

Reasoning about Sets using Redescription Mining

Mohammed J. Zaki
 Department of Computer Science
 Rensselaer Polytechnic Institute
 Troy, NY 12180, USA
 zaki@cs.rpi.edu

Naren Ramakrishnan
 Dept. of Computer Science
 Virginia Tech, Blacksburg
 VA 24061, USA
 naren@cs.vt.edu

ABSTRACT

Redescription mining is a newly introduced data mining problem that seeks to find subsets of data that afford multiple definitions. It can be viewed as a generalization of association rule mining, from finding implications to equivalences; as a form of conceptual clustering, where the goal is to identify clusters that afford dual characterizations; and as a form of constructive induction, to build features based on given descriptors that mutually reinforce each other. In this paper, we present the use of redescription mining as an important tool to reason about a collection of sets, especially their overlaps, similarities, and differences. We outline algorithms to mine all minimal (non-redundant) redescriptions underlying a dataset using notions of minimal generators of closed itemsets. We also show the use of these algorithms in an interactive context, supporting constraint-based exploration and querying. Specifically, we showcase a bioinformatics application that empowers the biologist to define a vocabulary of sets underlying a domain of genes and to reason about these sets, yielding significant biological insight.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications - Data Mining; I.2.6 [Artificial Intelligence]: Learning

General Terms: Algorithms.

Keywords: redescription, data mining, minimal generators, closed itemsets.

1. INTRODUCTION

Redescription mining is a recently introduced data mining problem [7] that seeks to find subsets of data affording multiple definitions. The input to redescription mining is a vocabulary of sets (or boolean propositions) over a domain and the goal is to construct two distinct expressions from this vocabulary that induce the same subset over the domain. Besides constituting a new class of patterns, we can think of redescriptions as a useful way to reason about overlaps, similarities, and differences in the given vocabulary.

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. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'05, August 21–24, 2005, Chicago, Illinois, USA.
 Copyright 2005 ACM 1-59593-135-X/05/0008 ...\$5.00.

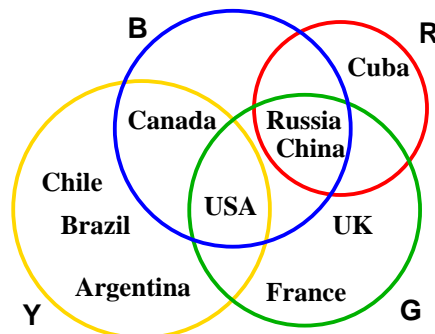


Figure 1: An example input to redescription mining.

To see how, consider the four sets shown in Fig. 1 over ten objects (in this case, countries). The colors green, red, blue, and yellow (from bottom right, counterclockwise) refer to the sets ‘permanent members of the UN security council,’ ‘countries with a history of communism,’ ‘countries with land area > 3,000,000 square miles,’ and ‘popular tourist destinations in the Americas (North and South).’ We will refer to such sets as *descriptors*.

Notice that the descriptors of Fig. 1 induce a partition containing 6 non-empty blocks although, with 10 objects, we could have had up to 10 non-empty blocks. Another way to see this deficiency is from the viewpoint of the sets: four sets can support $2^4 = 16$ different objects but there are only 10 objects and even these 10 objects are not all distinct. For instance, {Chile, Brazil, Argentina} are indistinguishable from each other, and so are the members of {UK, France}, {Russia, China}. Empty blocks in a partition are key to how redescriptions arise in a dataset.

Let us define the boolean propositions G , R , B , and Y (for ‘green,’ ‘red,’ ‘blue,’ and ‘yellow’) to denote containment in the above four sets. Consider the countries denoted by $B\bar{Y} \equiv B \wedge \neg Y$, i.e., the set of countries with land area > 3,000,000 square miles outside of the Americas (\bar{Y} and $\neg Y$ denote the negation of Y). We can systematically restate this expression as follows:

$$\begin{aligned}
 B\bar{Y} &= B\bar{Y}(RG + \bar{R}G + R\bar{G} + \bar{R}\bar{G}) \\
 &= B\bar{Y}RG + B\bar{Y}\bar{R}G + B\bar{Y}R\bar{G} + B\bar{Y}\bar{R}\bar{G} \\
 &= B\bar{Y}RG \\
 &= B\bar{Y}RG + \bar{B}YRG + BYRG + \bar{B}\bar{Y}RG \\
 &= (B\bar{Y} + \bar{B}Y + BY + \bar{B}\bar{Y})RG \\
 &= RG
 \end{aligned}$$

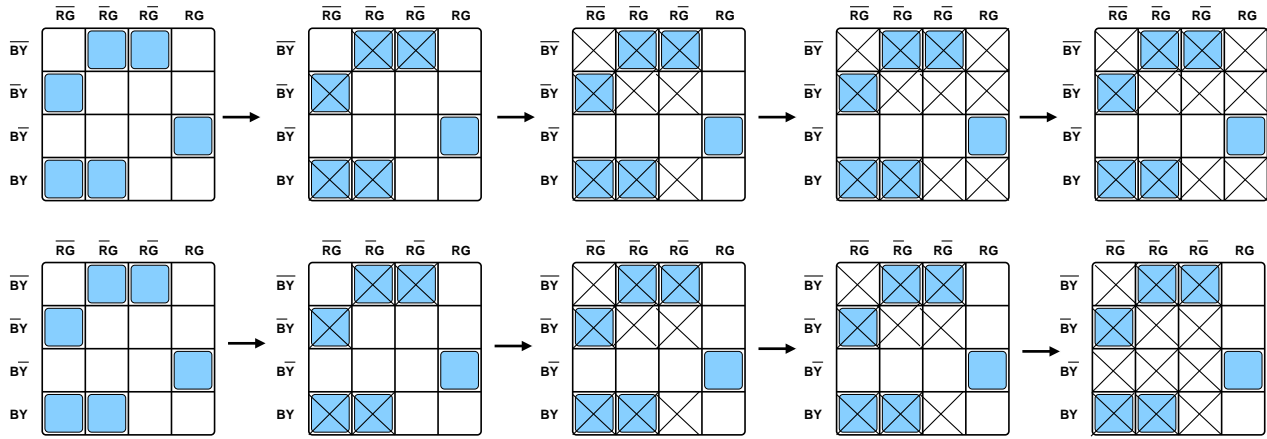


Figure 2: Obtaining redescrptions by simple operations on Karnaugh maps. The top row shows one legal sequence of moves and the bottom row shows another sequence; notice that both maps start from the same initial configuration. Removed cells are marked with a ‘x’. If the moves are different, we obtain a non-trivial redescription. In this case, the moves are (i) removing all but one colored cell from both maps, (ii) removing some uncolored cells from both maps, (iii) removing some uncolored cells from only the top map, followed by (iv) removing some uncolored cells from only the bottom map. The final maps capture the redescription between $\overline{B\overline{Y}}$ (top) and RG (bottom).

The first step is by marginalizing over the variables R and G , the second step is an application of the distributive law, the third step is data-specific, and gets rid of conjunctions that denote empty blocks, the fourth step is again data-specific but this time *introduces* empty conjunctions, which are grouped in the fifth step, and marginalized out finally. We have arrived at our first redescription:

$$\overline{B\overline{Y}} \Leftrightarrow RG$$

i.e., ‘Countries with land area > 3, 000, 000 square miles outside of the Americas’ are the same as ‘Permanent members of the UN security council who have a history of communism.’ This redescription re-defines the set {Russia, China}. A redescription is hence a shift-of-vocabulary, or a different way of communicating the same information. Redescription mining can therefore be viewed as a generalization of association rule mining, from finding implications to equivalences; as a form of conceptual clustering, where the goal is to identify clusters (here, of countries) that afford dual characterizations; and as a form of constructive induction, to build features based on given descriptors that mutually reinforce each other.

1.1 Understanding Redescrptions

An intuitive way to understand the structure of redescription space is via a simple game on Karnaugh maps. The leftmost part of Fig. 2 depicts the Karnaugh map for four boolean variables, reproduced in both rows. Each cell in the map is a conjunction over four boolean variables. A colored cell indicates a non-empty block for our example dataset in Fig. 1. For example, the cell $(\overline{B\overline{Y}}, RG)$ denotes the block {Russia, China}. We can interpret a map to be the disjunction of all its cells; since the starting maps are the same, they both represent the same 10 objects and hence constitute a trivial redescription, i.e., a tautology. The rules of the game are:

1. A colored cell can be removed as long as it is removed from both maps. Notice that this will make the maps represent fewer objects.

2. An uncolored cell can be removed from either (or both) maps. Notice that this move does not affect the number of objects represented by the maps.

Some sample moves are shown in Fig. 2. At the end of the game, we read off a redescription by relating the disjunctions of cells remaining in both maps:

$$\begin{aligned} & (\overline{B\overline{Y}R\overline{G}} \vee \overline{B\overline{Y}RG} \vee \overline{B\overline{Y}R\overline{G}} \vee \overline{B\overline{Y}RG}) \\ & \Leftrightarrow \\ & (\overline{B\overline{Y}RG} \vee \overline{B\overline{Y}RG} \vee \overline{B\overline{Y}RG} \vee \overline{B\overline{Y}RG}) \end{aligned}$$

simplification of which yields $\overline{B\overline{Y}} \Leftrightarrow RG$, as before.

The above viewpoint reveals two important insights. First, we can obtain redescrptions for *any* combination of the colored cells, by just retaining them in both rows of the game. Second, even for a single choice of these sets, there are an exponential number of redescrptions, each of which merely differs from another in the choice of uncolored cells that were retained. Interestingly, the form of expressions participating in a redescription follows in a very natural way from the subset of colored and uncolored cells that are retained.

In Fig. 2 both maps permit descriptions as conjunctions because we are retaining *all* cells in a single row or column. Fig. 3 presents situations (from different datasets) with expressions in different forms. The top left part of the figure represents a disjunction of all cells in the bottom three rows, or $B \vee Y$. The bottom left part of Fig. 3 represents a disjunction of all cells in the rightmost two columns, which is represented quite succinctly as R . These maps hence capture the redescription $B \vee Y \Leftrightarrow R$. On the other hand, the right maps denote the redescription $B \vee Y \Leftrightarrow R \vee G$. The reader can easily design examples where we can neither simplify into conjunctions or disjunctions, and must instead adopt a more general bias, such as CNF (conjunctive normal form). In this paper, due to space constraints, we restrict the discussion of redescription mining to those relating conjunctions on both sides (possibly involving negations). But the above discussion highlights how the framework can be extended to mining disjunctions and more general forms.

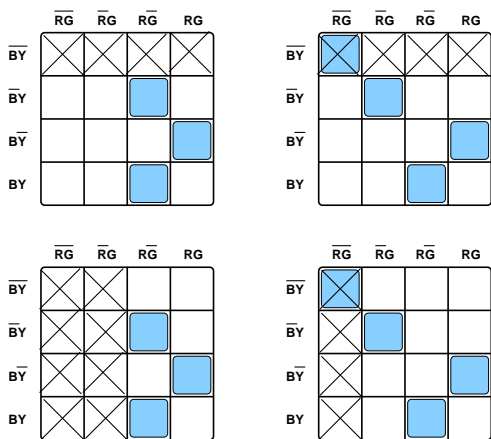


Figure 3: Examples of redescrptions in different biases. The left maps redescribe a disjunction (top) to a conjunction (bottom). The right maps redescribe a disjunction (top) into another disjunction (bottom).

1.2 Practical Usage Contexts

One reason why we adopt the conjunctions bias is its suitability for the application context of geneset exploration in bioinformatics. Here we are given \mathcal{G} , a set of genes and \mathcal{D} , a set of descriptors (gene subsets). Example descriptors are: ‘genes localized in cellular compartment nucleus,’ ‘genes up-expressed two-fold or more in heat stress,’ ‘genes encoding for proteins that form the Immunoglobulin complex,’ ‘genes involved in glucose biosynthesis,’ and even ‘genes targeted by Professor X for further study.’ It can be argued that, in the post-sequencing era, bioinformatics is suffering from an information overload of descriptors, as every scientist pursues a preferred way of identifying subsets of genes from the massive cardinality of the genome. The goal of redescription mining is to connect these diverse vocabularies, by relating set-theoretic constructs formed over the corresponding descriptors. For instance, we might find that ‘genes expressed in the desiccation experiment except those participating in universal stress response’ are the same as ‘genes significantly expressed 2-fold positively or negatively in the salt stress experiment.’ The underlying premise is that genesets that can indeed be defined in (at least) two ways are likely to exhibit concerted behavior and are, hence, interesting.

Typically a biologist would like to study gene sets by first identifying a focus set of genes and then systematically slicing and dissecting the set, to answer questions such as:

- What redescrptions do a given set of genes participate in? What redescrptions does the presence of a given descriptor induce?
- Starting with two dissimilar (not disjoint) sets of genes, say X and Y , how can we systematically remove elements from both sides, so that we obtain a redescription? Note that we are only allowed to discard elements that correspond to unions of cells (colored or uncolored) in the Karnaugh map.
- Are there genesets that cannot be redescribed into each other through systematic removals of elements? If so, what is the best approximation that can be achieved?
- Since redescrptions induce equivalence classes over the

space of possible descriptor expressions, what are the ‘densest’ such classes?

Because systematic projection of elements out of a set involves either set intersection or set differences, we can address all of the above questions by mining redescrptions between conjunctions of descriptors.

1.3 Connections to Association Rule Mining

The astute reader would have noticed the connections between redescription mining and association rule mining. If we think of objects (e.g., countries, genes) as transactions and descriptors (including their negations) as items, then a colored cell in the Karnaugh map corresponds to a *closed* itemset from the association mining literature [11]. This is because each cell is minimal in its contents (transactions) but maximal in its use of descriptors (items). In particular, all such closed sets will contain *all* descriptors, in either negated or non-negated form¹. A reducible row or column or submatrix of the Karnaugh map with some uncolored as well as colored cells corresponds to a non-closed itemset. For instance, the disjunction of all four cells in the third row, corresponding to $B\bar{Y}$, can also be represented by just the fourth cell, i.e., $B\bar{Y}RG$. The combination of descriptors $B\bar{Y}$ is hence not closed (and its closure is given by $B\bar{Y}RG$). A row or column or submatrix of the Karnaugh map with only colored cells that is reducible to conjunctive form (e.g., the first two columns in the last row of Fig. 2) is also a closed itemset; this will undoubtedly cover more transactions and be described by fewer items. With these observations, we can effectively relate our goal of mining redescrptions in conjunctive form to the task of mining closed itemsets (descriptor sets) and then obtaining all reducible submatrices of the Karnaugh map (again, restricting our bias to conjunctions) to yield redescrptions. It is interesting that this natural extension of the association rule framework (from implications to equivalences) has not been studied before (The reader should keep in mind that the Karnaugh map metaphor is used in this paper primarily as a conceptual tool to understand redescription spaces and that our algorithms do not explicitly reason with cells of the Karnaugh map, as outlined here.)

1.4 Contributions of this Paper

We hasten to add that such a generalization of the association rule framework is not as easy as it appears. First, the datasets for redescription mining are highly dense. Since each gene participates in either a descriptor or its negation, the datasets are exactly 50% sparse. Studying redescrptions within this context poses a unique set of challenging problems. One of our contributions is that we explain how we can curtail the complexity by adopting a constraint-based approach where we study the lattice of closed descriptor sets only around genes or descriptors of interest.

Our second contribution is a precise theoretical formulation for a basis for all redescrptions using the notion of minimal generators of closed descriptor sets. We formulate three classes of redescrptions: (i) exact, (ii) conditional, and (iii) approximate. They differ from the viewpoint of

¹Throughout this paragraph, it is important to remember that *both* the given descriptors and their negations constitute the items; without the negations, the analogies in this paragraph will not hold.

Jaccard’s coefficient. The Jaccard’s coefficient \mathcal{J} between two sets X and Y is the ratio of the size of their intersection to the size of their union, i.e., $\frac{|X \cap Y|}{|X \cup Y|}$. An exact redescription has $\mathcal{J} = 1$. An approximate redescription has $\mathcal{J} < 1$. Notice that this can happen when either X or Y is a subset of the other and also when the two sets straddle, i.e., $X - Y \neq \{\} \neq Y - X$. A conditional redescription can be viewed as originating as an approximate redescription but which has been transformed into an exact redescription by supplying further information. For instance, the redescription $X \Leftrightarrow Y$ could have $\mathcal{J} < 1$ but $X \cap Z \Leftrightarrow Y \cap Z$ might hold at $\mathcal{J} = 1$. In other words, conditional on Z , X and Y can be redescribed into each other. We write such redescriptions as $X \Leftrightarrow Y|Z$. Note that approximate redescriptions correspond to the minimal non-redundant exact or inexact rules described in [11], and thus we focus in this paper only on exact and conditional redescriptions.

Finally, the algorithmic approach proposed here also differs significantly from CARTwheels, an alternating algorithm for mining redescriptions presented in [7]. For instance, these two algorithms employ different biases: CARTwheels uses a disjunction of conjunctions bias, with length restrictions on the size of the expression (determined by the size of the CARTs used in the alternation), whereas our approach uses only a conjunctions bias (for this reason, it is not possible to compare the results of the two algorithms). More importantly, CARTwheels’s stochastic search policy can mine all redescriptions in a dataset only at the expense of redundancy (i.e., revisiting some redescriptions again and again). It is hence not suited for interactive and responsive analysis of very large-scale datasets. Our proposed framework overcomes these drawbacks; we showcase its application to studying the transcriptome of the yeast *S. cerevisiae* with public-domain datasets (taken from [7]).

2. FORMAL CONCEPTS

Let $\mathcal{D} = \{d_1, d_2, \dots, d_n\}$ be a set of binary-valued attributes or *descriptors*, and let $\mathcal{G} = \{g_1, g_2, \dots, g_m\}$ be a set of genes. Without loss of generality, we assume that, discounting identical genes, $m < 2^n$ (if $m = 2^n$ there can be no redescriptions, since all cells are colored; m cannot be greater than 2^n since n descriptors can only afford that much variability). A dataset \mathcal{S} is then a subset of $\mathcal{G} \times 2^{\mathcal{D}}$ (note that $2^{\mathcal{D}}$ denotes the power-set of \mathcal{D} , i.e., the set of all subsets of \mathcal{D}); in other words, the dataset \mathcal{S} is a set of tuples of the form (g, X) , where $g \in \mathcal{G}$ is a gene, and $X \subseteq \mathcal{D}$ a set of descriptors describing the given gene g . As mentioned in the introduction, the descriptors can be defined over many vocabularies, such as gene expression, functional categorization. A subset of genes $G \subseteq \mathcal{G}$ is also called a *geneset*, and a subset of descriptors $X \subseteq \mathcal{D}$ is also called a *dset*. For example, consider the dataset shown in Table 1. Here we have six genes and seven descriptors; gene g_1 participates in the descriptors $\{d_1, d_2, d_4, d_5, d_6\}$. In what follows, we omit set notation for convenience and instead represent the descriptors as a conjunction of boolean propositions as shown in Table 1.

For a dset $X \subseteq \mathcal{D}$, we denote its corresponding geneset as $g(X)$, i.e., the set of all genes described by X . For a geneset Y , we denote its corresponding dset as $d(Y)$, i.e., the set of descriptors common to all the genes in Y . The composition of the two functions, namely, g that maps from dsets to genesets, and d that maps from genesets to dsets, is called

Table 1: Sample dataset.

Gene	Descriptors
g_1	$d_1 d_2 d_4 d_5 d_6$
g_2	$d_2 d_3 d_5 d_7$
g_3	$d_1 d_2 d_4 d_5 d_6$
g_4	$d_1 d_2 d_3 d_5 d_6 d_7$
g_5	$d_1 d_2 d_3 d_4 d_5 d_6 d_7$
g_6	$d_2 d_3 d_4$

a *closure operator* [3], given as $c(X) = d(g(X))$. A dset X is said to be *closed* if and only if (iff) $c(X) = X$ [3, 11]. In other words, dset X must be a fixed point of the closure operator. For instance, $X = d_1 d_5$ is not closed since $c(d_1 d_5) = d(g(d_1 d_5)) = d(g_1 g_3 g_4 g_5) = d_1 d_2 d_5 d_6$. On the other hand $d_1 d_2 d_5 d_6$ is closed since $c(d_1 d_2 d_5 d_6) = d(g(d_1 d_2 d_5 d_6)) = d(g_1 g_3 g_4 g_5) = d_1 d_2 d_5 d_6$. Equivalently, a dset X is closed if there exists no proper superset $Y \supset X$ with $g(X) = g(Y)$. Note that a dset X is said to be frequent iff its corresponding geneset has enough cardinality, i.e., iff $|g(X)| \geq \text{minsup}$, where *minsup* is some user specified threshold.

Let X be a closed dset. We say that a dset Y is a *generator* of X iff 1) $Y \subseteq X$, and 2) $g(Y) = g(X)$. Equivalently, Y is a *generator* of X if $c(Y) = X$. Y is called a *proper generator* iff $Y \subset X$. A proper generator cannot be closed, since by definition, no closed subset of X can have the same geneset as X . We say that $Y \subset X$ is a *minimal generator* [1] of X iff Y is a proper generator of X , and there does not exist another proper generator $Z \subset Y$ of X .

Note that a closed dset X *maximally describes* its corresponding geneset $G = g(X)$, i.e., X represents the maximal set of descriptors describing the maximal set of genes G ; no other descriptor can be added to X to describe the same geneset G , and no other gene can be added to G without changing the dset X (i.e., without removing some descriptors from X). As mentioned before, in Fig. 2 this corresponds to colored cells and clusters of colored cells that are reducible (into a conjunctive form). On the other hand, a minimal generator of X , say Y , *minimally describes* the same geneset G , since by definition $g(X) = G = g(Y)$, and since Y is minimal, no descriptor can be removed from Y , and yet describe the same geneset G . In Karnaugh map terminology, a minimal generator is a submatrix containing both colored and uncolored cells that can be simplified to a conjunction representing only the colored cells.

Figure 4 shows all the 10 closed dsets along with their genesets and their minimal generators, for our example dataset, arranged in a lattice (i.e., a link exists between two closed dsets X and Y iff there does not exist another closed dset Z such that $X \subset Z \subset Y$). For example the dset $X = d_1 d_2 d_4 d_5 d_6$ maximally describes the geneset $G = g_1 g_3 g_5$, and its minimal generators are, $Y_1 = d_1 d_4$, $Y_2 = d_4 d_5$, and $Y_3 = d_4 d_6$.

DEFINITION 2.1. *Let $X, Y, Z \subseteq \mathcal{D}$ be dsets, and let $G \subseteq \mathcal{G}$ be a geneset. A conditional redescription for a geneset G is a rule of the form $G : (X \Leftrightarrow Y)|Z$, such that i) $X \neq \emptyset$ and $Y \neq \emptyset$, ii) $X \cap Y = X \cap Z = Y \cap Z = \emptyset$, iii) $g(X \cup Z) = g(Y \cup Z) = G$. Here the dset Z is called the condition. The rule means that the dsets X and Y are equivalent or describe the same geneset G given dset Z . If $Z = \emptyset$, then the rule is simply an (unconditional) redescription for geneset G ; in this case we get the simpler conditions: i) $X \neq \emptyset$ and $Y \neq \emptyset$, ii) $X \cap Y = \emptyset$ and iii) $g(X) = g(Y) = G$.*

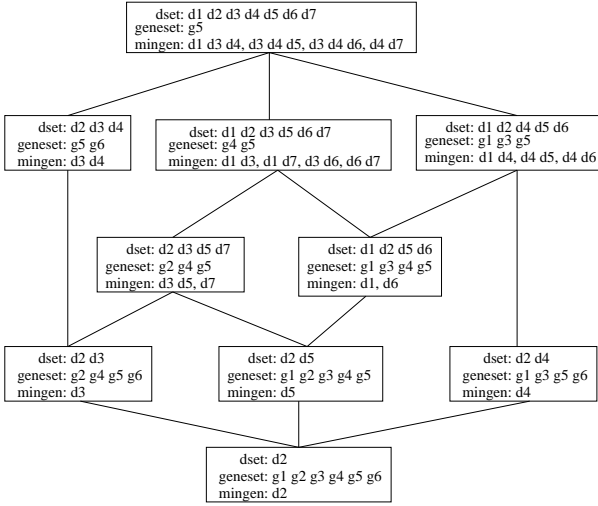


Figure 4: Closed dsets, genesets, and minimal generators.

Table 2: Non-redundant redescrptions.

Geneset	Redescription
$g_1 g_3 g_4 g_5$	$d_1 \iff d_6$
$g_2 g_4 g_5$	$d_3 d_5 \iff d_7$
$g_1 g_3 g_5$	$d_1 \iff d_5 d_4$ $d_5 \iff d_6 d_4$ $d_1 \iff d_6 d_4$
$g_4 g_5$	$d_1 d_3 \iff d_6 d_7$ $d_1 d_7 \iff d_3 d_6$ $d_3 \iff d_7 d_6$ $d_3 \iff d_7 d_1$ $d_1 \iff d_6 d_3$ $d_1 \iff d_6 d_7$
g_5	$d_1 \iff d_5 d_3 d_4$ $d_1 \iff d_6 d_3 d_4$ $d_5 \iff d_6 d_3 d_4$ $d_1 d_3 \iff d_7 d_4$ $d_3 d_6 \iff d_7 d_4$ $d_3 d_5 \iff d_7 d_4$

DEFINITION 2.2. Let $X, Y, Z \subseteq \mathcal{D}$ and let $G \subseteq \mathcal{G}$. $G : X \iff Y|Z$ is a minimal (conditional) redescription for geneset G iff there **does not exist** another redescription of the same geneset $G : X' \iff Y'|Z'$, such that $Z \subseteq Z'$ and $X' \subseteq X$ and $Y' \subseteq Y$. A minimal redescription is also called a non-redundant redescription.

In other words a non-redundant redescription describes a (maximal) geneset using a minimal number of descriptors. Table 2 shows all the non-redundant redescrptions for the given genesets in our example database. Those genesets not shown, have no redescrptions. For example, there are no redescrptions involving the geneset $G = g_5 g_6$. It is minimally described by the dset $Y = d_3 d_4$, which is a minimal generator of the maximal descriptor set (i.e., closed) $d_2 d_3 d_4$ for G .

3. ALGORITHMS

We now turn to efficient algorithms for mining the set of all non-redundant redescrptions. The process requires three

main steps: 1) mining the lattice of closed dsets from a given dataset, 2) computing the minimal generators of the closed dsets, and 3) non-redundant redescription mining from the minimal generators. We detail each step below.

3.1 Constructing Closed Dset Lattice

To generate the minimal redescrptions, we need to construct the lattice of dsets. However, current closed set mining algorithms such as Closet+ [9], Mafia [2], and Charm [12] do not output the lattice explicitly; their output is simply a list of all the closed sets found. It is possible to generate the lattice from a collection of closed sets \mathcal{C} , but unfortunately, lattice construction has time complexity $O(|\mathcal{C}|^2)$ [5], which is too slow for a large number of closed dsets.

We decided to extend Charm to directly compute the lattice while it generates the closed dsets. The basic idea is that when a new closed set X is found, we efficiently determine all its possible closed supersets, $\mathcal{P} = \{Y | Y \in \mathcal{C} \wedge X \subset Y\}$. The minimal elements in \mathcal{P} form the “immediate” supersets or parents of X in the closed dset lattice. This approach leads to a very efficient algorithm, which we call CHARM-L [13].

CHARM-L ($\mathcal{S} \subseteq \mathcal{G} \times 2^{\mathcal{D}}$):

- $[\emptyset] = \{d_i : d_i \in \mathcal{D}\}$
- CHARM-L-EXTEND ($[\emptyset], \mathcal{L}_r = \{\emptyset\}$)
- return** \mathcal{L} //lattice of closed sets

CHARM-L-EXTEND ($[P], \mathcal{L}_c$):

- for each** X_i in $[P]$ with increasing $|g(X_i)|$
- $[X_i] = \emptyset$
- UPDATE- \mathcal{C} ($X_i, [P]$)
- for each** $X_j > X_i$ in $[P]$
- $X = X_i \cup X_j, g(X) = g(X_i) \cap g(X_j)$ and $\mathcal{C}(X) = \mathcal{C}(X_i) \cap \mathcal{C}(X_j)$
- CHARM-L-PROPERTY(X, X_i, X_j)
- $\mathcal{L}_n = \text{SUBSUMPTION-CHECK-LATTICE-GEN}(\mathcal{L}_c, X_i, \mathcal{C}(X_i))$
- CHARM-L-EXTEND ($[X_i], \mathcal{L}_n$)
- delete $[X_i]$

Figure 5: The CHARM-L Algorithm.

Figure 5 gives the pseudo-code for CHARM-L(see [13] for full details). Let \mathcal{L} denote the closed dset lattice, and \mathcal{L}_r the root node of the lattice; we assume that $\mathcal{L}_r = \emptyset$. CHARM-L groups all dsets with prefix P , in an equivalence class, denoted $[P]$. CHARM-L starts by initializing the prefix class $[\emptyset]$ with the individual descriptors (line 1). It then makes a call to the extension subroutine, passing it the parent equivalence class and the lattice root as the current lattice node.

CHARM-L-EXTEND takes as input the current lattice node \mathcal{L}_c (initially the root node), and an equivalence class $[P]$. For each dset $X_i \in [P]$ (line 4), we combine it with other dsets $X_j > X_i$ in $[P]$ (line 7) to form a longer dset. The routine CHARM-L-PROPERTY tests inserts the newly created dset $X = X_i \cup X_j$ in the new class $[X_i]$. It also tests if two closed set properties are satisfied: 1) if $g(X_i) \subseteq g(X_j)$ we can replace the dset X_i with the larger dset $X_i \cup X_j$, since whenever X_i describes a geneset, it also involves the descriptors in X_j , and 2) if $g(X_i) \supset g(X_j)$ then we replace X_j with $X_j \cup X_i$ for the same reason (see [12] for more details). These properties allow CHARM-L to efficiently prune the search tree. The routine SUBSUMPTION-CHECK-

LATTICE-GEN checks if the new prefix X_i is a closed set and if so inserts it into the closed dset lattice.

Whenever CHARM-L generates a new closed dset it assigns it a unique closed dset identifier, called *cid*, and it maintains for each element $X_i \in [P]$ its corresponding *cid-set*, denoted $\mathbb{C}(X_i)$, which is the set of all cids of already mined closed dsets that are supersets of X_i . Given $\mathbb{C}(X_i)$ and $\mathbb{C}(X_j)$, one can obtain the set of closed dsets that contain $X = X_i \cup X_j$ by simply intersecting the two cidsets, i.e., $\mathbb{C}(X) = \mathbb{C}(X_i) \cap \mathbb{C}(X_j)$ (line 8). SUBSUMPTION-CHECK-LATTICE-GEN enumerates all closed sets which are not subsumed (i.e., do not have the same geneset as some superset), but in addition, it also generates a new lattice node \mathcal{L}_n for the new closed set X_i , and inserts it in the appropriate place in the closed dset lattice \mathcal{L} . This new lattice node \mathcal{L}_n becomes the current node in the next recursive call of the extension subroutine (line 11). Since the list of closed supersets of X_i may change whenever a new closed dset is added to the lattice, a check is made in line 6 to update $\mathbb{C}(X_i)$ for each remaining element in the class.

```

SUBSUMPTION-CHECK-LATTICE-GEN( $\mathcal{L}_c, X, \mathbb{C}(X)$ ):
1.   $\mathcal{P} = \{Z \in \mathcal{C} \mid Z.cid \in \mathbb{C}(X)\}$ 
    //eliminate subsumed dsets
2.  for each  $Z \in \mathcal{P}$  do
3.    if  $|g(X)| = |g(Z)|$  then return  $\mathcal{L}_c$ 
    //Insert  $X$  as parent of  $\mathcal{L}_c$ 
4.     $\mathcal{L}_n = X$ 
5.     $\mathcal{L}_c.parents.add(\mathcal{L}_n), \mathcal{L}_n.children.add(\mathcal{L}_c)$ 
    //Adjust Lattice
6.     $\mathcal{P}^{min} = \{Z \in \mathcal{P} \mid Z \text{ is Minimal}\}$ 
7.    for all  $Z \in \mathcal{P}^{min}$  do
8.       $\mathcal{L}_n.parents.add(Z), Z.children.add(\mathcal{L}_n)$ 
9.      for all  $Z_c \in Z.children$  do
10.       if  $Z_c \supset \mathcal{L}_n$  then
11.          $Z_c.parents.remove(Z),$ 
            $Z_c.children.remove(Z_c)$ 
12.   return  $\mathcal{L}_n$ 
    
```

Figure 6: Subsumption Checking & Lattice Growth.

Subsumption Check and Lattice Generation: To check if a dset X_i is closed (Figure 5, line 10), we apply SUBSUMPTION-CHECK-LATTICE-GEN shown in Figure 6. This routine takes as input the current lattice node \mathcal{L}_c , the new dset X , and the cidset $\mathbb{C}(X)$. The first task is to check if X is subsumed. For this we consider all closed dsets \mathcal{P} that are supersets of X (line 1). If X has the same geneset ($|g(X)|$) cardinality as any of its supersets $Z \in \mathcal{P}$ (lines 2), then X is subsumed (this is true, since for $X \subset Z$, $g(X) = g(Z) \iff |g(X)| = |g(Z)|$) and we return (line 3). Otherwise, the new lattice node is initialized as $\mathcal{L}_n = X$ (line 4). Each node in the lattice maintains a list of parents (immediate supersets) and children (immediate subsets). We add the new node \mathcal{L}_n as a parent of the current node \mathcal{L}_c , and \mathcal{L}_c as child of \mathcal{L}_n (line 5). Out of all the closed supersets of $\mathcal{L}_n = X$, the minimal supersets are found \mathcal{P}^{min} (line 6). Each minimal superset $Z \in \mathcal{P}^{min}$ becomes a parent of \mathcal{L}_n (and \mathcal{L}_n a child of Z) (line 8). Finally, for every child Z_c of Z , if $Z_c \subset \mathcal{L}_n$ then its parent pointers have to be adjusted; we remove Z from Z_c 's parents (and Z_c from Z 's children) (lines 9-10). Finally, we return the new lattice node \mathcal{L}_n (line 12).

Updating \mathbb{C} : Consider the UPDATE- \mathbb{C} routine in CHARM-L (Figure 5, line 6). After the recursive call to CHARM-L-EXTEND (Figure 5, line 11), new closed sets may have been generated, so we need to update the cidsets for all remaining elements in class $[P]$. That is for all dsets $X_j \in [P]$, with $X_j \geq X_i$, UPDATE- \mathbb{C} adds the cids of all newly generated closed sets to $\mathbb{C}(X_j)$.

3.2 Finding Minimal Generators

Once the set of all closed dsets, \mathcal{C} , for a given dataset has been found using CHARM-L, the next step is to generate the set of minimal generators, $\mathcal{M}(X)$, for each dset $X \in \mathcal{C}$. Note that a minimal generator Z of a closed dset X is a minimal dset that is a subset of X , but not a subset of any of X 's immediate closed subsets in the closed dset lattice \mathcal{L} . Let $\mathcal{Y} = \{Y_1, Y_2, \dots, Y_k\}$ be the set of immediate closed subsets of X in \mathcal{L} , and let $\mathcal{M}(Y_i)$ be the set of minimal generators of dset Y_i . Further define the dset $\Delta_i = X - Y_i$ to be those elements in X that are not in Y_i , and let $\Delta = \{\Delta_1, \Delta_2, \dots, \Delta_k\}$. A dset Z is called a *hitting set* of Δ iff $Z \cap \Delta_i \neq \emptyset$ for all $i \in [1, k]$. A dset Z is called a *minimal hitting set* if there does not exist another hitting set Z' , such that $Z' \subset Z$.

THEOREM 3.1. *Given dset X , the set of minimal generators of X , namely $\mathcal{M}(X)$ is the same as the set of minimal hitting sets of Δ .*

A similar theorem was independently reported in [6].

```

// $X$  is a closed dset,
// $\mathcal{Y}$ , the set of adjacent closed subsets of  $X$  in  $\mathcal{L}$ 
MinimalGenerators( $X, \mathcal{Y}$ ):
1.   $H(X) = \emptyset;$ 
2.   $\Delta = \{\Delta_i = X - Y_i \mid Y_i \in \mathcal{Y}\};$ 
3.  for each  $k$ -tuple  $(z_1, z_2, \dots, z_k)$ , with  $z_i \in \Delta_i$ 
4.     $Z = \{z_1, z_2, \dots, z_k\};$  //removes duplicate  $z_i$ 's
5.     $H(X) = H(X) \cup \{Z\};$ 
6.   $\mathcal{M}(X) = \{Z \in H(X) \mid Z \text{ is minimal in } H(X)\};$ 
    
```

Figure 7: Finding Minimal Generators.

Our novel algorithm to find minimal generators is shown in Figure 7, and is based on the above theorem. Given closed set X and the set \mathcal{Y} of its immediate closed subsets (say $|\mathcal{Y}| = k$), we first determine the set of differences Δ (line 2). Next we construct each possible k -tuple of the form (z_1, z_2, \dots, z_k) by picking exactly one descriptor from each difference, i.e., $z_i \in \Delta_i$ (line 3). Let Z be the set constructed from this k -tuple, i.e., $Z = \{z_1, z_2, \dots, z_k\}$ (line 4). Whereas the k -tuple may have duplicate items, Z will, by set definition, remove any duplicate elements. By construction, it is clear that Z is a hitting set for Δ . All such hitting sets are added to a set $M(X)$ (line 5), and finally only the minimal dsets in $M(X)$ are added to $\mathcal{M}(X)$ (line 6), which thus contains all minimal hitting sets of Δ , which are also the minimal generators of X .

For example, consider the dset $X = d_1d_2d_5d_6$. As shown in Figure 4, it has only one immediate closed subset $Y_1 = \{d_2, d_5\}$. Thus $\Delta_1 = d_1d_6$. In line 6, each element in Δ_1 will be picked in turn and added to $H(X)$, and since a single element is minimal, we have $\mathcal{M}(X) = \{d_1, d_6\}$, as shown in Figure 4. For the dset $X = d_1d_2d_3d_4d_5d_6d_7$, with immediate subsets $\mathcal{Y} = \{d_2d_3d_4, d_1d_2d_3d_4d_5d_6d_7, d_1d_2d_4d_5d_6\}$, we have $\Delta = \{d_1d_5d_6d_7, d_4, d_3d_7\}$. Picking all 3-tuples with

one element from each Δ_1 we get the hitting sets $H(X) = \{d_1d_3d_4, d_1d_4d_7, d_3d_4d_5, d_4d_5d_7, d_3d_4d_6, d_4d_6d_7, d_3d_4d_7, d_4d_7\}$. The minimal sets in $H(X)$ are given as $\mathcal{M}(X) = \{d_1d_3d_4, d_3d_4d_5, d_3d_4d_6, d_4d_7\}$ which are the minimal generators for X . Figure 4 shows the minimal generators for all closed dssets.

3.3 Non-redundant Redescription Generation

```

GenerateRedescriptions ( $X \in \mathcal{C}$ ):
  for all pairs  $Y, Z \in \mathcal{M}(X)$ 
     $Q = Y \cap Z$ ;
    output :  $g(X) : Y - Q \iff Z - Q|Q$ 
    
```

Figure 8: Non-redundant Redescription Generation.

Let \mathcal{C} be the set of all closed dssets. Given any closed dssets $X \in \mathcal{C}$ and its corresponding geneset $G = g(X)$, as well as the set of minimal generators of X , given as $\mathcal{M}(X) = \{Y|Y \text{ is a minimal generator of } X\}$, Figure 8 shows the algorithm for generating minimal/non-redundant (conditional) redescrptions. For each distinct pair $Y, Z \in \mathcal{M}(X)$, we generate the rule $G : Y - Q \iff Z - Q|Q$, where $Q = Y \cap Z$ is the condition dsset. Since we use minimal generators to produce the rule, we can guarantee that the rule is minimal.

For example, consider the closed dsset $X = d_1d_2d_5d_6$ with geneset $G = g(X) = g_1g_3g_4g_5$ and minimal generators $\mathcal{M} = \{d_1, d_6\}$, as shown in Figure 4. The only possible (unconditional) redescription is $G : d_1 \iff d_6$, as shown in Table 2. For the closed dsset $X = d_1d_2d_4d_5d_6$, with geneset $G = g(X) = g_1g_3g_5$, and minimal generator set $\mathcal{M} = \{d_1d_4, d_4d_5, d_4d_6\}$, we obtain the following conditional redescrptions: $G : d_1 \iff d_5|d_4, d_5 \iff d_6|d_4, d_1 \iff d_6|d_4$. All other redescrptions shown in Table 2 are obtained in a similar manner. Note that an unconditional redescription is found for any disjoint pair of minimal generators, whereas a conditional redescription is found for any non-disjoint pair of minimal generators.

Observe that (exact) redescrptions always occur in pairs. When $X \iff Y$ holds, so does $\neg X \iff \neg Y$. Due to our support threshold *minsup*, however, we may not mine both forms explicitly. Even if both forms conform to the support threshold, recall that we mine only those redescrptions where both sides obey the conjunctions bias; so if X or Y is a conjunction of more than one descriptor, its negation would be a disjunction, and hence outside the purview of our bias.

3.4 Constraint-Based Mining

As described above, our methods are oblivious of the meaning of a descriptor, i.e., the mining process is not aware which of them are ‘positive’ and which are negated. In general a large number of negated descriptors leads to a combinatorial blowup in the number of closed dssets. To make mining tractable we introduce several constraints. The first constraint enforces the presence of certain descriptors in the mined redescrptions. The input is in the form of a set of constraints: $\{C_1, C_2, \dots, C_k | C_i \subset \mathcal{D}, i = 1, \dots, k\}$. Each constraint C_i specifies those descriptors that must *all* be present in a redescription. Each redescription output must satisfy *at least one* constraint C_i . Thus the set of constraints are treated as a disjunction over conjunctions of descriptors. We efficiently check each constraint during mining. In the CHARM-L pseudo-code shown in Figure 5, we do the

Table 3: Datasets used in this paper.

DB	NumGenes	NumDesc	AvgSize
G1	74	824	88
G2	332	1700	81
G3	168	1189	52

following test on each prefix class $[P]$ (before line 4). Let $U = \bigcup_i X_i \in [P]$, then it suffices to check if U is a superset of at least one constraint C_j . If not, U cannot possibly satisfy any constraint and thus we can discard the entire search tree under $[P]$. This leads to very effective pruning. Another check of the constraints is made before adding a new closed set to the lattice.

We also introduce a constraint on the genesets. As in the case of dssets, the input is in the form of sets over \mathcal{G} and the meaning is similar. The check of these gene constraints are also done effectively while mining as follows: whenever we obtain a new geneset $g(X)$ in Fig. 5 (line 4), we keep track of the genes not in the intersection. If at least one constraint C_i remains unaffected, the new dsset is kept for the next step, or else it is pruned.

We also implemented the ability to impose a length constraint on the mined closed dssets. To implement this, it is not correct to simply stop extending a dsset if its length exceeds some threshold, since this would most likely produce a non-closed set, leading to wrong minimal generators. The correct way is to impose the length constraint on the minimal generators! Note that every minimal generator will be visited by the CHARM-L algorithm while computing closed sets. So while we never produce a minimal generator with length exceeding the constraint, a closed dssets might be longer. This ensures that the set of dssets with the length constraint are a subset of those without the constraint.

Finally, we have developed an interactive language, using MATLAB style scripting capabilities, to support the exploration of genesets through redescription analysis. The language contains primitives to define subsets of genes and descriptors, to request that a redescription be attempted for a selected set of genes (or involving those induced by a selected set of descriptors), to impose constraints on the mining process, manage the resulting mined redescrptions, and to investigate how the space of possible answers changes with varying inputs. The next section contains running examples of how a bioinformatician would use this language.

4. EXPERIMENTAL RESULTS

We now present an application of redescription mining to studying gene expression datasets from microarray experiments conducted on the budding yeast *Saccharomyces cerevisiae*. We utilize the three datasets from [7] (see Table 3) to study the scalability and performance of our implementations. The biological results are explained and detailed with specific reference to one of them, namely G1.

The specific details of the datasets can be had from [7] but we briefly review their characteristics here. All datasets define a small set of yeast genes (NumGenes in Table 3) but relatively greater number of descriptors (NumDesc). The average number of descriptors per dataset is also high (Avg-Size). The descriptors are drawn a variety of sources. Some denote expression levels in specific microarray measurements taken from Gasch et al. [4] and Wyrick et al. [10]. For instance, ‘genes negatively expressed one-fold or below in

the 15 minute time point of the 1M sorbitol experiment' is one such descriptor. A second class of descriptors asserts membership of genes in targeted taxonomic categories of the Gene Ontology (biological processes (GO BIO), cellular components (GO CEL) or molecular functions (GO MOL)). A final class of descriptors is based on clustering time course datasets using a k-means clustering algorithm [8] and using the clusters as descriptors. All descriptors are given an identifier as well as mnemonic for ease of interpretation.

4.1 Biological Results

We now present two interactive scenarios of how a biologist will use our algorithms, with the G1 dataset, as well as one example of a cluster of genes dense in conceptual descriptor space. For the results presented in this section, we negated each of the 824 descriptors in the G1 dataset and added it back to the descriptor pool, so that the algorithm can be used to mine set differences in addition to intersections. This yields a 74×1648 input boolean matrix. Recall that, by construction, such a dataset will be exactly 50% sparse.

```

> load yeast_descriptors;
> descriptors([YOR374W]);
ans = [d127, d183, d184, ... ];

> explain(d184);
ans = (GASCH_ENV_05004) Heat Shock 20 mins hs-1 >= 5

> genes([d184]);
ans = [g4, g10, g12, ...];

> find(descriptors,'Heat Shock*15*');
ans = [d146, d181, d183, ...];

> explain(d183);
ans = (GASCH_ENV_05003) Heat Shock 15 mins hs-1 >= 5
...
> Jaccards("d183","d184");
ans = 0.857;

> dsubset = find(descriptors,'GO*MOL*');
> dsubset = dsubset + [d183 d184];
> gsubset = genes(dsubset);
> constraints = mustinclude([d183 d184]);
> minsup = 15; jac = 1;
> redescribe(gsubset,dsubset,constraints,minsup,jac);
no redescrptions found.
...
> dsubset = [];
> dsubset = find(descriptors,'GO*BIO* | GO*CEL* | GO*MOL*');
> dsubset = dsubset + [d183 d184];
> gsubset = genes(dsubset);
> redescribe(gsubset,dsubset,constraints,minsup,jac);
1 redescrptions found.
ans = "d183 d1212 d1284 d1339" is redescrbed as "d184 d1133"

> prettyprint(ans);
ans = (GASCH_ENV_05003) Heat Shock 15 mins hs-1 >= 5
EXCEPT (GO_MOL_1578) mannose transporter
EXCEPT (GO_CEL_30312) external protective structure
EXCEPT (GO_BIO_06000) fructose metabolism
IS REDESCRIBED AS
(GASCH_ENV_05004) Heat Shock 20 mins hs-1 >= 5
EXCEPT (GO_MOL_05554) molecular function unknown;
...

```

Figure 9: Interactive exploration of genesets using redescription analysis.

4.1.1 Interactive Scenario 1

The first scenario is depicted in Fig. 9 and illustrates a biologist who explores descriptors around his favorite gene

— YOR374W, an ORF in *S. cerevisiae* that encodes for an aldehyde dehydrogenase (an enzyme - E.C.1.2.1.5 - that catalyzes the reaction from {aldehyde, NAD+, H₂O} to {acid, NADH}), and which he knows to be very highly expressed in time point 20 minutes of the Gasch heat shock condition (more than five fold). Aldehyde dehydrogenase is important enzymatically because the system must obtain increased energy from acetaldehyde under strenuous growth conditions. The biologist begins the analysis by identifying the descriptors that YOR374W participates in. One of them is descriptor d184 that denotes *all* ORFs that are expressed more than five fold in the above time point; it contains 19 genes. Looking at the nearby time point (15 minutes) the biologist notices that the corresponding descriptor (d183) contains 21 genes, with 18 in common with d184. The Jaccard's coefficient between these descriptors is already high (0.857) but the biologist is curious to determine if there could be an exact redescription by using the GO vocabularies. This might be significant if physiologically some response/repair could be associated with the transition between the time points. He adds the GO molecular function taxonomy into the session but the Jaccard's coefficient doesn't improve. Then, he removes the GO molecular function taxonomy and adds the cellular component taxonomy and the biological process taxonomy, in turn. None of the three choices improves the quality of the redescription. Finally, he adds all three taxonomies simultaneously, and the system presents him with a perfect redescription:

$$d183 - d388 - d460 - d515 \Leftrightarrow d184 - d309$$

(In Fig. 9 note that the set subtractions represented as conjunctions of negated descriptors; in this convention, subtraction of descriptor *d388* is captured as conjunction of descriptor *d1212*, whose index is 824 plus the original *d388*): In other words, to make d183 equivalent to d184, we need to subtract descriptors d388, d460, and d515 on the left (to remove 3 genes) and subtract descriptor d309 on the right (to remove 1 gene), bringing the commonality to 18, as desired. As Fig. 9 explains, d388 refers to the GO molecular function category: mannose transporter, d460 refers to the GO cellular component category: external protective structure, and d515 refers to the GO biological process category: fructose metabolism. d309, on the right side, incidentally happens to refer to genes whose molecular function, according to GO, is unknown. The implied message, from the above redescription, is that as we go from time point 15 minutes to time point 20 minutes genes belonging to the above three categories drop out of the highly expressed (≥ 5 fold) category.

4.1.2 Interactive Scenario 2

The above example showed how we can relate two different time points but the task is relatively easy given the high degree of initial overlap between the sets. Let us add more complexity to the mix and, this time, relate not only a different time point but also a less stringent threshold. One such descriptor is d141 which is the set of genes expressed more than (just) two fold in the 10 minutes time point (not 20 minutes, as in descriptor d184). Since d141 contains 50 genes, it cannot enjoy a Jaccard's coefficient of more than 0.38 with descriptor d184. This time, the system is configured to use all available descriptors. It arrives at 34 redescrptions involving d141 as well as d184, but all of

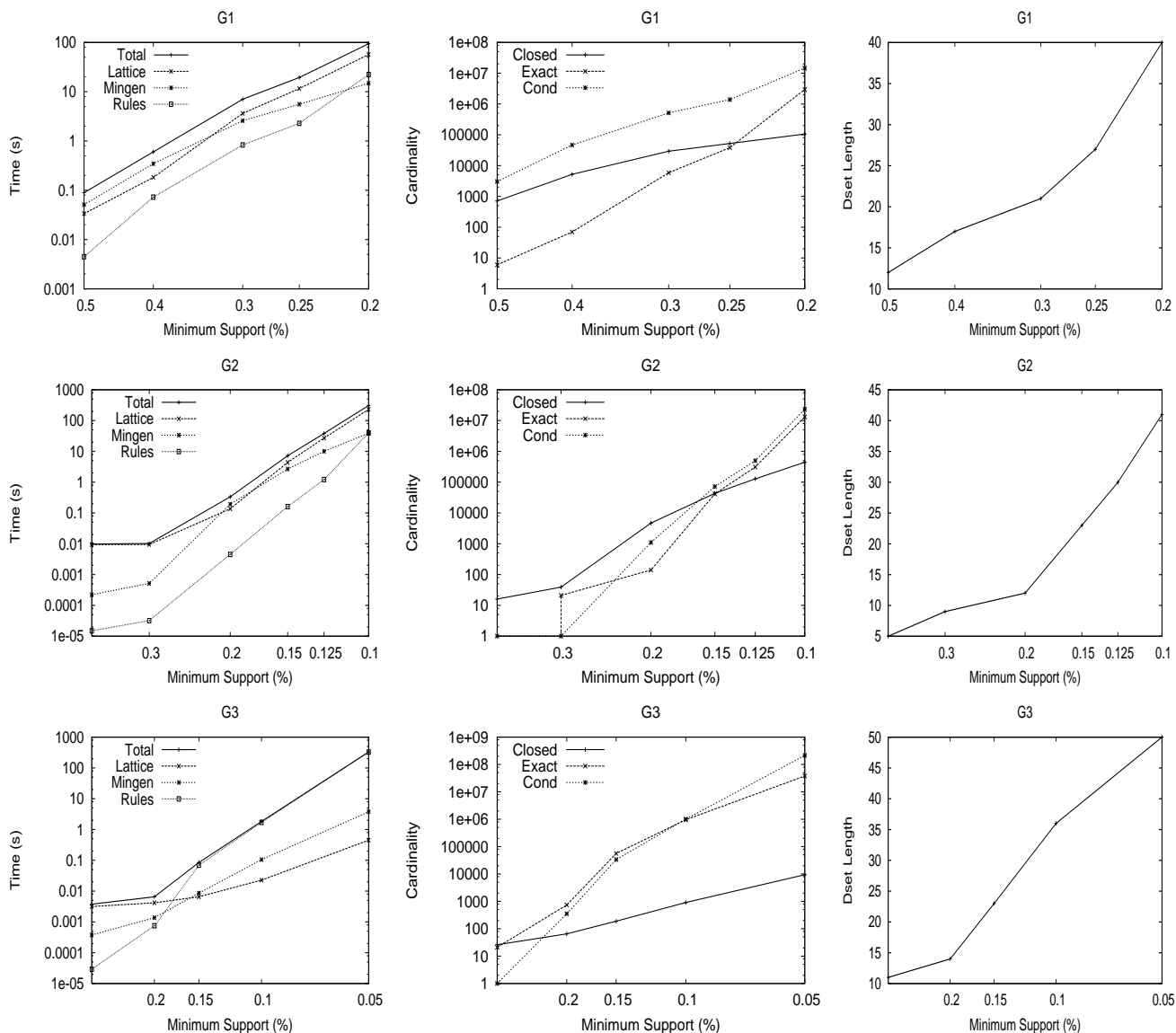


Figure 10: Mining time, rule cardinality, and length of longest closed dset for three datasets.

them are expressed conditionally; example:

$$d141 - d515 - d608 \Leftrightarrow d184|d183$$

Recall that d515 refers to the fructose metabolism category, but it cannot possibly help in improving the Jaccard’s coefficient as it removes only one gene from d141. Descriptor d608, on the other hand, refers to the set of genes expressed four-fold or above in a different experiment (histone depletion). In other words, to go from genes expressed 2-fold or above to genes expressed 5-fold or above (across time points), we have to remove the genes associated with fructose metabolism and that have relatively high positive expression in histone depletion. The conditional descriptor d183 can be viewed as a further conjunct on both sides, but since it already has significant overlap with d183, has only an incremental ‘trimming’ role to offer (e.g., the removal of few genes from both sides). The implied message from this redescription is that the genes that are only moderately expressed (between 2- and 5-fold) in the heat shock time

points are those that are highly expressed in the histone depletion time point. Further inspection of the 34 redescrptions mined above reveals that all of them contain the only other aldehyde dehydrogenase represented in the dataset - YMR170C (NAD(P)+). This brings out the ability of redescription mining to bring concerted genes together.

4.1.3 Identifying Dense Conceptual Clusters

If we view closed itemsets as biclusters then it is of interest to determine dense conceptual clusters underlying a given set of constraints. We restricted our attention to a set of genes clustered together in a k-means descriptor (d77; HS2_KMC_01) for the Heat Shock experiment. When queried for all redescrptions (at $\mathcal{J} = 1$ and a minimum support threshold of 13) that involve this descriptor, we obtained 867 redescrptions! Recall that the total number of input descriptors is less than double of this number. On closer inspection, we noticed that all these redescrptions comprised only 98 descriptors and all were conditional re-

descriptions, with an almost majority having $d77$ as the conditional. Hence, this means that, coupled with $d77$, many of these descriptors forms a minimum generator of the closure of $d77$. Sometimes, two or more of these descriptors need to be conjoined in order to form the minimum generator. Thus these descriptors, together with the genes they cover (14 of them, out of the total 74) form a dense bicluster in the dataset.

4.2 Performance Results

We ran some more experiments to test the performance of our approach. Here we only consider ‘positive’ descriptors, since the whole set of descriptors (including negative ones) can only be mined using constraints. For example, if all descriptors are used, then the average size would be the same as NumDesc, for each dataset. Our experiments were run on a 3.2Ghz Pentium4 machine with 2GB of memory running Linux, with a 7200rpms 200GB IDE disk.

Figure 10 shows the running time for the various steps in mining redescrptions, as well as the rule cardinalities for the different datasets as a function of minimum support threshold. The left column shows the time, where the legends ‘Total’ means the total execution time for the entire algorithm, ‘Lattice’ means the time it takes to build the frequent closed dset lattice, ‘Mingen’ means the time it takes to generate the minimal generators for each closed dset, and ‘Rules’ means the time it takes to extract the self and conditional redescrptions. The middle column shows the cardinalities, where the legends ‘Closed’ means the number of frequent closed dsets mined, ‘Exact’ means the number of exact redescrptions and ‘Cond’ means the number of conditional redescrptions mined. The right column shows the longest closed dset found at a given support threshold.

We can observe that for the G1 and G2 datasets most of the time is spent in mining the closed dsets and constructing the lattice, whereas the minimal generator time is lower and rule generation time is even lower. However, as we decrease minimum support, more closed dsets are found and there is an increase in the number of exact redescrptions. At the same time there is an even bigger explosion in the number of conditional rules. This leads to an increase in the running time for rule generation. Notice also that for G1 and G2 the longest closed set has size 40. For G3, the effect is even more pronounced. We find that the number of exact and conditional rules increases dramatically, and thus the rule generation time dominates. The times for closed dset mining, lattice generation, and minimal generators is negligible. The longest dset mined for G3 has length 50!

5. DISCUSSION

We have demonstrated a formal approach to redescription mining, along with examples of how a biologist would use such a facility interactively. As biologists are empowered to create their own vocabularies and descriptors and reason with them, there will be greater understanding of large scale bioinformatics datasets.

In future work, we plan to increase the expressiveness of our formulation in many ways. First, we would like to redescrbe not just in a propositional logic, as described here, but employing a form of predicate logic. An example from bioinformatics would be the use of a homology relation to relate, for instance, descriptors from a yeast vocabulary to descriptors in a vocabulary designed for mouse genes. This

is a natural generalization of the type of redescrptions considered here. Second, we would like to create chains of (approximate) redescrptions, effectively forming a *story* from one gene set to another. Given two disjoint sets, for instance, even though there could be no redescription connecting them, there could be a *chain* of approximate redescrptions going from one to the other. Finally, we would like to use redescrptions as a basis for knowledge management in domains rich in descriptors. Towards this goal, we aim to extend the expressiveness of our mining algorithms towards other classes of expressions such as disjunctions, and also more generality, e.g., CNF or DNF.

Acknowledgments

Zaki’s work was supported in part by NSF CAREER Award IIS-0092978, DOE Career Award DE-FG02-02ER25538, NSF grant EIA-0103708, and NSF grant EMT-0432098. Ramakrishnan’s work was supported in part by NSF grants IBN-0219332 and EIA-0103660. We acknowledge the help of Deept Kumar, who furnished us with the datasets from [7] and Laxmi Parida, for useful discussions.

6. REFERENCES

- [1] Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *1st International Conference on Computational Logic*, July 2000.
- [2] D. Burdick, M. Calimlim, and J. Gehrke. MAFIA: a maximal frequent itemset algorithm for transactional databases. In *IEEE Intl. Conf. on Data Engineering*, pages pp. 443–452, April 2001.
- [3] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1999.
- [4] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Storz, D. Botstein, and P.O. Brown. Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes. *Mol. Biol. Cell*, Vol. 11:pages 4241–4257, 2000.
- [5] L. Nourine and O. Raynaud. A fast algorithm for building lattices. *Information Processing Letters*, 71:199–204, 1999.
- [6] J.L. Pfaltz and R.E. Jamison. Closure systems and their structure. *Information Sciences*, 139:275–286, 2001.
- [7] N. Ramakrishnan, D. Kumar, B. Mishra, M. Potts, and R.F. Helm. Turning CARTwheels: An Alternating Algorithm for Mining Redescrptions. In *Proc. KDD’04*, pages 266–275, Aug 2004.
- [8] A. Sturn, J. Quackenbush, and Z. Trajanoski. Genesis: Cluster Analysis of Microarray Data. *Bioinformatics*, Vol. 18(1):pages 207–208, 2002.
- [9] J. Wang, J. Han, and J. Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining*, August 2003.
- [10] J.J. Wyrick, F.C. Holstege, E.G. Jennings, H.C. Causton, D. Shore, M. Grunstein, E.S. Lander, and R.A. Young. Chromosomal Landscape of Nucleosome-Dependent Gene Expression and Silencing in Yeast. *Nature*, Vol. 402:pages 418–421, 1999.
- [11] M. J. Zaki. Generating non-redundant association rules. In *6th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining*, pages pp. 34–43, August 2000.
- [12] M. J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *2nd SIAM International Conference on Data Mining*, pages pp. 457–473, April 2002.
- [13] M. J. Zaki and C.-J. Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):462–478, April 2005.