathbf{L}}$ contained in ℓ with target $\tau_{\mathbf{R}}$ and depth 1
  end for
  $\tau_{\mathbf{L}} \leftarrow$ extract classification vector from the right-hand side trees at level k, $T_{\mathbf{L}}^{(k,*)}$
end for
$q_{\mathbf{L}} \leftarrow$ extract query from positive branches of stacked trees $T_{\mathbf{L}}^{(*,*)}$
$q_{\mathbf{R}} \leftarrow$ extract query from positive branches of stacked trees $T_{\mathbf{R}}^{(*,*)}$
return $(q_{\mathbf{L}}, q_{\mathbf{R}})$

Fig. 13. The `LayeredT` algorithm.

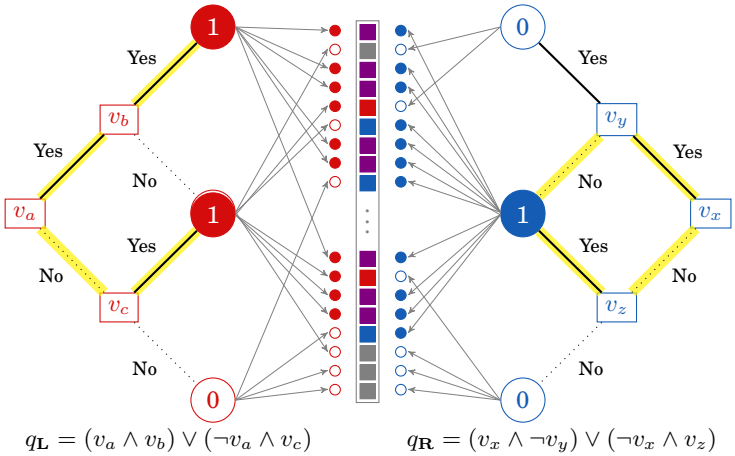


Fig. 14. Extraction of a redescription from a pair of matched trees. On either side, the tree represents a succession of tests on some variables' values (represented as square nodes) and leading to a classification decision (represented as circular leaf nodes). The label (positive or negative) assigned by the tree to each entity is represented as a circle (filled or empty, respectively). The support obtained when matching the trees is shown at the center using the color code as in other visualizations. Finally, the branches leading to positive leaf nodes, which participate in the queries, are highlighted in yellow.

(highlighted in yellow in the schema) are combined into a query, using conjunctions ($\wedge$) to combine literals within one branch and disjunctions ($\vee$) to combine different branches.

The output of our algorithms consists of the collected redescriptions over all induced tree pairs.

6. EXPERIMENTAL EVALUATION

We now turn to the empirical evaluation of the algorithms. We start by studying how well the proposed tree-based algorithms work on finding planted redescriptions and evaluating the interpretability of the results (Section 6.2). We then investigate how well the results from different redescription mining algorithms (the tree-based ones, `ReReMi`, and `CARTwheels`) generalize to unseen data (Section 6.3). Finally, we evaluate the scalability of `Siren`'s multi-threaded redescription mining framework (Section 6.4). But first, let us introduce the real-world datasets used in this section.

6.1. Datasets

Our first dataset, Bio, is the dataset used in the bioclimatic niche-finding task (see Section 2). The entities represent geographic areas of Europe, the left-hand side records the presence of various mammal species [Mitchell-Jones et al. 1999] while the right-hand side consists of bioclimatic variables, that is, monthly average rainfall and monthly average, minimum, and maximum temperatures [Hijmans et al. 2005] (number of entities $|E| = 2575$, number of left-hand side variables $|V_{\mathbf{L}}| = 194$, and number of right-hand side variables $|V_{\mathbf{R}}| = 48$).

To allow experiments with the `CARTwheels` algorithm, we extract a subset of this data, which we denote as Bio_S, by selecting only areas from Northern Europe, namely areas located at latitudes between 50 and 71° North, and keeping only the average monthly temperatures and rainfall ($|E| = 1271$, $|V_{\mathbf{L}}| = 194$, and $|V_{\mathbf{R}}| = 24$).

To demonstrate the scalability of our algorithms, we consider a dataset similar to Bio but spanning the entire globe rather than only Europe. This Globe dataset contains the same set of bioclimatic variables as Bio and all terrestrial mammals from the IUCN Red List spatial data [IUCN 2014]. The alignment of the two data sets was done by Lawing et al. [2016]. The resulting dataset is significantly larger than Bio, having $|E| = 54013$, $|V_{\mathbf{L}}| = 4754$, and $|V_{\mathbf{R}}| = 48$.

The Cover dataset is another large dataset, with more than half a million entities, but with only a few dozen variables. It comes from the UCI Machine Learning repository.[8] Entities once again represent geographic areas. As the right-hand side variables, we consider the wilderness area, soil type, and cover type variables and keep the other variables such as elevation and slope on the left-hand side ($|E| = 581012$, $|V_{\mathbf{L}}| = 10$, and $|V_{\mathbf{R}}| = 45$).

Finally, the DBLP dataset is extracted from the popular computer science bibliography database.[9] The entities are researchers and one side records the co-authorship graph, while the other side records the number of their publications in each of the major conferences considered ($|E| = 2345$, $|V_{\mathbf{L}}| = 2345$, and $|V_{\mathbf{R}}| = 19$).

6.2. Behavior of the tree-based algorithms

Finding Planted Redescriptions. Before tackling these real-world datasets, we tested our algorithms on synthetic data where we planted redescriptions corrupted by noise.

We generated synthetic datasets in which we hid redescriptions to assess whether our algorithms are able to recover them, following a procedure similar to that of Galbrun and Miettinen [2012b]. Specifically, we generated binary matrices with 300 entities (rows) and 10 variables (columns) and planted a query involving three variables in each of them so as to obtain pairs of datasets containing a perfectly accurate redescription supported by 30 to 50 entities. We then added random noise with densities ranging between 0.01 and 0.1 and which could either be destructive (i.e. potentially reducing the accuracy of the planted redescription) or not. The data on one side of these synthetic

[8] https://archive.ics.uci.edu/ml/datasets/Covertype
[9] http://www.informatik.uni-trier.de/~ley/db/

pairs of matrices was turned into real values by substituting zeros and ones by values sampled uniformly at random from the intervals $[0, 0.25]$ and $[0.75, 1]$ respectively.

Both algorithms behaved as expected, recovering the planted redescriptions with the lowest noise densities and returning redescriptions with better accuracy than the planted ones under higher levels of noise, because more accurate redescriptions can be incidentally created as a result of inserting noise.

Quality of the redescriptions. To further explore the behavior of our algorithms, this time on real-world data, we conducted experimental runs with both our algorithms while varying their parameters. In particular, we used either the *Gini coefficient* or the *information gain* as the impurity measure for learning the decision trees and set the minimum number of entities per leaf, $l_{\min}$, to 1, 5, 10 or 50. In all these experiments, we considered versions of the datasets which consist of a Boolean side and a numerical side and used the variables from the Boolean side of the data ($V_{\mathbf{L}}$) to initialize the tree algorithms.

On the Bio dataset, both algorithms always returned statistically significant redescriptions (p-value < 0.01). All other conditions being equal, using the Gini coefficient as impurity measure with `SplitT` resulted in slightly deeper trees and, consequently, in longer redescriptions. Using the information gain produced more duplicate redescriptions for both algorithms.

We present examples of redescriptions mined by our algorithms from the Bio dataset. The `LayeredT` algorithm with information gain and $l_{\min} = 50$ found the following redescription of accuracy $\mathrm{J} = 0.691$ and support $|E_{1,1}| = 663$:

$$q_{\mathbf{L}} = (\text{Kuhl's pipistrelle} \wedge \neg\,\text{Alpine marmot}) \\ \vee\ (\neg\,\text{Kuhl's pipistrelle} \wedge \text{house mouse} \wedge \neg\,\text{common shrew})$$
$$q_{\mathbf{R}} = ([t3+\ < 11.05] \wedge [6.375 \leq t3\sim\ ] \wedge [3.55 \leq t1+\ ]) \vee ([11.05 \leq t3+\ ] \wedge [-3.95 \leq t2-\ ])$$

The `SplitT` algorithm with Gini coefficient and $l_{\min} = 20$ found the following redescription of accuracy $\mathrm{J} = 0.865$ and support $|E_{1,1}| = 1716$:

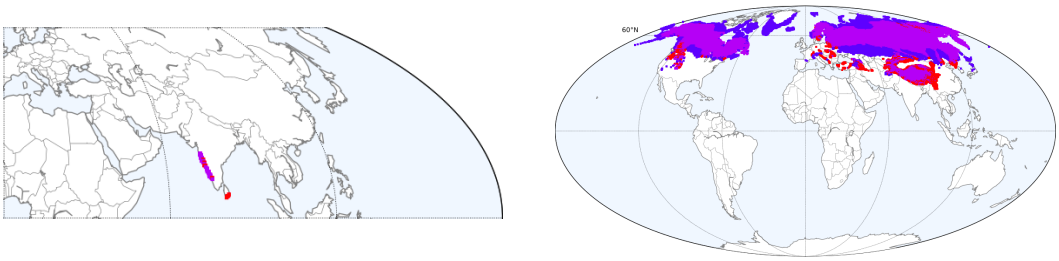
$$q_{\mathbf{L}} = (\text{mountain hare} \wedge \text{wild boar} \wedge \neg\,\text{European snow vole})\ \vee\ (\neg\,\text{mountain hare})$$
$$q_{\mathbf{R}} = ([t5\sim\ < 10.35] \wedge [t7+\ < 13.45]) \vee ([10.35 \leq t5\sim\ ] \wedge [13.55 \leq t6\sim\ ])$$

The corresponding tree diagrams are shown respectively in Figure 5 (left) and Figure 8.

Comparison of `SplitT` and `ReReMi` on the Globe data. We use the Globe dataset to highlight some differences between the typical results obtained using the `SplitT` and `ReReMi` algorithms. Particularly, in this data, the results given by `SplitT` explained small numbers of entities, while `ReReMi` returned redescriptions with very high support. One type of result is not, per se, better than the other, but they do highlight the differences in the way the algorithms work. We would also like to emphasize that restricting the minimum or maximum support would allow the user to steer the algorithms towards redescriptions with more (or fewer) entities in the support sets. Here, we used 15 and 500 as the minimum sizes of $E_{1,1}$ and $E_{0,0}$ respectively.

We present two example redescriptions in Figure 15. The redescription returned by `SplitT` has $|E_{1,1}| = 23$, concentrating on the west coast of India. It describes the area where the lesser woolly horseshoe bat lives but neither the dusky leaf-nosed bat nor the Bengal fox are observed. Notably the right-hand side query $q_{\mathbf{R}}$ uses only precipitation variables.

The right-hand side example of Figure 15 is obtained with `ReReMi` and has a significantly larger support of $|E_{1,1}| = 10\,875$. It describes the areas inhabited either by the Eurasian or Canada lynx as the areas where March's maximum temperature is between $-24.4\,^{\circ}\mathrm{C}$ and $3.4\,^{\circ}\mathrm{C}$. These are but two examples of redescriptions found from



$q_{\mathbf{L}} =$ lesser woolly horseshoe bat $\wedge\neg$ dusky leaf-nosed bat $\wedge\neg$ Bengal fox	$q_{\mathbf{L}} =$ Eurasian lynx $\vee$ Canada lynx
$q_{\mathbf{R}} = [1109.5 \leq p7 \sim\ ] \wedge [p5 \sim\ \leq 292.0] \wedge [96.5 \leq p10 \sim\ ]$	$q_{\mathbf{R}} = [-24.40 \leq t3 +\ \leq 3.40]$
$J = 0.719, \quad \lvert E_{1,1} \rvert = 23$	$J = 0.675, \quad \lvert E_{1,1} \rvert = 10875$

Fig. 15. Example redescriptions obtained from the Globe data with `SplitT` (left) and `ReReMi` (right). The maps use the Mollweide projection and the left one is restricted between the equator and the latitude $60°$ north and east from the prime meridian.

Table I. Accuracy of redescriptions. We report the number of redescriptions mined (#), the average accuracy (Jaccard coefficient, J) in the training data and overall as well as the average of the ratio between the accuracy in the training set and overall ($\pm$ standard deviation).

Data	Algorithm	#	J training	J overall	J ratio
Bio_S	`SplitT`	46	0.87 ($\pm$0.09)	0.86 ($\pm$0.09)	0.98 ($\pm$0.01)
	`LayeredT`	77	0.61 ($\pm$0.17)	0.61 ($\pm$0.17)	0.99 ($\pm$0.02)
	`ReReMi`	9	0.88 ($\pm$0.04)	0.58 ($\pm$0.22)	0.66 ($\pm$0.24)
	`CARTwheels`	88	0.87 ($\pm$0.07)	0.87 ($\pm$0.07)	0.99 ($\pm$0.01)
Bio	`SplitT`	137	0.82 ($\pm$0.12)	0.81 ($\pm$0.12)	0.99 ($\pm$0.01)
	`LayeredT`	156	0.49 ($\pm$0.20)	0.49 ($\pm$0.20)	1.01 ($\pm$0.01)
	`ReReMi`	56	0.92 ($\pm$0.04)	0.55 ($\pm$0.27)	0.59 ($\pm$0.30)
DBLP	`SplitT`	37	0.75 ($\pm$0.19)	0.70 ($\pm$0.17)	0.94 ($\pm$0.07)
	`LayeredT`	577	0.13 ($\pm$0.07)	0.12 ($\pm$0.07)	0.96 ($\pm$0.11)
	`ReReMi`	23	0.36 ($\pm$0.05)	0.06 ($\pm$0.04)	0.18 ($\pm$0.15)

this data, but they illustrate the different types of redescriptions `SplitT` and `ReReMi` typically find.

6.3. Generalization to unknown

As we argued in Section 4.3, testing how well the results generalize to unseen data is an important step in preventing the user from being carried away by the power of the interactive mining tool. Besides, generalizability is of course a very desirable feature for any data analysis result, irrespective of how it was obtained. Given that the different redescription mining algorithms find very different types of queries (trees versus the linearly parseable queries of `ReReMi`) and find them in different ways, it is not clear how well the results of the different methods generalize.

We compared the redescriptions obtained with the `SplitT` and `LayeredT` algorithms proposed in this article, to those returned by the `ReReMi` algorithm (see Section 5.1) and the `CARTwheels` algorithm [Ramakrishnan et al. 2004] (implementations provided by the authors).

To study the ability of the redescriptions to generalize to unseen data, we selected 80% of the entities, mined redescriptions from this training set, then assessed their accuracy on the full original datasets.

For the DBLP dataset, entities were split at random between training set and hold-out set. For Bio and Bio$_S$, we had to take into account the north–south trends in climate data. Predicting conditions in northern areas with data from southern ones (or vice versa) is unlikely to succeed. Hence we sampled longitudinal stripes, and the hold-out set consisted of all points lying (roughly) on the sampled longitudes (in total 20% of observations were held out).

With Bio and Bio$_S$, we set the minimum support to 200 and the maximum support to 1800 (with DBLP, to 5 and 1300 respectively) and the maximum p-value to 0.05 for all algorithms and all datasets.

The `CARTwheels` algorithm was able to handle only the Bio$_S$ data, running out of memory with the other datasets. As `CARTwheels` also requires fully binary data, we discretized the climate data on Bio$_S$. We tested various techniques, and the results we report here are based on segmenting each numerical variable into 4 segments, as this gave the best results among all of the tested methods.

Statistics for this experiment are reported in Table I. As our goal is to find redescriptions that hold well in the unseen data, we measure the quality of the generalization using the average ratio of the accuracy of the redescriptions in the training data to their accuracy in the full data. If this ratio is substantially below 1, we conclude that the redescriptions did not generalize well.

On Bio$_S$, `SplitT` and `CARTwheels` achieve essentially the same quality, both returning high-accuracy redescriptions that generalize well. `CARTwheels`' larger number of redescriptions is due to the fact that it returned multiple almost-identical redescriptions. `LayeredT` also reported redescriptions that generalize well, although their overall accuracy was lower than with the other methods, while `ReReMi` apparently overfitted. On Bio, we observe similar behaviors from `SplitT`, `LayeredT`, and `ReReMi` (`CARTwheels` could not be used on this data) as in Bio$_S$, with `SplitT` giving the best overall results.

On the DBLP data, `SplitT` is the only algorithm returning results that are accurate and also generalize well, although most of its redescriptions had lower supports than the redescriptions returned by the other methods: `LayeredT` returns redescriptions with very low accuracy, and while `ReReMi`'s accuracy is better on the training data, it again overfits and has very low generalizability.

6.4. Scalability

In our final experiment, we studied how well `Siren`'s parallel redescription mining framework scales to multiple processors. Recall from Section 4.4 that `Siren` can distribute the computation of the initial pairs and of candidate expansions to different threads. Meanwhile, the main thread collects the redescriptions produced and filters them. This adds some overhead to the otherwise embarrassingly parallel workload, and the purpose of the experiments in this section is to study the effects of that overhead.

In this experiment, we use five datasets. The first two are variants of the DBLP dataset with both sides consisting of either Boolean or numerical variables, respectively. The next two datasets are the full Bio and the Globe datasets, both with Boolean variables on one side and numerical variables on the other. The last one is the Cover dataset, with numerical variables on one side and mostly Boolean variables on the other. We applied the `ReReMi` and `SplitT` algorithms on each dataset in turn, with 1, 2, 4 and 8 threads, while limiting the number of candidates expanded to 500 for all but the Globe dataset, which contains a large number of variables and for which we thus set this threshold to 1000.

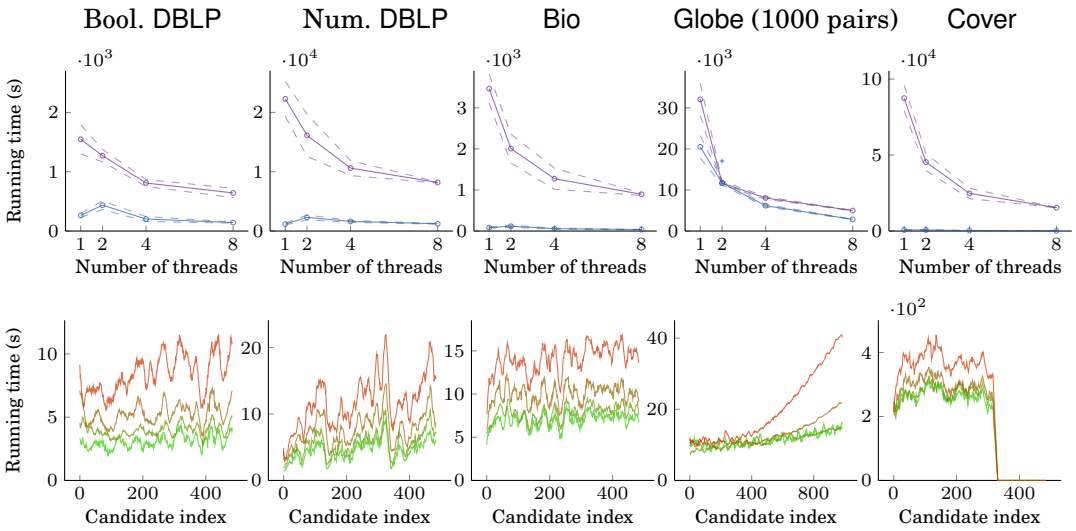


Fig. 16. Running times for the ReReMi algorithm with increasing numbers of threads. The top row of plots shows the total running time and the running time for generating initial pairs (purple and blue curves respectively) with standard deviation interval. The bottom row of plots shows the running time per candidate expansion as a function of the index of the candidate in the processing queue, averaged over a sliding window of 15 candidates, from using a single thread (green curve) to using 8 threads (red curve).

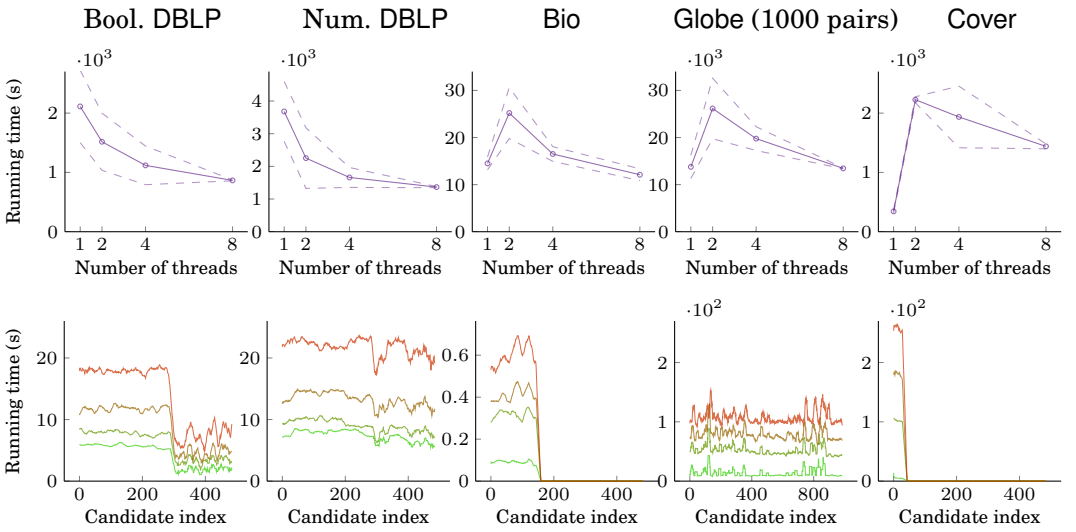


Fig. 17. Running times for the SplitT algorithm with increasing numbers of threads. The top row of plots shows the total running time with standard deviation interval. The bottom row of plots shows the running time per candidate expansion as a function of the index of the candidate in the processing queue, averaged over a sliding window of 15 candidates, from using a single thread (green curve) to using 8 threads (red curve).

Figure 16 shows the running times for applying the ReReMi algorithm with an increasing number of threads. The top row of plots shows the total running time and the running time for generating initial pairs (purple and blue curves respectively) with the respective standard deviation intervals. The bottom row of plots, on the other

hand, shows the running time per candidate expansion as a function of the index of the candidate, that is, depending at which stage of the algorithm execution the candidate is handled. The green curve corresponds to using a single thread and the red curve to using 8 threads, with intermediate cases in between. To remove minor variations, we averaged these values with a sliding window of 15 candidates.

Similarly, Figure 17 shows the running times for applying the SplitT algorithm with an increasing number of threads. Note that in this case, we do not report the running times for computing the initial target vectors. This is because this computation, done for each suitable variable separately, is already very efficient and therefore not distributed.

In Figure 16 we notice that increasing the number of threads clearly speeds up the mining process overall. As could be expected, however, the benefits of adding new threads tend to diminish as the number of threads increases. Looking at the running time per candidate we can see a trend, most noticeable for the Boolean DBLP and the Globe datasets, where later candidates require more time than earlier ones. This is mainly due to the filtering performed by the main thread, which requires to compare the newly built redescription to previously retained results. As the mining advances, the number of those results increases and the filtering becomes slower. For the initial pairs, using two threads rather than only one requires to divide the task without adding much computing power, hence the latter is actually more efficient. This can be reversed by adding further threads. An important benefit of using multiple threads is that the first initial pairs can already start being expanded and redescriptions obtained before the initial pair computation completes. Hence, the first results can be returned much earlier to the user.

This is particularly visible with the Globe dataset (see Figure 16, top row, second column on the right) where the number of variable pairs is very large and their computation takes up a major part of the runtime. In fact, with two threads the expansion of 1000 candidates pairs completes before all variables pairs have been tested. In this case, the initial pair generation is interrupted to avoid generating pairs needlessly. On the plot, we indicate with a cross the runtime of the full initial pair generation when using two threads. Depending on cases, it could be interesting to distribute the load of initial pairs generation and candidates expansion differently among the available threads.

Having multiple threads is even more useful with the Cover dataset (see Figure 16, top row, first column on the right). There, the computation of initial pairs is very fast, since the number of variables is small, but the computation of individual expansions is quite demanding, due to the very large number of entities. Therefore, computing the candidates in a distributed fashion is very beneficial.

When applying the SplitT algorithm on the Bio dataset, the number of initial vectors is limited by the number of Boolean variables, hence there are only few candidates to grow, the computation completes quickly, and the addition of threads does not help much, if at all. For both variants of the DBLP dataset, however, this is not the case and the effect of adding threads is similar to the one observed with the ReReMi algorithm. For the Boolean variant, we see a clear drop in the running time per candidate after about the first 300 candidates. This is because the candidates are processed starting with the most promising ones, at some point the candidates cease to produce expansions of acceptable quality and the growing procedure is cut short.

Similarly, with the Cover dataset the SplitT algorithm has very few candidates to expand (at most one per variable) and hence distributing their computation comes with a high overhead but no benefit at all.

In summary, we observed that the use of multiple processors can produce substantial speed-ups in many cases and especially with the ReReMi algorithm, so that the overall mining only requires half of the time needed with a single thread or even less. Clearly, this helps improve the responsiveness of Siren.

7. DISCUSSION AND RELATED WORK

In this section, we first briefly point out some earlier interactive data mining tools. We then discuss issues regarding the manipulation of sets of redescriptions, the selection of good patterns, provenance and sensemaking, meanwhile providing directions for future work.

7.1. Existing tools for interactive pattern mining

The Knowledge Factory, presented by Webb [1996] as a knowledge acquisition environment, has possibly been the first attempt to design an interface for a data mining task, in this case association rule learning. A more recent notable example is the KNIME system [Berthold et al. 2009], allowing users to build data analysis workflows. Paurat et al. [2014] presented a method for interactively creating 2D embeddings, while MIME [Goethals et al. 2011] is a tool for interactive and visual pattern mining. Apolo [Chau et al. 2011] and TourViz [Chau et al. 2012], on the other hand, are both dedicated to the interactive analysis of graphs. Very recently, Mihelčić and Šmuc [2016] proposed InterSet, a tool for the interactive exploration of sets of redescriptions.

7.2. Handling sets of redescriptions

InterSet [Mihelčić and Šmuc 2016] provides visualizations and an interface for working with sets of redescriptions. Incorporating a similar capability to `Siren` would be useful, as well. The ability to visualize and compare several redescriptions is important for exploring the results of full and partial automated mining, for supporting the process of redundancy filtering, as well as when tuning the settings and examining the effect of different constraints.

In particular, redescription mining often suffers from *redundant* results, where many redescriptions redescriptions describe the same phenomenon. Redundancy filters can be used to automatically weed through a list of redescriptions and select non-redundant ones based on support and variables overlap. Highlighting differences in supports using projections, and differences in conditions on the variables using parallel coordinate plots, would support the understanding of these filters. Recent work by Kalofolias et al. [2016] presents an approach fo computing how *surprising* a redescription is given the already-seen redescriptions; if some redescription is not sufficiently surprising, it can be considered redundant and removed.

In addition, at a higher level, contrasting the statistics of different collections of redescriptions would also facilitate the comparison of results obtained under different constraints, with some areas emphasized, with some variables enabled or disabled, and so on, thereby allowing to better understand the effect of different parameterizations.

7.3. Pattern selection and associated pitfalls

Recently, research interest has arisen at the crossroads of interactive data mining and pattern selection [Boley et al. 2013; van Leeuwen 2014; Miettinen 2014].

At the heart of interactive data mining is the user's ability to tell the algorithm that he wants more or less of a certain type of results. In principle, this is not a problem in `Siren`: the user could simply select a redescription he wants to remove from the beam search or to extend further. The problem, however, is that there can be (and usually are) other, similar redescriptions that the user might also want to remove or extend. He can do that manually, of course, but with large numbers of redescriptions, the process becomes unbearably tedious very soon.

A solution to this problem would be to remove (or extend) all similar redescriptions. But how should similarity be defined? To give an example, consider a case where the user finds a redescription saying that the area where the polar bear lives is the area

with January's mean temperature below $-20\,^{\circ}\mathrm{C}$, in other words, polar bears live in a cold climate. This is hardly a surprising result, and the user might want to remove it (and other similar results) from the search. But we can characterize the cold areas using other variables than just January's mean temperature, so it is not enough to just stop extending any redescription with the polar bear and January's mean temperature in it. Also, we cannot just remove all the redescriptions with the polar bear – that could remove some very interesting redescriptions, too. Finally, we could consider the area in which the redescription holds. But even that leaves a lot to be hoped for: If we remove all redescriptions that contain that area, we probably remove too many redescriptions, but if we instead remove redescriptions contained in the area, we probably miss most of the redescriptions we should remove.

The problem of removing and extending similar redescriptions is closely related to that of redundancy reduction. There are often multiple redescriptions that represent the same phenomenon (think of the polar bear living in cold areas), and ideally, we would like to present only one of them to the user. In other words, we do not want to present to the user any redescription that does not add any (or add only marginally) new information over the redescriptions he has already seen. But as with deciding which redescription is similar to a selected one, also quantifying the redundancy between redescriptions is a difficult problem.

The goal of data mining is to find new and interesting information from the data. In interactive data mining in general, and with the tools discussed in this paper in particular, the user can guide the data mining method towards the results he prefers. This raises new problems. First, we have to control that the data supports the results the user finds and second, we must be careful that the user actually finds new information, not just the information he already knew.

The first problem, making sure that the obtained results are supported by the data, is ages old in sciences. In short, it is the question of testing the significance of a hypothesis, and there is a vast body of statistical literature tackling it. Our proposed algorithms mitigate the problem by computing a p-value, but as it is based on a fixed null hypothesis, it is not adequate in every case.

The second problem is more conceptual: Taken to an extreme, the interactivity removes the data mining from the interactive data mining. If the user more or less directly tells the algorithm the redescription he wants to see, the `Siren` program turns into a mere plotting interface. Even on the less extreme case, the user can easily (and unwittingly) guide the algorithm towards the kind of results he wanted to see. Together with the fact that we can only check against a fixed null hypothesis, this causes a considerable risk of false findings. The onus is on the user to make sure he does not misuse the tool.

7.4. Provenance and sensemaking

The term 'sensemaking' denotes the process by which the user of a visual tool makes sense of information presented to him. Originating in concepts developed in the cognitive sciences, recent works aim at studying and modeling this workflow, in order to adapt interactive tools to better support it [Endert et al. 2014a; Endert et al. 2014b; Sun et al. 2014; Pienta et al. 2015].

In the context of data analytics, 'provenance' has been used to refer to the history of changes made to the data and interactions with the interface that occur during the sensemaking process [Ragan et al. 2016]. Authors have argued that better logging could help elucidate the sensemaking process [Guo et al. 2016] and improve the analytics tools [Alspaugh et al. 2014].

Indeed, when interpreting a redescription, one should bear in mind the assumptions attached to it. For example, whether some variables were disabled or whether the focus

was put on some area when it was generated. Hence, keeping track of the constraints used when mining a redescription is essential. However, if the user is allowed to stop the extension process, modify the constraints, and resume the search, this might be fairly intricate and interpretation of the results become impossible.

Recording the interaction history would provide a logging mechanism that would support provenance, by keeping track of the operations applied to the data, and of the settings that gave rise to the results. Furthermore, in order to support the sensemaking process, the tool should support annotation in order to keep track of the thought path during the analysis. For example, this could be achieved by generating annotable screenshots of the current window of interest, and by adding comments to the interaction history and macros. Organizing the history and macros into blocks would help to further clarify the logical structure of the analysis. In addition, with the ability to link to objects in the current environment – such as redescriptions, groups of entities, or literals – these could be explicitly related to each other. Exporting and importing easily such annotations and macros is a very important feature to support collaboration and deferred interpretation of the results.

Currently, as a preliminary solution, a history of edits made to redescriptions is kept in the form of a list of intermediate results. Also, the active settings can be saved alongside the data and results in dedicated archives, but nothing guarantees that all the results in the archive were obtained with the stored settings. Clearly, much work remains to be done in this direction. To be relevant, such features should best be developed in close collaboration with a community of users of the tool, a development process that comes with its own set of challenges.

8. CONCLUSION

Our contributions in this work are two-fold. On one hand, we presented the `Siren` tool for interactive and visual redescription mining. On the other hand, we also presented two new algorithms for redescription mining based on decision tree induction. `Siren` provides a framework to which new visualizations and redescription mining algorithms can be added easily. Its state-of-the-art features, such as linked and interactive visualizations, have proven to be very useful in practice. The new tree-based algorithms are capable of finding different types of redescriptions compared to our older `ReReMi` algorithm. Subjectively, the redescriptions found with these methods are not clearly superior to those found with `ReReMi`, but our experiments indicate that the tree-based methods are capable of finding redescriptions that generalize better to unseen data.

Further work is needed to properly assess the usefulness of the various visualizations in `Siren`, either via a controlled study, or by gathering user experiences from real-world use cases. The users' ability to control the mining process is somewhat limited, and a possibility to indicate a preference (or lack thereof) toward a particular type of results could open up new interaction methods, as we argued in the previous section. In general, the current redescription mining algorithms find large redescriptions sets, while it is probably preferable to aim at finding a good set of redescriptions instead.

REFERENCES

Sara Alspaugh, Archana Ganapathi, Marti Hearst, and Randy Katz. 2014. Better Logging to Improve Interactive Data Analysis Tools. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA '14)*. 19–25. Retrieved 23 October 2016 from http://poloclub.gatech.edu/idea2014/

A. Annoni, Lars Bernard, Arvid Lillethun, Johannes Ihde, Javier Gallego, Michael Rives, Erik Sommer, Hugo Poelman, Sophie Condé, Mark Greaves, Pertti Uotila, Jorge Teixeira Pinto, João Agria Torres, Raino Lampinen, Maxime Kayadjanian, Volker Schmidt-Seiwert, Andrus Meiner, Claude Luzet, and Albrecht Wirthmann. 2004. *Short Proceedings of the 1st Workshop on European Reference Grid*. JRC-Institute for Environment and Sustainability, Ispra, Italy.

Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. 2009. KNIME — the Konstanz Information Miner: Version 2.0 and Beyond. *SIGKDD Explor. Newsl.* 11, 1 (2009), 26–31. DOI:http://dx.doi.org/10.1145/1656274.1656280

Mario Boley, Bo Kang, Pavel Tokmakov, Michael Mampaey, and Stefan Wrobel. 2013. One Click Mining — Interactive Local Pattern Discovery through Implicit Preference and Performance Learning. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA '13)*. ACM, New York, 27–35. DOI:http://dx.doi.org/10.1145/2501511.2501517

Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 1984. *Classification and regression trees*. Wadsworth International Group, Belmont, CA.

Duen Horng Chau, Leman Akoglu, Jilles Vreeken, Hanghang Tong, and Christos Faloutsos. 2012. TourViz: Interactive Visualization of Connection Pathways in Large Graphs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, New York, 1516–1519. DOI:http://dx.doi.org/10.1145/2339530.2339769

Duen Horng Chau, Aniket Kittur, Jason I. Hong, and Christos Faloutsos. 2011. Apolo: Making Sense of Large Network Data by Combining Rich User Interaction and Machine Learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, 167–176. DOI:http://dx.doi.org/10.1145/1978942.1978967

Thomas Dean and Mark Boddy. 1988. An Analysis of Time-Dependent Planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI '88)*. MIT Press, Cambridge, 49–54.

Alex Endert, M. Shahriar Hossain, Naren Ramakrishnan, Chris North, Patrick Fiaux, and Christopher Andrews. 2014a. The human is the loop: new directions for visual analytics. *J. Intell. Inf. Syst.* 43, 3 (2014), 411–435. DOI:http://dx.doi.org/10.1007/s10844-014-0304-9

Alex Endert, Chris North, Remco Chang, and Michelle Zhou. 2014b. Toward Usable Interactive Analytics: Coupling Cognition and Computation. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA '14)*. 52–56. Retrieved 23 October 2016 from http://poloclub.gatech.edu/idea2014/

Daria Gaidar. 2015. *Mining redescriptions in S. aureus data*. Master's thesis. Saarland University.

Esther Galbrun and Pauli Miettinen. 2012a. A Case of Visual and Interactive Data Analysis: Geospatial Redescription Mining. In *Proceedings of the ECML-PKDD workshop on Instant Interactive Data Mining*. Retrieved 23 October 2016 from http://adrem.ua.ac.be/iid2012/

Esther Galbrun and Pauli Miettinen. 2012b. From black and white to full color: Extending redescription mining outside the Boolean world. *Stat. Anal. Data Min.* 5, 4 (2012), 284–303. DOI:http://dx.doi.org/10.1002/sam.11145

Esther Galbrun and Pauli Miettinen. 2012c. Siren: An Interactive Tool for Mining and Visualizing Geospatial Redescriptions. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, New York, 1544–1547. DOI:http://dx.doi.org/10.1145/2339530.2339776

Esther Galbrun and Pauli Miettinen. 2014. Interactive redescription mining. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*. ACM, New York, 1079–1082. DOI:http://dx.doi.org/10.1145/2588555.2594520

Esther Galbrun and Pauli Miettinen. 2016. Analysing Political Opinions Using Redescription Mining. In *Proceedings of the 2016 IEEE International Conference on Data Mining Workshop (ICDMW '16)*. IEEE, Los Alamitos. To appear.

Adrianna Gallo, Pauli Miettinen, and Heikki Mannila. 2008. Finding Subgroups having Several Descriptions: Algorithms for Redescription Mining. In *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM '08)*. SIAM, Philadelphia, 334–345. DOI:http://dx.doi.org/10.1137/1.9781611972788.30

Neha Goel, Michael S. Hsiao, Narendran Ramakrishnan, and Mohammed J. Zaki. 2010. Mining Complex Boolean Expressions for Sequential Equivalence Checking. In *Proceedings of the 19th IEEE Asian Test Symposium (ATS '10)*. IEEE, Los Alamitos, 442–447. DOI:http://dx.doi.org/10.1109/ATS.2010.81

Bart Goethals, Sandy Moens, and Jilles Vreeken. 2011. MIME: A framework for interactive visual pattern mining. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*. ACM, New York, 757–760. DOI:http://dx.doi.org/10.1145/2020408.2020529

Joseph Grinnell. 1917. The Niche-Relationships of the California Thrasher. *The Auk* 34, 4 (1917), 427–433. DOI:http://dx.doi.org/10.2307/4072271

Hua Guo, Steven R. Gomez, Caroline Ziemkiewicz, and David H. Laidlaw. 2016. A Case Study Using Visualization Interaction Logs and Insight Metrics to Understand How Analysts Arrive at Insights. *IEEE Trans. Vis. Comput. Graphics* 22, 1 (Jan. 2016), 51–60. DOI:http://dx.doi.org/10.1109/TVCG.2015.2467613

Jeffrey Heer and Ben Shneiderman. 2012. Interactive Dynamics for Visual Analysis. *Commun. ACM* 55, 4 (April 2012), 45–54.

Robert J. Hijmans, Susan E. Cameron, Juan L. Parra, Peter G. Jones, and Andy Jarvis. 2005. Very High Resolution Interpolated Climate Surfaces for Global Land Areas. *Int. J. Climatol.* 25 (2005), 1965–1978.

Alfred Inselberg. 2009. *Parallel coordinates: Visual multidimensional geometry and its applications*. Springer, New York. DOI:http://dx.doi.org/10.1007/978-0-387-68628-8

IUCN. 2014. The IUCN Red List of Threatened Species. Version 2014.1. (2014). Retrieved 3 July 2016 from http://www.iucnredlist.org

Janis Kalofolias, Esther Galbrun, and Pauli Miettinen. 2016. From Sets of Good Redescriptions to Good Sets of Redescriptions. In *Proceedings of the 16th IEEE International Conference on Data Mining (ICDM '16)*. IEEE, Los Alamitos. To appear.

Deept Kumar. 2007. *Redescription mining: Algorithms and applications in bioinformatics*. Ph.D. Dissertation. Virginia Polytechnic Institute and State University, Department of Computer Science, Virginia Tech.

A. Michelle Lawing, Jussi T. Eronen, Jessica L. Blois, Catherine H. Graham, and P. David Polly. 2016. Community functional trait composition at the continental scale: the effects of non-ecological processes. *Ecography* 39 (2016), 13. DOI:http://dx.doi.org/10.1111/ecog.01986

Pauli Miettinen. 2014. Interactive Data Mining Considered Harmful (If Done Wrong). In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA '14)*. 85–87. Retrieved 23 October 2016 from http://poloclub.gatech.edu/idea2014/

Matej Mihelčić and Tomislav Šmuc. 2016. InterSet: Interactive Redescription Set Exploration. In *Proceedings of the 19th International Conference on Discovery Science (DS '16)*. Springer, Cham, 35–50. DOI:http://dx.doi.org/10.1007/978-3-319-46307-0_3

A. J. Mitchell-Jones, G. Amori, W. Bogdanowicz, B. Krystufek, P. J. H. Reijnders, F. Spitzenberger, M. Stubbe, J. B. M. Thissen, V. Vohralik, and J. Zima. 1999. *The Atlas of European Mammals*. Academic Press, London.

Petra Kralj Novak, Nada Lavrač, and Geoffrey I. Webb. 2009. Supervised Descriptive Rule Discovery: A Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Mining. *J. Mach. Learn. Res.* 10 (2009), 377–403.

Gregorio Palmas, Myroslav Bachynskyi, Antti Oulasvirta, Hans-Peter Seidel, and Tino Weinkauf. 2014. An edge-bundling layout for interactive parallel coordinates. In *Proceedings of the 2014 IEEE Pacific Visualization Symposium (PacificVis '14)*. IEEE, Los Alamitos, 57–64. DOI:http://dx.doi.org/10.1109/PacificVis.2014.40

Laxmi Parida and Naren Ramakrishnan. 2005. Redescription Mining: Structure Theory and Algorithms. In *Proceedings of the 20th AAAI Conference on Artificial Intelligence (AAAI '05)*. MIT Press, Cambridge, 837–844.

Daniel Paurat, Roman Garnett, and Thomas Gärtner. 2014. Interactive Exploration of Larger Pattern Collections: A Case Study on a Cocktail Dataset. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA '14)*. 98–106. Retrieved 23 October 2016 from http://poloclub.gatech.edu/idea2014/

Richard G. Pearson and Terence P. Dawson. 2003. Predicting the Impacts of Climate Change on the Distribution of Species: Are Bioclimate Envelope Models Useful? *Global Ecol. Biogeogr.* 12 (2003), 361–371. DOI:http://dx.doi.org/10.1046/j.1466-822X.2003.00042.x

Robert Pienta, James Abello, Minsuk Kahng, and Duen Horng Chau. 2015. Scalable graph exploration and visualization: Sensemaking challenges and opportunities. In *Proceedings of the International Conference on Big Data and Smart Computing (BigComp '15)*. IEEE, Los Alamitos, 271–278. DOI:http://dx.doi.org/10.1109/35021BIGCOMP.2015.7072812

J. R. Quinlan. 1986. Induction of decision trees. *Machine Learning* 1, 1 (1986), 81–106. DOI:http://dx.doi.org/10.1023/A:1022643204877

Eric D. Ragan, Alex Endert, Jibonananda Sanyal, and Jian Chen. 2016. Characterizing Provenance in Visualization and Data Analysis: An Organizational Framework of Provenance Types and Purposes. *IEEE Trans. Vis. Comput. Graphics* 22, 1 (2016), 31–40. DOI:http://dx.doi.org/10.1109/TVCG.2015.2467551

Naren Ramakrishnan, Deept Kumar, Bud Mishra, Malcolm Potts, and Richard F. Helm. 2004. Turning CARTwheels: An alternating algorithm for mining redescriptions. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*. ACM, New York, 266–275. DOI:http://dx.doi.org/10.1145/1014052.1014083

Naren Ramakrishnan and Mohammed J. Zaki. 2009. Redescription Mining and Applications in Bioinformatics. In *Biological Data Mining*, Jake Chen and Stefano Lonardi (Eds.). Chapman & Hall/CRC, Boca Raton, 561–586.

Maoyuan Sun, Lauren Bradel, Christopher L North, and Naren Ramakrishnan. 2014. The role of interactive biclusters in sensemaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, 1559–1562. DOI:http://dx.doi.org/10.1145/2556288.2557337

Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. 2000. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* 290, 5500 (Dec. 2000), 2319–2323. DOI:http://dx.doi.org/10.1126/science.290.5500.2319

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. Mining Multi-label Data. In *Data Mining and Knowledge Discovery Handbook* (second ed.), Oded Maimon and Lior Rokach (Eds.). Springer, New York, 667–685. DOI:http://dx.doi.org/10.1007/978-0-387-09823-4_34

Lan Umek, Blaz Zupan, Marko Toplak, Annie Morin, Jean-Hugues Chauchat, Gregor Makovec, and Dragica Smrke. 2009. Subgroup Discovery in Data Sets with Multi-dimensional Responses: A Method and a Case Study in Traumatology. In *Proceedings of the 12th Conference on Artificial Intelligence in Medicine (AIME '09)*. Springer, Berlin, 265–274. DOI:http://dx.doi.org/10.1007/978-3-642-02976-9_39

Matthijs van Leeuwen. 2014. Interactive Data Exploration Using Pattern Mining. In *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*. Lecture Notes in Computer Science, Vol. 8401. Springer, Berlin, 169–182. DOI:http://dx.doi.org/10.1007/978-3-662-43968-5_9

Geoffrey I. Webb. 1996. Integrating machine learning with knowledge acquisition through direct interaction with domain experts. *Knowl.-Based Syst.* 9, 4 (1996), 253–266. DOI:http://dx.doi.org/10.1016/0950-7051(96)01033-7

Tetiana Zinchenko, Esther Galbrun, and Pauli Miettinen. 2015. Mining Predictive Redescriptions with Trees. In *Proceedings of the 2015 IEEE International Conference on Data Mining Workshop (ICDMW '15)*. IEEE, Los Alamitos, 1672–1675. DOI:http://dx.doi.org/10.1109/ICDMW.2015.123