J Intell Inf Syst
DOI 10.1007/s10844-017-0448-5

Redescription mining augmented with random forest of multi-target predictive clustering trees

Matej Mihelčić[1,3] · Sašo Džeroski[2,3] · Nada Lavrač[2,3] · Tomislav Šmuc[1]

Received: 30 June 2016 / Revised: 8 November 2016 / Accepted: 13 January 2017

Abstract In this work, we present a redescription mining algorithm that uses Random Forest of Predictive Clustering Trees (RFPCTs) for generating and iteratively improving a set of redescriptions. The approach uses information about element membership in different queries, generated from a single constructed PCT, to explore redescription space, while queries obtained from the Random Forest of PCTs increase candidate diversity. The approach is able to produce highly accurate, statistically significant redescriptions described by Boolean, nominal or numerical attributes. As opposed to current tree-based approaches that use multi-class or binary classification, we explore the benefits of using multi-label classification and multi-target regression to create redescriptions. Major benefit of the approach, compared to other state of the art solutions, is that it does not require specifying minimal threshold on redescription accuracy to obtain highly accurate, optimized set of redescriptions. The process of Random Forest based augmentation and different modes of redescription set creation are evaluated on three datasets with different properties. We

Electronic supplementary material The online version of this article (doi:10.1007/s10844-017-0448-5) contains supplementary material, which is available to authorized users.

✉ Matej Mihelčić
matej.mihelcic@irb.hr

Sašo Džeroski
saso.dzeroski@ijs.si

Nada Lavrač
nada.lavrac@ijs.si

Tomislav Šmuc
tomislav.smuc@irb.hr

[1] Ruđer Bošković Institute, Bijenička cesta 54, 10000 Zagreb, Croatia

[2] Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia

[3] Jožef Stefan International Postgraduate School, Jamova cesta 39, 1000 Ljubljana, Slovenia

Published online: 08 February 2017

use the same datasets to compare the performance of our algorithm to state of the art redescription mining approaches.

Keywords Knowledge discovery · Redescription mining · Random forest · Predictive clustering trees · World countries · Computer science bibliography · Bioclimatic niches

1 Introduction

Pattern mining (Agrawal et al. 1993; Giacometti et al. 2014; Han et al. 2007; Mooney and Roddick 2013) aims at discovering descriptive rules learned from data. Redescription mining (RM) (Ramakrishnan et al. 2004) shares this goal and is directed towards finding different descriptions of patterns by using one or more disjoint sets of descriptive attributes (these disjoint sets are also called views). The input to redescription mining algorithms consists of one or more tables containing all attributes from the given view and their corresponding values for all elements contained in the dataset. One input example obtained from the DBLP database (DBLP dataset 2010; Galbrun 2013), containing information about authors of scientific papers, can be seen in Table 1.

Redescription mining is an unsupervised, descriptive knowledge discovery task. Similarly as Association rule mining (Agrawal et al. 1993), it discovers associations between attributes. However, instead of finding one-directional implication relations it finds bi-directional equivalence relations. Association discovery in two-view data (van Leeuwen and Galbrun 2015) finds both uni and bi-directional associations but is aimed at explaining how these two views are related. Redescription mining is related to multi-view (Bickel and Scheffer 2004) and multilayer (Gamberger et al. 2014) clustering, though the main goal here is to find accurate redescriptions of interesting subsets of data, while clustering tends to find clusters that are not always easy to interpret. Finding similarities between different elements and connections between different descriptive attribute sets (views) ultimately leads to better understanding of the underlying data. The output of redescription mining is a set of redescriptions which are tuples of rules (logical formulas). The aim is to make these rules understandable and interpretable.

We present one redescription example $R_{ex} = (q_{1_{ex}}, q_{2_{ex}})$, where $q_{1_{ex}} = \neg\text{UAI} \wedge \neg\text{PKDD} \wedge \text{SEBD} \wedge \text{SIGMOD} \wedge \text{LPNMR}$ and $q_{2_{ex}} = \text{Thomas Eiter} \wedge \text{Gianluigi Greco}$. For all authors described by the redescription R_{ex} it must be valid that they co-authored a paper with Thomas Eiter and co-authored a (not necessarily the same) paper with Gianluigi Greco. They have also published at least one paper on conferences SEBD, SIGMOD and LPNMR but have not published any papers in conferences PKDD and UAI. Interestingly, Thomas Eiter and Gianluigi Greco have co-authored the paper *Boosting Information Integration: The INFOMIX System* published in SEBD in 2005. This makes the group of described authors very well connected in terms of co-authorship.

Applications of redescription mining Redescription mining is highly applicable in biology, economy, pharmacy, ecology and many other fields, where it is important to understand connections between different descriptors and to find regularities that are valid for different element subsets. For instance, it might be interesting to associate gene functions with gene locations in different genomes, to study similarities or differences in structure of different organisms, to relate proteins and different chemical compounds to understand the effects of

Table 1 Input example for the DBLP dataset

(a) View 1: Author-conference bipartite graph

Entity	ISAAC	FCT	...	PLILP
J.D.Tygar	false	false	...	false
.	.	.	...	.
.	.	.	...	.
.	.	.	...	.
JohnH.Reif	true	true	...	false
.	.	.	...	.
.	.	.	...	.
.	.	.	...	.
ChrisClifton	true	false	...	false
.	.	.	...	.
.	.	.	...	.
.	.	.	...	.
AthenaVakali	false	false	...	false

(b) View 2: Co-authorship network (self-authorship excluded)

Entity	J.D.Tygar	AdolfyHoisie	...	AthenaVakali
J.D.Tygar	false	false	...	false
.	.	.	...	.
.	.	.	...	.
.	.	.	...	.
JohnH.Reif	true	false	...	false
.	.	.	...	.
.	.	.	...	.
.	.	.	...	.
ChrisClifton	true	false	...	false
.	.	.	...	.
.	.	.	...	.
.	.	.	...	.
AthenaVakali	false	false	...	false

interactions, with potential application in design of new and more effective medicines, to associate animal species habitats with weather locations, in order to obtain knowledge about the effects of these conditions on habitats and co-habitats of different animal species, or to associate authors of scientific papers with different scientific conferences to obtain groups of related authors sharing some area of research. In these applications, implication relations, as provided with association rule mining, are not strong enough to allow explaining the underlying phenomena. Due to strong equivalence relations it produces, redescription mining is well suited for relating a set of attributes, which are generally understood well (such as questionnaires and various written or motorical tests devised by researchers) to a set of attributes, containing different measurements (medical, biological) which are not always understood well.

Related work in redescription mining The field of redescription mining was introduced in the work from Ramakrishnan et al. (2004) which presents a novel decision tree - based redescription mining algorithm called the CARTwheels. The algorithm builds two decision trees (one for each view) that are joined in the leaves. Redescriptions are found by examining the paths from the root node of the first tree to the root node of the second and the algorithm uses multi class classification to guide the search between the two views. Other approaches for redescription mining include: the approach proposed by Zaki and Ramakrishnan (2005) which uses a lattice of closed descriptor sets to find redescriptions, the approach proposed by Parida and Ramakrishnan (2004) for mining exact and approximate redescriptions based on relaxation lattice. Further, Gallo et al. (2008) present the greedy algorithm and the MID (Mining Interesting Descriptors) algorithm based on frequent itemset mining.

Galbrun and Miettinen (2012a) extend the greedy approach by Gallo et al. (2008) to work on numerical data, thus increasing capabilities of redescription mining algorithms. Galbrun and Kimming extend redescription mining to a relational (Galbrun and Kimmig 2014) setting, while Galbrun and Miettinen make extensions to allow interactive redescription mining (Galbrun and Miettinen 2012c). Recently, two novel tree-based algorithms were proposed by Zinchenko (2014). These approaches use decision trees in a non-Boolean setting and present different methods of layer by layer tree construction, which allows making informed splits based on nodes at each level of the tree.

Methodology In this work, we present a multi-target predictive clustering trees (PCTs) (Blockeel 1998; Kocev et al. 2013) based redescription mining algorithm developed to create a large number of diverse redescriptions. As in our previous work (Mihelčić et al. 2015a, 2015b), all created PCTs use multi-target classification or regression to find highly accurate, statistically significant redescriptions, which differentiates it from other tree based approaches, especially the CARTwheels approach. Using multi-target PCTs allows us to build one model to find multiple redescriptions using nodes at all levels of the tree and due to inductive transfer (Piccart 2012), multi-target trees can outperform single label classification or regression trees. Each node in one separately created PCT model, used to guide the search, represents a separate rule that is used in the construction of a PCT from the opposite view. Generated redescriptions are used to iteratively improve and expand a redescription set of user suggested, not necessarily fixed size (which alleviates the hard constraints on redescription set size used in Mihelčić et al. 2015b). The algorithm presented in Mihelčić et al. (2015b) has been extended to incorporate the random forest of PCTs as an augmenting model. This increases the accuracy and diversity of produced redescriptions. The approach relies on the fact that a great number of PCTs can be trained and converted to rules in parallel. Thus, this augmented process can be executed very efficiently and almost at the same running time, if executed in parallel threads, as the process containing only one PCT. Additional benefit of the approach is that it allows creation of highly optimized redescription sets without requiring users to constrain redescription accuracy. This is an advantage, compared to current state of the art approaches, because it usually needs to be determined through experimentation. Finally, rule minimization procedure, presented in our previous work (Mihelčić et al. 2015b), allows reducing the number of attributes that describe a given pattern without changing redescription accuracy or support. This allows obtaining shorter rules even when using trees of bigger depth size.

Structure After introducing the necessary notation (Section 2), we present the extended algorithm and perform the run-time complexity analysis of redescription mining process (Section 3). In Section 4, we evaluate algorithm extensions, compare its performance with several state of the art approaches on three datasets with different properties and present redescription examples obtained by these approaches. Finally, we conclude and outline directions for future work in Section 5.

2 Notation and definitions

Redescription mining in general considers redescriptions constructed on a set of views $\{W_1, W_2, \ldots, W_n\}$, $n \geq 1$, however we use only two views $\{W_1, W_2\}$ since all current redescription mining approaches use maximally two views. Using more than two views significantly increases computational complexity and requires data containing several disjoint sets of attributes describing the same set of elements. The corresponding attribute (variable) sets are denoted by V_1 and V_2. Each view contains the same set of $|E|$ elements and two different sets of attributes of size $|V_1|$ and $|V_2|$. Value $W_1(i, j)$ is the value of element e_i for the attribute a_j in view W_1. The data $D = (V_1, V_2, E, W_1, W_2)$ is a quintuple of the attribute sets, the element set, and the appropriate view mappings. A query (denoted q) is a logical formula F containing attributes from V_1 or V_2 as variables and the set of elements described by a query is called its support. A redescription $R = (q_1, q_2)$ is defined as a pair of queries, one for each view in the data and its support is the set of elements supported by both queries that constitute this redescription: $supp(R) = supp(q_1) \cap supp(q_2)$. We use $attr(R)$ to denote the multiset of attributes used in the redescription R and $attrs(R)$ to denote the corresponding set of attributes. The accuracy of a redescription $R = (q_1, q_2)$ is measured with the Jaccard index (Jaccard similarity coefficient):

$$J(R) = \frac{|supp(q_1) \cap supp(q_2))|}{|supp(q_1) \cup supp(q_2)|}$$

The Jaccard index is not the only measure used in the field because it is possible to obtain redescriptions with large support for which it is highly probable to have very good overlap of their queries. In this cases it is preferred to have redescriptions that reveal some more specific knowledge about the studied problem that is harder to obtain by random sampling from the underlying data distribution. This is why we compute the statistical significance (p-value) of each obtained redescription. The marginal probability of a query q_1, q_2 is denoted as $p_1 = \frac{|supp(q_1)|}{|E|}$ and $p_2 = \frac{|supp(q_2)|}{|E|}$ respectively. We define the set of elements in the intersection of the queries with $o = supp(q_1) \cap supp(q_2)$. The corresponding p-value (Galbrun 2013) is defined as

$$pV(q_1, q_2) = \sum_{n=|o|}^{|E|} \binom{|E|}{n} (p_1 \cdot p_2)^n \cdot (1 - p_1 \cdot p_2)^{|E|-n}$$

The p-value tells us if we can dismiss the null hypothesis that assumes that we obtained a given subset of elements by joining two random rules with marginal probabilities equal

to the fraction of covered elements. If the obtained p-value is lower than some predefined threshold, called the significance level, then this null hypothesis should be rejected. This estimate is optimistic when the assumption that all elements can be sampled with equal probability does not hold (which is often the case in practice).

We use two redescription quality measures based on properties of a redescription set that contains them. These measures, created with similar intuitions to those presented by Knobbe and Ho (2006), provide information about the level of redundancy of a given redescription with respect to described elements and attributes used in redescription queries in every other redescription contained in the given redescription set. The measure providing information about the redundancy of elements contained in the redescription support is called the average redescription element Jaccard index and is defined as:

$$AEJ(R_i) = \frac{1}{|\mathcal{R}| - 1} \cdot \sum_{j=1}^{|\mathcal{R}|} J(supp(R_i), supp(R_j)),\ i \neq j$$

Analogously, the measure providing information about the redundancy of attributes contained in redescription queries, called the average redescription attribute Jaccard index, is defined as:

$$AAJ(R_i) = \frac{1}{|\mathcal{R}| - 1} \cdot \sum_{j=1}^{|\mathcal{R}|} J(attrs(R_i), attrs(R_j)),\ i \neq j$$

To emphasize importance of the redescription size from the point of understandability ($|attr(R)| \geq 20$ considered to be highly complex to understand), we calculate the normalized redescription size as follows:

$$R_{size} = \begin{cases} \frac{|attr(R)|}{20} & , |attr(R)| < 20 \\ 1 & , 20 \leq |attr(R)| \end{cases}$$

3 The CLUS-RM algorithm

In this section, we describe the algorithm for mining redescriptions named CLUS-RM. The algorithm optimizes a redescription set of a size determined by redescription properties or suggested by a user. It uses multi-target predictive clustering trees (PCTs) (Blockeel 1998; Kocev et al. 2013) to create a cluster hierarchy that is used to explore the redescription space. In addition, a random forest of predictive clustering trees is used to diversify the search and to increase the overall redescription accuracy. We start by explaining the pseudo code of the algorithm (Algorithm 1) and then go into details of each procedure in the algorithm.

Algorithm 1 The CLUS-RM algorithm

Input: First view data (W_1), Second view data (W_2), Settings file
Output: A set of redescriptions $\mathcal{R}$

1: **procedure** CLUS-RM
2: $[\mathcal{D}_{W_1 init},\ \mathcal{D}_{W_2 init}] \leftarrow$ prepareTargetsForInitialPCT(W_1, W_2)
3: [PCTW_1, PCTW_2] $\leftarrow$ createSidesInitialPCT($\mathcal{D}_{W_1 init},\ \mathcal{D}_{W_2 init}$)
4: [RW_1, RW_2] $\leftarrow$ extractRules(PCTW_1, PCTW_2)
5: initializeArrays(elFreq, attrFreq, redScoreEl, redScoreAt, numEx, numAttr, numRetRed)
6: **while** RunInd¡maxIter **do**
7: [TmpRW_1, TmpRW_2] $\leftarrow$ emptyRuleSet()
8: [SRW_1, SRW_2] $\leftarrow$ emptyRuleSet()
9: $[\mathcal{D}_{W_1 Targ},\ \mathcal{D}_{W_2 Targ}] \leftarrow$ prepareTargets(RW_2, RW_1)
10: [PCTW_1, PCTW_2] $\leftarrow$ createPCT($\mathcal{D}_{W_1 Targ},\ \mathcal{D}_{W_2 Targ}$)
11: [RFPCTW_1, RFPCTW_2] $\leftarrow$ createSRFPCT($\mathcal{D}_{W_1 Targ},\ \mathcal{D}_{W_2 Targ}$)
12: TmpRW_1 $\leftarrow$ TmpRW_1 $\cup_*$ extractRules(PCTW_1)
13: TmpRW_2 $\leftarrow$ TmpRW_2 $\cup_*$ extractRules(PCTW_2)
14: SRW_1 $\leftarrow$ SRW_1 $\cup_*$ extractRules(RFPCTW_1)
15: SRW_2 $\leftarrow$ SRW_2 $\cup_*$ extractRules(RFPCTW_2)
16: RW_1 $\leftarrow$ RW_1 $\cup_*$ TmpRW_1
17: RW_2 $\leftarrow$ RW_2 $\cup_*$ TmpRW_2
18: $\mathcal{R} \leftarrow$ MineRed(RW_1, RW_2, SRW_1, SRW_2, expansionType, ConstSet, iteration, opSet, elFreq, attrFreq, redScoreEl, redScoreAt)
19: $\mathcal{R} \leftarrow minimizeReds(\mathcal{R})$
20: **return** $\mathcal{R}$

The algorithm starts by creating initial clusters for both views (line 2 and 3 in Algorithm 1) which is achieved by transforming a non-labeled dataset into a labeled dataset of positive, original elements (elements originally present in the dataset) and artificially generated, negative elements (elements not originally present in the dataset but artificially constructed and added to the dataset). For each element in the original view, we construct one negative, synthetic element (see Fig. 2) in such a way so that the original correlations among the attributes are broken. We achieve this by random shuffling of attribute values between the elements. The procedure allows experimentation with the number of shuffling steps and the number of attributes that are copied from the original elements to the artificial element. Complete randomization is achieved when the number of shuffling steps equals the number of attributes in the dataset and exactly one attribute value is copied to the artificial element at each step from a randomly chosen original element. The original elements are assigned a target label of 1.0, while the artificial elements are assigned a target label of 0.0 (see Table 2). Target label is used to give information to a supervised learning algorithm, such as PCT, which elements were originally present in the data and which were artificially constructed. The division between the original and the artificial elements (the idea previously used in the work from Gamberger et al. 2014), allows us to construct a cluster hierarchy, simultaneously creating descriptions of the original elements. The described procedure is one possible way to construct the initial clusters; other approaches include

Table 2 Creation of artificial elements for the random initialization procedure

Entity	W_1A_1	W_1A_2	W_1A_3	Target	Entity	W_2A_1	W_2A_2	W_2A_3	Target
(a) Original dataset for view 1					(b) Original dataset for view 2				
E_1	1.1	2.5	3.4		E_1	TRUE	FALSE	FALSE	
E_2	1.5	2.2	4.0		E_2	TRUE	TRUE	FALSE	
E_3	5.5	−0.6	−0.2		E_3	FALSE	FALSE	TRUE	
E_4	4.4	−0.2	2.		E_4	TRUE	TRUE	TRUE	
E_5	3.2	1.7	2.9		E_5	TRUE	FALSE	TRUE	
(c) Initial dataset for view 1					(d) Initial dataset for view 2				
E_1	1.1	2.5	3.4	1.0	E_1	TRUE	FALSE	FALSE	1.0
E_2	1.5	2.2	4.0	1.0	E_2	TRUE	TRUE	FALSE	1.0
E_3	5.5	−0.6	−0.2	1.0	E_3	FALSE	FALSE	TRUE	1.0
E_4	4.4	−0.2	2.0	1.0	E_4	TRUE	TRUE	TRUE	1.0
E_5	3.2	1.7	2.9	1.0	E_5	TRUE	FALSE	TRUE	1.0
E_1'	4.4	2.5	2.9	0.0	E_1'	TRUE	FALSE	TRUE	0.0
E_2'	3.2	−0.6	4.0	0.0	E_2'	FALSE	FALSE	TRUE	0.0
E_3'	3.2	−0.6	2.9	0.0	E_3'	TRUE	TRUE	TRUE	0.0
E_4'	4.4	−0.2	4.0	0.0	E_4'	FALSE	TRUE	FALSE	0.0
E_5'	5.5	1.7	2.9	0.0	E_5'	FALSE	FALSE	TRUE	0.0

For example, the artificial element E_1' in view 1 is created by copying a value for attribute W_1A_1 from original element E_4, for attribute W_1A_2 from E_1 and for attribute W_1A_3 from E_5. Since the element E_1' is artificially created, it is assigned a target value 0.0

assigning a random target attribute or using clusters computed by some other clustering algorithm. However, the initialization procedure used in our algorithm should preserve any strong (specific) connections and correlations that exist in the original data which are broken by using an approach that assigns random target labels.

After creating the initial dataset, we build predictive clustering trees on both views by performing regression on the target label and using other attributes as descriptive (line 3 in Algorithm 1). The decision to use regression trees instead of decision trees is purely technical, since it generates more rules because of the additional threshold associated with the target variable. These trees are converted to rules (line 4 in Algorithm 1) that describe element sets and are necessary for the next step of the algorithm. The rule lists $\mathrm{R}W_1$ and $\mathrm{R}W_2$ contain generated rules, and a new rule is added to the list if it differs from all other rules in a predefined number of attributes or if it describes a new unique element subset (the $\cup_*$ operator in Algorithm 1). The iterative process of the algorithm begins right after rule creation (line 6 in Algorithm 1). Here, we create targets based on the rules obtained in the previous step or in the initialization step (line 9 in Algorithm 1). The rules obtained by predictive clustering on W_1 are used to build targets for clustering on W_2 (denoted W_1T_1, W_1T_2), and vice versa. For each element in the dataset we assign label 1.0 if the element is described by some specific rule, otherwise 0.0 (see Table 3). For example, the attribute

Table 3 Intermediate generation of labels based on discovered rules

E	W_1A_1	W_1A_2	W_1A_3	W_2T_1	W_2T_2	E	W_2A_1	W_2A_2	W_2A_3	W_1T_1	W_1T_2
(a) Dataset for view 1						(b) Dataset for view 2					
E_1	1.1	2.5	3.4	1.0	0.0	E_1	TRUE	FALSE	FALSE	0.0	1.0
E_2	1.5	2.2	4.0	1.0	0.0	E_2	TRUE	TRUE	FALSE	0.0	1.0
E_3	5.5	−0.6	−0.2	0.0	0.0	E_3	FALSE	FALSE	TRUE	1.0	0.0
E_4	4.4	−0.2	2.0	1.0	0.0	E_4	TRUE	TRUE	TRUE	1.0	0.0
E_5	3.2	1.7	2.9	1.0	1.0	E_5	TRUE	FALSE	TRUE	1.0	1.0

W_2T_1 from dataset for view 1 represents the condition $If\ W_2A_1 = TRUE$ (constructed on dataset for view 2), which describes elements E_1, E_2, E_4, E_5. By placing this target attribute in the view 1 dataset, we guide the PCT construction (lines 9 and 10 in Algorithm 1) to create a cluster containing and describing the same set of elements with descriptive variables of view 1 (a choice that satisfies this condition is $If\ W_1A_3 > 0$).

Random forest of PCTs is constructed (line 11 in Algorithm 1) by using the same targets as to construct the PCT used to guide the search (line 10 in Algorithm 1). PCTs in the forest represent a set of weak learners trained on subspaces of attributes with the purpose of diversifying produced redescriptions and increasing their accuracy. The use of random forest has several advantages: 1) due to restricted size of the attribute subspace used to make a split, it is able to avoid local optima, 2) it explores much larger number of attribute associations (depending on the number of trees used in the forest) which is very important for produced redescriptions. The number of PCTs to be used in a random forest and the size of a random subspace are user defined parameters. The random subspace size is usually set to $\sqrt{N}$ or $log_2(N)$ in predictive tasks, where N equals the number of attributes contained in the selected view. However, in redescription mining it is important to discover different attribute interactions. Thus it is useful to have guarantees on attribute membership in different random subspaces. We have computed the necessary size of a random subspace, given a random forest of PCT with defined parameters, so that an arbitrary attribute occurs with a given probability in at least one split of every tree in the forest. The subset size is computed as: $k = N \cdot (1 - \sqrt[q]{1-p})$, where $q = (2^d - 1)$, d equals the average PCT depth and p denotes the desired probability to evaluate an arbitrary attribute in at least one split of every PCT in a random forest of given properties. Since, for very small number of attributes, this number quickly drops to 0, the subset size equals $k = max(\lceil N \cdot (1 - \sqrt[q]{1-p}) \rceil, \lceil log_2(N) \rceil)$. Assigning probability to attribute occurrence in at least one split of every tree in a forest allows influencing accuracy and diversity of queries used to produce redescriptions. High occurrence probability should be used on sparse datasets, when higher accuracy is required, while using lower probability on dense datasets increases diversity and brings computational advantage since smaller subsets need to be evaluated. Random forest of PCTs can be trained in parallel with minor loss in computation time, compared to the algorithm presented in our previous work (Mihelčić et al. 2015b). Rules obtained in the previous step are combined into redescriptions (line 18 in Algorithm 1) if they satisfy a given set of constraints $ConstSet$. It consists of minimal Jaccard index ($minJ$), maximum allowed p-value

($maxPval$), minimum and maximum support ($minSupp$, $maxSupp$) which have to be satisfied for a redescription to be considered as a candidate for the redescription set. The default value of 0.01 is used for p-value and a minimal support of 2 elements if corresponding parameters are not specified. Specifying the Jaccard index constraint is optional. After performing redescription creation and redescription set optimization, all queries produced by the random forest models are discarded. Finally, queries of the resulting redescriptions are minimized in line 19 of Algorithm 1 by using the query minimization procedure presented in our previous work (Mihelčić et al. 2015b).

3.1 The procedure for creating redescriptions

The algorithm for creating redescriptions from rules (Algorithm 2) joins view 1 rules (or their negation, if allowed by the user) with rules (or its negation) from view 2 (see Fig. 1 and line 2 in Algorithm 2). We distinguish three cases of creating redescriptions from rules (expansion types):

1. Unguided initial: $UInit \leftarrow ((SRW_1 \cup RW_1) \times^{opSet\setminus\{\vee\}}_{ConstSet} (SRW_2 \cup RW_2))$
2. Unguided: $U \leftarrow ((SRW_1 \cup RW_{1_{newRuleIt}}) \times^{opSet\setminus\{\vee\}}_{ConstSet} (SRW_2 \cup RW_{2_{newRuleIt}}))$
3. Guided: $G \leftarrow ((SRW_1 \cup RW_{1_{newRuleIt}}) \times^{opSet\setminus\{\vee\}}_{ConstSet} RW_{2_{oldRuleIt}}) \cup (RW_{1_{oldRuleIt}} \times^{opSet\setminus\{\vee\}}_{ConstSet} (SRW_2 \cup RW_{2_{newRuleIt}}))$

The $\times^{opSet}_{ConstSet}$ operator denotes a Cartesian product of two sets, allowing the use of logical operators from $opSet$ and leaving only those redescriptions that satisfy a given set of constraints $ConstSet$. The unguided expansion allows obtaining redescriptions with more diverse subsets of elements that can later be improved through the iteration process.

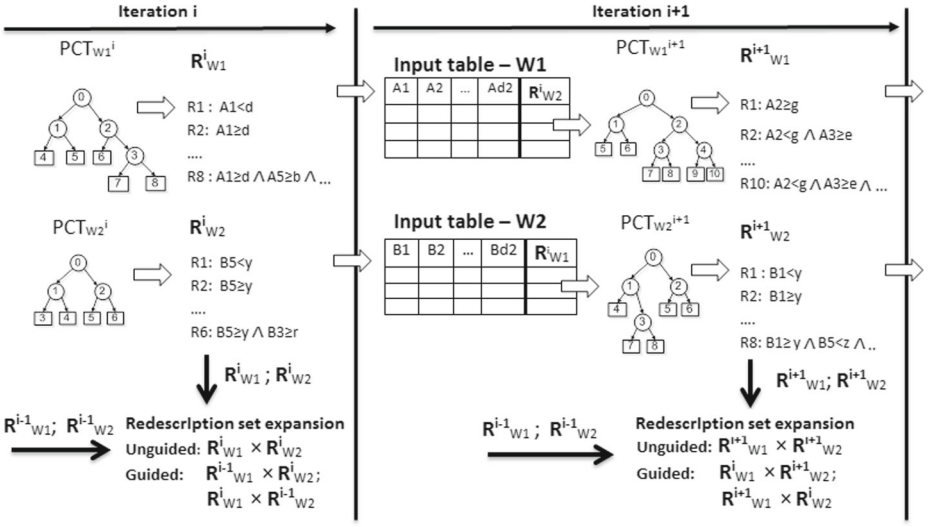

Fig. 1 Illustration of rule, redescription construction and iterations

The algorithm finds first $numRed$ redescriptions if the size is fixed by the user, or max(20,$numRed$) redescriptions if the size is suggested or fully automatically determined (line 4 in Algorithm 2). This minimal number of redescriptions is used to provide a set which is not very large but still provides different information about the elements and contains enough redescriptions to perform statistical analysis. After the minimal number of distinct redescriptions is found, the set is iteratively improved by exchanging the redescription with the worst comparative score with the newly created redescription (lines 3-21 in Algorithm 2). Five different arrays ($elFreq$, $attrFreq$, $redScoreEl$, $redScoreAt$, $redDstC$) are used to incrementally improve and add redescriptions to the redescription set. The element/attribute frequency arrays contain information about element/attribute occurrence in redescriptions from a redescription set. Redescription scores (line 9 in Algorithm 2) are computed as $redScoreEl(R) = \sum_{e \in supp(R)}(elFreq[e] - 1)$, $redScoreAt(R) = \sum_{a \in attr(R)}(attrFreq[a] - 1)$, $redDstC(R) = \sum_{e \in supp(R)} \delta_{0,elFreq[e]-1}$. The score of a new redescription (line 18 in Algorithm 2) is computed in the same way by using existing frequencies from the set. For a redescription R' such that $R_i = argmax_{R \in \mathcal{R}}\ score(R', R)$, where $score(R', R) = (\frac{(1.0-R'.elSc+1.0-R'.atrSc+R'.J+}{5}\ \frac{R'.eDC+R'.pVSc)}{5} - \frac{(1.0-R.elSc+1.0-R.attrSc+R.J+R.eDC+R.pVSc)}{5})$ and the $score(R', R_i) > 0$, all arrays are updated so that the frequencies of elements described by R_i and attributes contained in its queries are decreased by one, while the frequencies of elements and attributes associated with R' are increased (line 19 in Algorithm 2). The p-value score (R.pVSc) is computed as:

$$R.pVSc = \begin{cases} \frac{log_{10}(R.pval)}{-17.0} & \text{if } R.pval < 10^{-17} \\ 1.0 & \text{if } R.pval \geq 10^{-17} \end{cases}$$

We linearise and normalize redescription p-values to obtain a score that is used as one criteria in the optimization process with the aim of describing subsets of elements with queries that are unlikely to be created easily by matching a pair of randomly constructed queries. The $R.eDC$ denotes the element exclusive coverage and is defined as the fraction of elements that are described only by redescription $R(R.eDC = \frac{redDstC(R)}{|E|})$. $R.elSc$ and $R.atSc$ are obtained as a fraction of element frequencies for elements in redescription support or attributes used in its queries to total frequency of all elements or attributes. The score is defined to construct a redescription set by adding redescriptions that describe elements with low frequency by using non frequent attributes (to disallow redundancy) and, at the same time, finds as accurate and significant redescriptions as possible. Redescriptions describing unexplored elements are rewarded in the process (since we would like to describe as much elements as possible given the redescription set size). The score can be extended by defining importance weights for each criteria, providing users with the possibility to fine tune the redescription set optimization process.

Element weighting has been used before in subgroup discovery (Gamberger and Lavrač 2002; Lavrač et al. 2004) to model covering importance for elements. Our approach is similar but uses different weighting mechanism, adapts it to the redescription mining setting by combining element and attribute weights and incorporates it into the framework of iterative redescription set refinement in which some redescriptions can be replaced with more suitable candidates. The *exclusive coverage* has been used in the work from Knobbe and Ho (2006) as one criteria for extracting a set of patterns.

Algorithm 2 MineRed

Input: RW_1, RW_2, expansion type, ConstSet, iteration number, opSet, elFreq, attrFreq, redScoreEl, redScoreAt

Output: A set of redescriptions $\mathcal{R}$

1: **procedure** MINERED
2: expansionSet ← returnExpansionSet(expansionType, opSet, RW_1, RW_2, SRW_1, SRW_2)
3: **for** $R' \in expansionSet$ **do**
4: **if** (|$\mathcal{R}$|<ConstSet.MaxRed AND Const.SetSize==Fixed) OR (|$\mathcal{R}$|<max(20,ConstSet.MinRed) AND Const.SetSize!=Fixed) **then**
5: updateFrequencies(elFreq, attrFreq, R')
6: $\mathcal{R} \leftarrow \mathcal{R} \cup R'$
7: **if** (|$\mathcal{R}$|==ConstSet.MaxRed AND Const.SetSize==Fixed) OR (|$\mathcal{R}$| == max(20,ConstSet.MinRed) AND Const.SetSize!=Fixed) **then**
8: **for** $R \in \mathcal{R}$ **do**
9: computeScores(elFreq,attrFreq, redScoreEl, redScoreAt, redDstC, R)
10: **if** Const.SetSize!=Fixed **then**
11: Stats← computeStatistics($\mathcal{R}$)
12: **else if** (|$\mathcal{R}$|==ConstSet.MaxRed AND Const.SetSize==Fixed) OR (|$\mathcal{R}$| ≥max(20,ConstSet.MinRed) AND Const.SetSize!=Fixed) **then**
13: **if** Const.SetSize!=Fixed AND expand($\mathcal{R}$, R')==TRUE **then**
14: $\mathcal{R} \leftarrow \mathcal{R} \cup R'$
15: updtFreqAndScores(elFreq, attrFreq, redScoreEl, redScoreAt,redDstC, R')
16: Stats← computeStatistics($\mathcal{R}$)
17: continue
18: compScore(elFreq,attrFreq, redScoreEl, redScoreAt, redDstC,R')
19: $R_b \leftarrow argmax_{R \in \mathcal{R}}\ score(R', R)$
20: updtFreqAndScores(elFreq, attrFreq, redScoreEl, redScoreAt,redDstC, R', R)
21: $\mathcal{R} \leftarrow \mathcal{R} \backslash R_b \cup R'$
22: **if** $\vee \in$ opSet **then**
23: **for** R$\in \mathcal{R}$ **do**
24: **if** expansionType==unguidedExpansion AND iteration==0 **then**
25: $ind \leftarrow 0$
26: **else**
27: $ind \leftarrow newRuleIt$
28: $r'_{W_1} \leftarrow argmax(R.maxRef(r), R.maxRef(\neg r),\ r \in RW_{1_{ind}} \cup SRW_1)$
29: $R_{ref} \leftarrow (r'_{W_1} \vee R.rW_1 \times R.rW_2)$
30: $r'_{W_2} \leftarrow argmax(R_{ref}.maxRef(r), R_{ref}.maxRef(\neg r), r \in RW_{2_{ind}} \cup SRW_2)$
31: $R_{ref} \leftarrow (R_{ref}.rW_1 \times r'_{W_2} \vee R.rW_2)$
32: updtFreqAndScores(elFreq, attrFreq, redScoreEl, redScoreAt,redDstC,R, R_{ref})
33: $\mathcal{R} \leftarrow \mathcal{R} \backslash R \cup R_{ref}$
34: **return** $\mathcal{R}$

If redescription set size is automatically determined, for each $R \in \mathcal{R}$ the algorithm computes $sc(R) = min_{R'' \in \mathcal{R},\ R'' \neq R}|score(R, R'')|$ (lines 11 and 16 in Algorithm 2). Measures of difference in quality characteristics between a given redescription and its closest neighbour serve as a statistics used to determine which newly created redescription should be

used to expand the redescription set (increase it in size). We compute the Tukey's range $[Q_1 - k \cdot (Q_3 - Q_1),\ Q_3 + k \cdot (Q_3 - Q_1)]$ with $k = 1.5$ and denote the upper boundary as out. For each newly created redescription R', we compute $sc(R') = min_{R'' \in \mathcal{R}} score(R', R'')$. If the redescription set contains a redescription with preferred quality score compared to newly created redescription R', the $sc(R')$ will be negative, thus the produced redescription is not allowed to expand the redescription set. Alternatively, if a newly created redescription has a positive score difference when compared to every redescription currently found in the redescription set, we require its difference to be at least as great as the computed value out from the set of score differences for redescriptions contained in the redescription set ($sc(R') \geq out$). If this condition is satisfied, the redescription is added to the redescription set, thus increasing its size (line 16 in Algorithm 2). Redescription satisfying this strict criterion has an exceptional quality, compared to all other redescriptions in the set. This can occur, for instance, if a redescription with maximal accuracy is found that describes a part of element and attribute space not explored by any redescription from the redescription set. If the required condition is not satisfied, newly created redescription is used to optimize the redescription set, possibly replacing some existing member. We are very conservative in increasing redescription set size suggested by the user because small sets are easier to explore thus preferable in redescription mining setting (Galbrun and Miettinen 2012a). The redescription set expansion follows the general algorithm structure defined in the work by Bringmann and Zimmermann (2007), though instead of enumerating all patterns, we optimize the set by creating and discarding a large number of redescriptions at each iteration. This makes the proposed algorithm memory efficient and allows redescription set optimization to be performed very quickly.

The algorithm can use three types of logical operators (disjunction, conjunction and negation) where using disjunction operators increases redescription accuracy and support (lines 22–33 in Algorithm 2). For a redescription $R = (q_1, q_2)$, we find rules r that maximize:

1. $J(supp(q_1 \vee r) \backslash supp(R), supp(q_2) \backslash supp(R))$
2. $J(supp(q_1 \vee \neg r) \backslash supp(R), supp(q_2) \backslash supp(R))$
3. $J(supp(q_1) \backslash supp(R), supp(q_2 \vee r) \backslash supp(R))$
4. $J(supp(q_1) \backslash supp(R), supp(q_2 \vee \neg r) \backslash supp(R))$

The rule r is found so that it covers elements that are supported by q_2 but not by q_1 ($R.maxRef(r'),\ r' \in RW_1$) and vice versa.

3.2 Algorithm time complexity

We train one predictive clustering tree model and a set of weak PCT learners contained in the random forest. Work from Stojanova et al. (2012) shows that predictive clustering tree construction has the worst time complexity of $O(z \cdot m \cdot |E|^2)$ to completely induce the tree, where m denotes the number of descriptive variables in a selected view and z the total number of internal nodes in the tree. The number of PCT models in a random forest is a constant defined by the user which makes the complexity of training all PCT models equal to $O(z \cdot m \cdot |E|^2)$.

The elements are stored in the HashSet and the HashMap data structure with open addressing which have the time complexity of $O(1)$ for add, remove, contains and size assuming the hash function behaves in a random enough manner (uniform hashing).

As described in our previous work (Mihelčić et al. 2015b), the initialization step has the complexity of $O(|E| \cdot (|V_1| + |V_2|))$, the PCT to rules transformation has the complexity

of $O(z)$, creation of redescriptions $O(z^2 \cdot |E|)$ and $O(z \cdot d \cdot |E|)$ if we have a balanced tree, where d equals the tree depth, which is a constant. Updating the attribute and element frequency tables and the total redescription scores has the complexity of $O(|E| + d)$ in average case and $O(z^2 \cdot (|E| + d))$ in the worst case when the set size grows proportionally with the number of created redescriptions. The computation of rules containing negation and disjunction operators has a complexity of $O(z \cdot |E|)$.

The minimization procedure has the time complexity of $O(|\mathcal{R}| \cdot ((a + a') \cdot |E| + (a^3 + a'^3) \cdot |E|))$, where a, a' represent the number of attributes in redescription rules which are constrained with the tree depth d (or a constant multiple of d in case of rules containing disjunctions). Since we have a an expandable set of redescriptions, the greatest possible number of redescriptions is a multiple of z^2. Thus, the worst case time complexity of the minimization procedure is $O(z^2 \cdot d^3 \cdot |E|^2)$ or $O(z^2 \cdot |E|^2)$ since d is a constant. In practice, due to very strict constraints, the size of a redescription set is very close to user suggested value and can be considered a constant. Thus, the average time complexity is $O(d^3 \cdot |E|)$ or $O(|E|)$, since d is a constant.

The total algorithm average time complexity equals: $O(z \cdot (|V_1| + |V_2|) \cdot |E|^2 + z^2 \cdot |E|)$ while the worst time complexity, assuming inadequate hashing function and a large resulting redescription set, is $O(z \cdot (|V_1| + |V_2| + z) \cdot |E|^2)$.

Optimizations that could speed up computing redescriptions include Local Sensitive Hashing (Cohen et al. 2000) and the use of rule indexing that allows combining only those rules certain to cross the user defined thresholds if redescription accuracy constraints are defined.

4 Algorithm evaluation and comparison

In this section, we evaluate different extensions of the CLUS-RM algorithm and compare redescriptions produced by our algorithm with the current state of the art algorithms ReReMi (Galbrun 2013), Split trees and Layered trees (Zinchenko 2014). The algorithms are compared on three datasets with different properties. Since the Split trees and the Layered trees algorithms do not work with data containing missing values, we make comparison analysis only with the ReReMi algorithm on the Country dataset. On this dataset, we evaluate redescription accuracy by using two different measures: the pessimistic Jaccard index and the query - non missing Jaccard index presented in our previous work (Mihelčić et al. 2015b). We use the notation from Galbrun and Miettinen (2012a) to denote $E_{1,1} = supp(q_1) \cap supp(q_2)$, $E_{1,0} = supp(q_1) \backslash supp(q_2)$, $E_{0,1} = supp(q_2) \backslash supp(q_1)$, $E_{1,?} = supp(q_1) \cap missing(q_2)$, $E_{?,1} = missing(q_1) \cap supp(q_2)$, where $R = (q_1, q_2)$ and $missing(q)$ represents a set of elements containing missing values for some attribute in q. Pessimistic Jaccard index is defined as: $J_{pes}(R) = \frac{|E_{1,1}|}{|E_{1,0}|+|E_{0,1}|+|E_{1,1}|+|E_{1,?}|+|E_{?,1}|+|E_{0,?}|+|E_{?,0}|+|E_{?,?}|}$ and the query-non missing Jaccard index as: $J_{qnm}(R) = \frac{|E_{1,1}|}{|E_{1,0}|+|E_{0,1}|+|E_{1,1}|+|E_{1,?}|+|E_{?,1}|}$. These two measures are used because they guarantee that each element found in redescription support has defined values for all attributes in a whole query, if only conjunction operators are used, or in a part of a query describing this element subset, if all operators are used. Query non-missing Jaccard is more optimistic than pessimistic Jaccard ($J_{pess} \leq J_{qnm}$), because it disregards elements - having undefined value for both redescription queries. We used the Siren tool (Galbrun and Miettinen 2012b) to perform redescription mining with ReReMi, Split trees and Layered trees algorithms.

4.1 Evaluation data

Evaluations and comparisons are performed on three datasets with different characteristics.

- The Country dataset (http://unctadstat.unctad.org/EN/, http://data.worldbank.org/) (Gamberger et al. 2014) describes 199 different countries in the year 2012. The dataset has two views, both containing numerical attributes with possible missing values. The first view contains 49 attributes with country information obtained from the World Bank. The second view contains 312 attributes obtained from the UNCTAD database representing the ratio of import and export of a commodity compared to total import or export of a country in the year 2012.
- The Bio dataset (Mitchell-Jones et al. 1999; Hijmans et al. 2005; Galbrun 2013) describes 2575 geographical locations in Europe. The dataset contains information about climate conditions (48 numerical attributes) for a certain location and the information about the presence of mammal species (194 boolean attributes) on these locations. The climate condition attributes contain average, maximum, minimum temperature and average monthly precipitation.
- The DBLP dataset (DBLP dataset 2010; Galbrun 2013) contains information about authors of scientific papers (6455 authors in total). The first view describes the author-conference bipartite graph (304 boolean attributes) and the second view describes the co-authorship network (6455 boolean attributes). This dataset is very sparse which makes it hard to find highly accurate redescriptions.

4.2 Algorithm parameters

In this section we explain all the parameters, constraints on redescriptions (Table 4) and settings used to perform evaluations and comparisons with various redescription mining algorithms.

Table 4 Constraints on redescriptions used by all current RM algorithms

Constraint	Value range	Usual method of selecting a value
minimal Jaccard index	[0, 1]	This parameter is obtained by experimentation. While the goal is to get as highly accurate redescriptions as possible, it is sometimes necessary to lower the initially set minimal Jaccard index to obtain redescriptions. In general, higher Jaccard index increases redescription accuracy but decreases diversity
maximal p-value	[0, 1]	This parameter is usually set to one of the two values: 0.05 or 0.01 since it denotes the significance level used as a threshold to accept redescriptions
minimal support	$[1, \lvert E \rvert]$	This parameter is usually set to values > 1, since describing only one entity is rarely interesting. Setting the threshold is domain specific. It depends on what kind of groups with respect to the size might be interesting to the domain expert. Redescriptions with very large support ($> 0.8 \cdot \lvert E \rvert$) usually have lower theoretical and empirical statistical significance since it is easier to obtain high accuracy by random sampling of queries with such a large support. Also, these queries contain large part of the value distribution for the attributes used in its queries thus random permutation of values between entities has smaller effect on the accuracy of such redescriptions

For all algorithms, we used maximal p-value threshold of 0.01 (the strictest significance level). The minimal Jaccard index was set to 0.2 level for the DBLP dataset (based on results presented in Galbrun (2013), Table 6.1, p. 46), 0.6 level for the Bio dataset (based on results in Galbrun (2013), Table 7, p. 301) and 0.5 level for the Country dataset (obtained by experimentation). Minimal support was set to 10 elements for the DBLP (based on Galbrun (2013), p. 46) and the same value is used for the Bio dataset. Minimal support is set to 5 elements for the Country dataset, since this dataset contains substantially smaller amount of elements and redescriptions describing properties of 5 different countries still seem interesting.

The algorithm specific parameter values used to create redescriptions are listed below.

- CLUS-RM - allows specifying maximum support which was set to 5036 for the DBLP dataset, 2060 for the Bio dataset and 120 for the trade dataset. Since it is possible to obtain redescriptions that describe all elements in the dataset by using disjunction, conjunction and negation operators, we set the maximum support to disallow such redescriptions. We used the *average tree depth* 8 for all datasets (this effectively determines the maximal number of attributes occurring in produced rules). 120 iterations were performed on the Bio and the DBLP dataset and 800 iterations on the Country dataset. Larger number of iterations produces larger variety of redescriptions. Since the Country dataset is much smaller than the DBLP and the Bio dataset, we could run larger number of iterations with smaller execution times than those obtained on the DBLP and the Bio dataset. We used regression trees in all experiments with random forest containing 50 trees (our estimate is that middle-sized workstations and smaller servers are already capable of running 50 threads in parallel, thus we used maximally 50 trees in the forest).
- ReReMi - we used *LHS/RHS max number of variables* $= 15$, *min contribution* $= 3$, *min uncovered* $= 200$. For the DBLP dataset we set *max number of pairs* $= 1000$, *Batch output* $= 10$ and *Batch capacity* $= 50$, for the Bio dataset we used *max number of pairs = 200*. *Batch output* and *Batch capacity* parameters were at their default values 1 and 4 respectively. For the view containing numerical values we used the default values. We also created redescription sets with the ReReMi algorithm that used only conjunction and literal level negation operators by using equivalent values of other setup parameters as to construct sets that were generated by using all operators. On the Country dataset, we used *max product buckets* $= 200$, *max number of pairs* $= 1000$, for the set in which we allowed using only logical conjunction operator and *max number of pairs* $= 500$ in case in which we mined redescriptions by using all logical operators.
- Split trees and Layered trees - we used *max rounds* $= 1000$, and *max tree depth* $= 15$.

Explanations of all parameters used for the ReReMi, Split trees and Layered trees algorithm can be seen on the web page of the tool Siren: http://siren.gforge.inria.fr/_static/miner_confdef.xml. Values for parameters *min contribution* and *min uncovered* were set after discussion with the authors of the tool, parameters specifying maximal number of attributes, bathc output, capacity and maximal number of pairs were increased compared to default values to obtain larger number and more accurate redescriptions. After obtaining redescriptions with the ReReMi, Split trees and Layered trees algorithm, we used the *Filter redundant redescriptions* option to remove duplicate and redundant redescriptions with the *max overlap option* equal to 0.99.

For the evaluation of the CLUS-RM algorithm extensions, we use the same algorithm parameters as specified earlier. The exception is the number of iterations for the DBLP dataset which is set to 40. Also, we optimize a set containing 200 redescriptions.

4.3 Evaluating CLUS-RM extensions

In this section, we evaluate the effects of (a) using random forest based augmentation and (b) redescription set creation of user suggested size without specifying redescription accuracy constraints. First, we create a redescription set by using CLUS-RM with one PCT by using parameters specified in Section 4.2. Next, we use random forest based augmentation, with identical parameters and 50 trees in the forest, to optimize the set of the same size. The final experiment uses random forest based augmentation with identical parameters as before but without specifying redescription accuracy constraints and by allowing set expansion. In all experiments, we fix one random initialization for the initial step and use it to obtain all redescription sets. Also, random seeds of PCTs and the random forest is preserved between the experiments. In this way, we can explore the effects of different modifications made to the original CLUS-RM algorithm.

The experimental results related to the described extensions are presented in Figs. 2, 3, 4 and 5.

4.3.1 Effects of using random forest based augmentation

The evaluation on the Country data, presented in Fig. 2, reveals that using random forest based augmentation in fact decreases redescription set accuracy. The Country dataset contains a small number of elements, thus the algorithm manages to create very optimized set by using only one PCT. The large number of additional, diverse, redescriptions created with the random forest of PCTs is used to describe elements that are not described often in the set by using different subsets of attributes. This is reflected by lower average element and attribute Jaccard index in the set produced with the algorithm using random forest based augmentation. The algorithm iteratively describes the fraction of elements that can be described very accurately until the occurrence frequency of these elements in redescriptions does not become to high. When this happens, a number of accurate redescriptions are replaced with redescriptions that increase diversity but have lower accuracy. According to the one - tailed Mann - Whitney U test of statistical significance, the redescription set produced by using random forest tends to contain redescriptions with smaller average element (significant with $p = 0.01381$) and attribute Jaccard index (significant with $p < 2.2 \cdot 10^{-17}$) when query non-missing Jaccard was used and with smaller average attribute Jaccard index (significant with $p = 1.6 \cdot 10^{-9}$) when pessimistic Jaccard was used. Additional benefit of using random forest based augmentation on this set is that the produced set tends to have smaller query size in redescriptions (significant with $p = 0.01677$) when query non-missing Jaccard was used and (significant with $p = 8.4 \cdot 10^{-14}$) when pessimistic Jaccard was used.

To show that using random forest also increases the number of highly accurate redescriptions, we perform an additional experiment on the Country data by using query non-missing Jaccard index as accuracy measure. In this experiment, we set very strict accuracy threshold (minimal Jaccard index ≥ 0.9) required for redescription to be considered as a candidate to optimize redescription set. We return all distinct, highly accurate redescriptions created by CLUS-RM in 300 algorithm iterations by using one PCT and a PCT with random

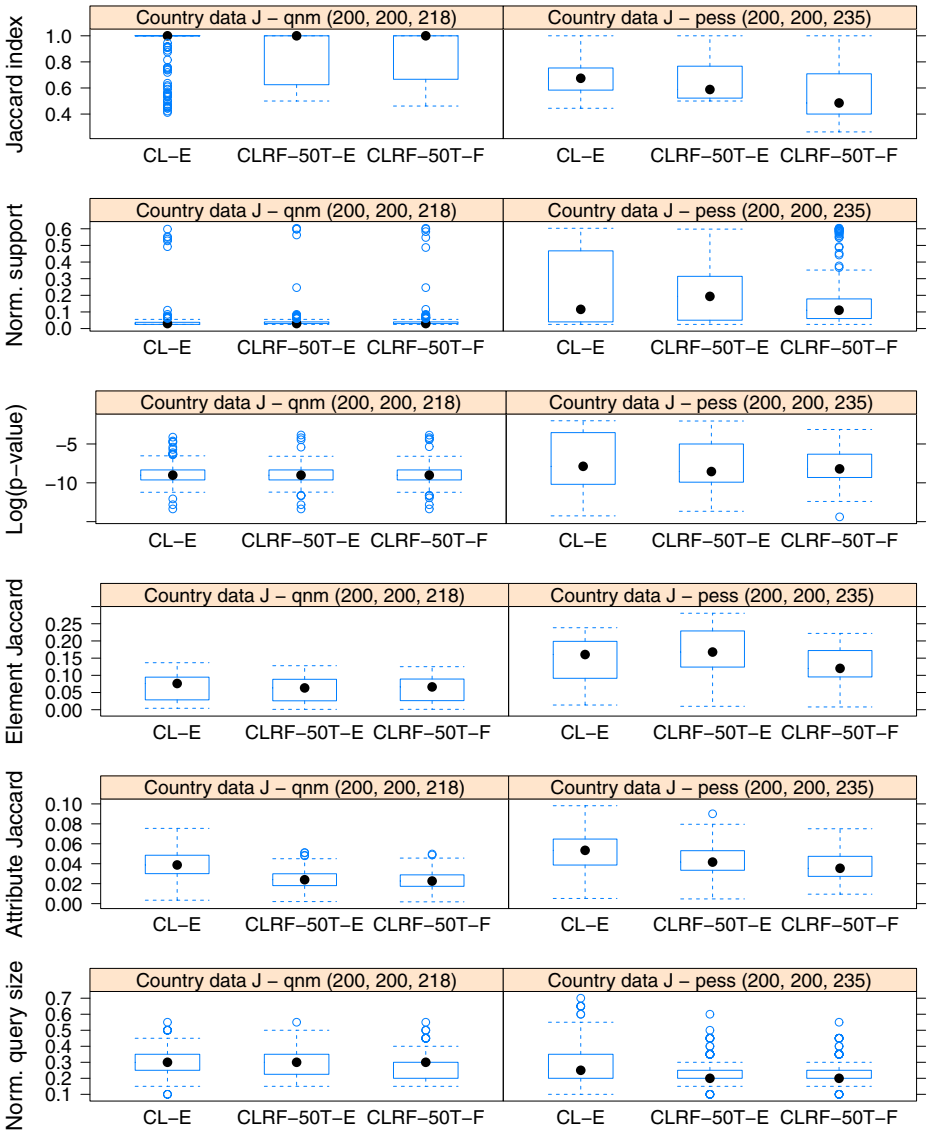

Fig. 2 Comparisons of redescription sets of size 200 produced by CLUS-RM using one PCT (CL-E), a PCT and a random forest containing 50 PCTs (CLRF-50T-E), and the CLUS-RM with flexible set size without specifying redescription accuracy using a PCT and a random forest containing 50 PCTs (CLRF-50T-F). The comparison is performed on the Country data

forest of PCTs. The change of number of highly accurate redescriptions through iterations are presented in Fig. 3 and the comparative histogram displaying redescription accuracy distribution is shown in Fig. 4.

The results presented in Figs. 3 and 4 demonstrate the superior number of highly accurate redescriptions created by CLUS-RM algorithm augmented with random forest containing

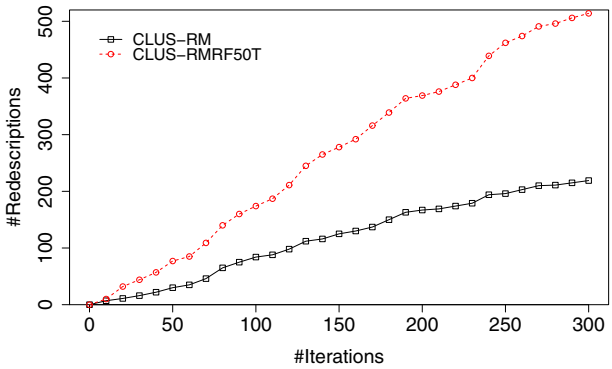

Fig. 3 Comparison of number of produced redescriptions with accuracy ≥ 0.9 by the CLUS-RM and the augmented variant with random forest containing 50 PCTs

50 PCTs. The difference between the number of generated highly accurate redescriptions increases at each algorithm iteration.

Using random forest based augmentation to create redescriptions on the Bio dataset, presented in Fig. 5, allows creating redescription set containing redescriptions that tend to have higher accuracy ($p = 3.96 \cdot 10^{-11}$) than those produced when only one PCT is used. It also tends to have smaller average attribute Jaccard index ($p = 5.1 \cdot 10^{-5}$).

On the DBLP dataset (see Fig. 5), using random forest allows creating redescription set that tends to contain redescriptions with higher accuracy ($p = 0.01088$) than those produced when only one PCT is used. Average element and attribute Jaccard index tend to be lower in the redescription set produced by using random forest, both with $p < 2.2 \cdot 10^{-16}$. Redescription query size tends to be lower in the redescription set produced by using random forest ($p = 0.04614$).

The experimental results presented in Figs. 2 and 5 suggest that using random forest in CLUS-RM allows creating redescription sets with significantly lower average attribute Jaccard index on all used datasets which is important for exploring different associations. The same experiments suggest that it often results in obtaining significantly smaller

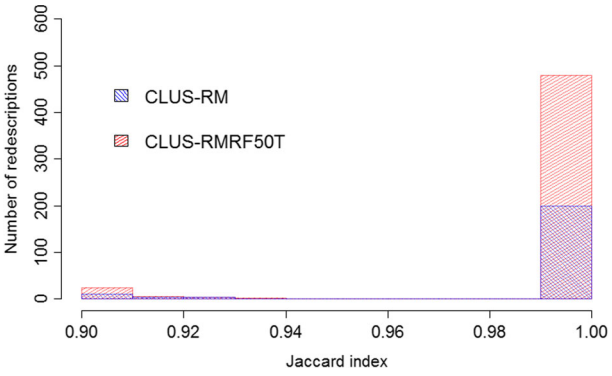

Fig. 4 Redescription accuracy distribution comparison between a set created by CLUS-RM and the augmented variant with random forest with 50 PCTs

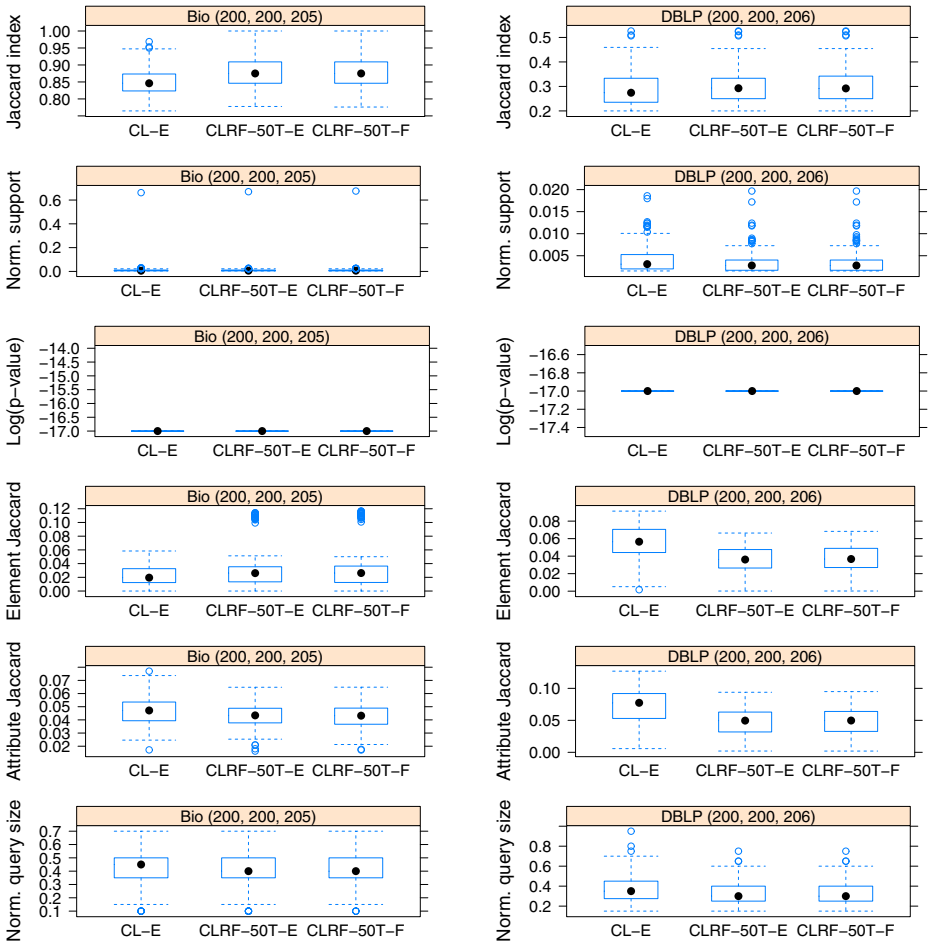

Fig. 5 Comparisons of redescription sets of size 200 produced by CLUS-RM using one PCT (CL-E), a PCT and a random forest containing 50 PCTs (CLRF-50T-E), and the CLUS-RM with flexible set size without specifying redescription accuracy using a PCT and a random forest containing 50 PCTs (CLRF-50T-F). The comparison is performed on the Bio data (*left*) and the DBLP data (*right*)

redescription queries which is important for redescription understandability. The results presented in Figs. 3, 4 and 5 show that it significantly increases redescription accuracy.

Since our method optimizes multiple objective criteria, it is not always possible to obtain domination even when the number of highly accurate redescriptions increases. This is visible from the results presented in Figs. 2, 3 and 4. Here we can see that although the method augmented with random forest produces significantly larger number of highly accurate redescriptions, the overall redescription accuracy in the optimized redescription set decreases. The reason for this is that the benefits of describing larger number of diverse elements from the dataset by using more diverse attributes using redescriptions with smaller queries outweighs the benefits of adding more accurate but redundant and more complex redescriptions. On the DBLP dataset, the augmented model outperforms the basic algorithm on all measures (though it has slightly smaller redescription support).

4.3.2 Effects of creating redescription set of variable size without specifying accuracy constraints

The automatic set expansion procedure used on the Country dataset (Fig. 2) increased the size of redescription set from initial 200 to 218 when query non-missing Jaccard was used. 10 redescriptions out of 18 have the maximal accuracy 1.0. The difference in accuracy between the set obtained by setting the redescription accuracy threshold and the set obtained without setting the threshold is not statistically significant. When pessimistic Jaccard was used, the procedure increased the size of redescription set with 35 additional redescriptions. However, the trade-off between accuracy and diversity resulted in lowering redescription accuracy to increase the element and attribute diversity.

The redescription accuracy in the set created without specifying accuracy constraints on the Bio dataset (Fig. 5) is not significantly different from that created with the fixed accuracy threshold.

On the DBLP dataset (Fig. 5), the difference in accuracy between redescription set created by specifying redescription accuracy threshold and the set created without specifying this threshold is not statistically significant.

The results on all datasets, with the exception of using pessimistic Jaccard on the Country dataset, demonstrate that no significant drop in redescription accuracy occurs when redescription accuracy threshold is not set. The drop in accuracy occurs because of the trade-off between diversity, query size and accuracy.

4.4 Comparison with state of the art methods

Redescription mining algorithm comparison was mainly done in the literature by selecting and discussing properties of individual redescriptions. We try to make objective evaluation of redescription sets produced by different algorithms by using the same set of redescription constraints. Another condition we imposed is to have the same size of the final redescription sets. This is done by first finding redescription set with the ReReMi, Split trees, Layered trees algorithm, and then forcing the same size of the redescription set on the CLUS-RM, since it produces much more redescriptions than these algorithms.

The results are divided based on usage of logical operators. In the first experiment we allow using disjunctions, conjunctions, negations (DCN) and in the second experiment only conjunctions and negations (CN). For the generated redescription sets, we plot comparative boxplots for the Jaccard index, the log_{10} of the $p - value$, the average element and attribute Jaccard index and the redescription query size. We also compute the Man-Whitney U test to assess the statistical significance of the difference in algorithm performance. We only analyse redescription sets containing at least 10 redescriptions satisfying constraints. As a consequence, we can not make comparisons with Split trees and Layered trees algorithm in the CN mode on the DBLP and Bio dataset, and with Layered trees algorithm on the DBLP dataset in the DCN mode. We perform comparisons on the Country data only with ReReMi algorithm - by using only conjunction operator in CN mode. Other algorithms can not work on data containing missing values.

4.4.1 Comparison on the Country dataset

Comparative results on the Country dataset that contains missing values are obtained by optimizing J_{pes} with ReReMi algorithm and recalculating the score for each redescription to J_{qnm}. An optimized redescription set with the CLUS-RM was also created by using the

pessimistic Jaccard index as one of the optimization criteria. The resulting sets are compared based on several quality criteria.

The results in Fig. 6 show that the redescription set produced by our approach has higher median for redescription accuracy when all operators are used and query non-missing Jaccard is used to evaluate redescription accuracy. A slightly broader distribution is a result of redescription diversification. The Mann-Whitney U test of statistical significance shows that the set produced with CLUS-RM tends to contain more accurate redescriptions (significant with $p = 4.545 \cdot 10^{-5}$), it also tends to contain more significant redescriptions

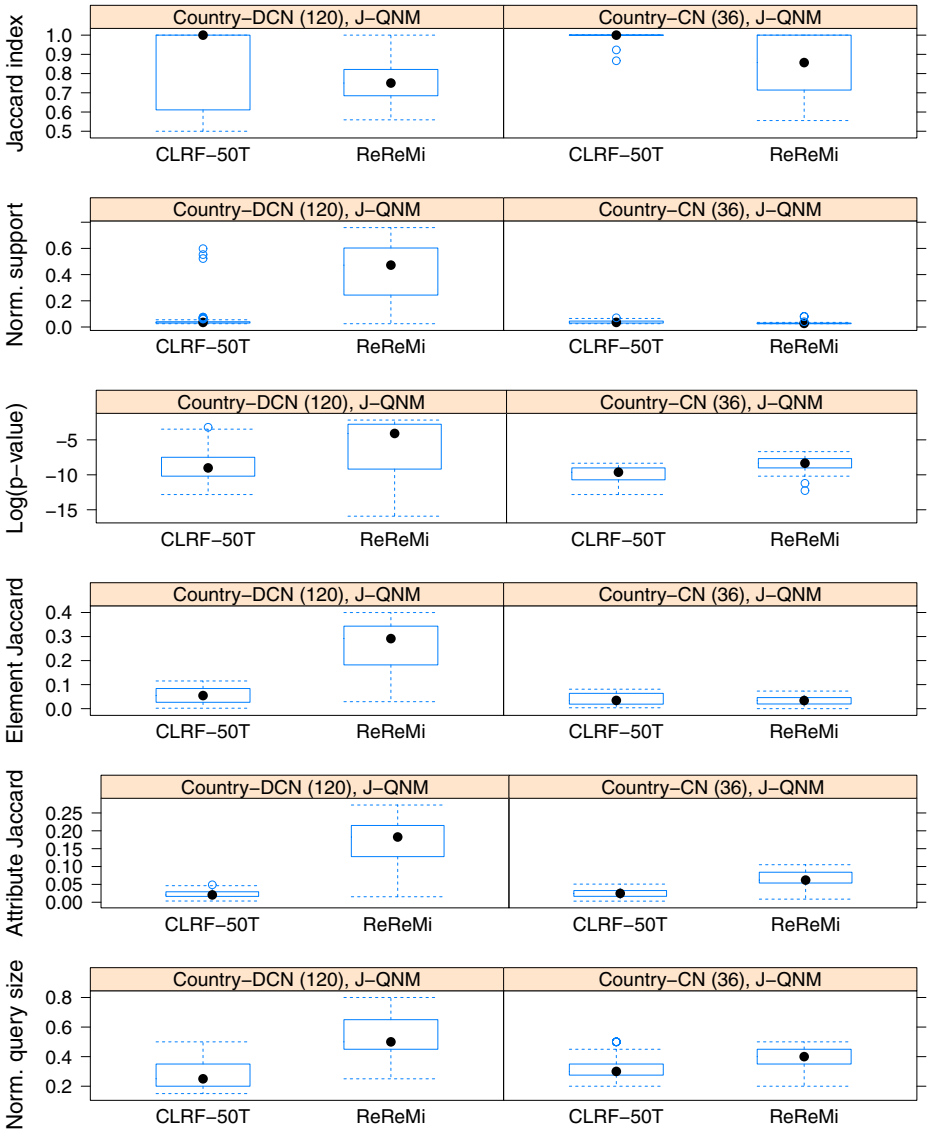

Fig. 6 Comparison of redescription sets produced by CLUS-RM and ReReMi algorithms on the Country dataset. Redescription accuracy is evaluated with the query non-missing Jaccard index

(significant with $p = 2 \cdot 10^{-9}$). The redescription set produced by the CLUS-RM tends to contain redescriptions with lower average element and attribute Jaccard index and with smaller query size (significant with $p < 2.2 \cdot 10^{-16}$). Redescriptions in the set produced by the CLUS-RM tend to contain redescriptions with smaller support (significant with $p < 2.2 \cdot 10^{-16}$). When only conjunction operators are allowed and query non-missing Jaccard is used, all measured quality criteria, except the average element Jaccard index, show that redescription set produced by CLUS-RM has significant advantage over ReReMi produced set. p-values obtained with one - tailed Mann-Whitney U test, presented in redescription quality criteria order as in Fig. 6, are $p_J = 1.9 \cdot 10^{-7}$, $p_{supp} = 9.9 \cdot 10^{-5}$, $p_{pVal} = 1.3 \cdot 10^{-6}$, $p_{EJ} = 0.795$, $p_{AJ} = 9.3 \cdot 10^{-15}$, $p_{size} = 0.0007$. Redescription set (DCN) produced by ReReMi has the element coverage (EC) 1.0, the attribute coverage (AC) 0.35 and the set (CN) has $EC = 0.53$, $AC = 0.36$. The redescription set (DCN) produced by CLUS-RM has $EC = 0.99$ and $AC = 0.59$ and the set (CN) has $EC = 0.56$, $AC = 0.36$.

When pessimistic Jaccard index is used (Fig. 7) to evaluate redescription accuracy, the redescription set created by CLUS-RM contains redescriptions that tend to have lower accuracy compared to ReReMi algorithm ($p = 0.01559$), they tend to have smaller support ($p = 8.84 \cdot 10^{-16}$), lower redescription p-value ($p = 3.88 \cdot 10^{-07}$), lower average element ($p = 2.076 \cdot 10^{-14}$) and attribute ($p < 2.2 \cdot 10^{-16}$) Jaccard index and smaller redescription query size ($p < 2.2 \cdot 10^{-16}$). When only conjunction operators are used, the CLUS-RM produced set contains redescriptions that tend to have higher redescription accuracy ($p = 7.873 \cdot 10^{-7}$), larger redescription support ($p = 0.00074$), lower redescription p-value ($p = 4.956 \cdot 10^{-6}$), lower average attribute Jaccard index ($p = 0.02652$) and smaller redescription query size ($p = 0.001727$). Redescription set created by CLUS-RM by optimizing pessimistic Jaccard index has $EC = 1.0$ and $AC = 0.5$ in the DCN mode, and $EC = 0.39$ and $AC = 0.31$ in the CN mode. The element coverage is slightly lower compared to ReReMi produced redescription set in the CN mode while the attribute coverage is comparable in CN mode and higher for CLUS-RM in the DCN mode. While ReReMi returned 2 redescriptions with $J_{pess} = 1.0$, CLUS-RM produced redescription set with 15 such redescriptions.

4.4.2 Comparison on the Bio dataset

Comparison results of redescription sets produced by CLUS-RM and ReReMi algorithm on the Bio dataset are presented in Fig. 8.

The comparison results on the Bio dataset suggest that redescription set created by CLUS-RM contains redescriptions that tend to have higher accuracy compared to ReReMi algorithm ($p = 1.599 \cdot 10^{-5}$) when all logical operators are used to create redescriptions. They also tend to have lower redescription p-values ($p = 2.052 \cdot 10^{-9}$), lower average element ($p < 2.2 \cdot 10^{-16}$) and attribute ($p = 7.24 \cdot 10^{-8}$) Jaccard index as well as smaller redescription query size ($p = 0.03107$). However, they also tend to have smaller redescription support ($p < 2.2 \cdot 10^{-16}$). Our approach finds many redescriptions closer to minimal support boundary on the Bio dataset which complements ReReMi redescriptions that have a drift towards redescriptions with maximum support as reported in the work from Galbrun (2013). When only conjunction operators are used to create redescriptions, our approach created redescription set that tends to contain more accurate redescriptions ($p = 8.31 \cdot 10^{-15}$), lower average element ($p = 1.403 \cdot 10^{-15}$) and attribute ($p = 7.666 \cdot 10^{-16}$) Jaccard index and smaller redescription query size ($p = 6.156 \cdot 10^{-5}$). It also tends to contain redescriptions with smaller redescription support ($p = 1.71 \cdot 10^{-15}$). The redescription set produced with ReReMi algorithm has $EC = 1.0$ and $AC = 0.39$ in

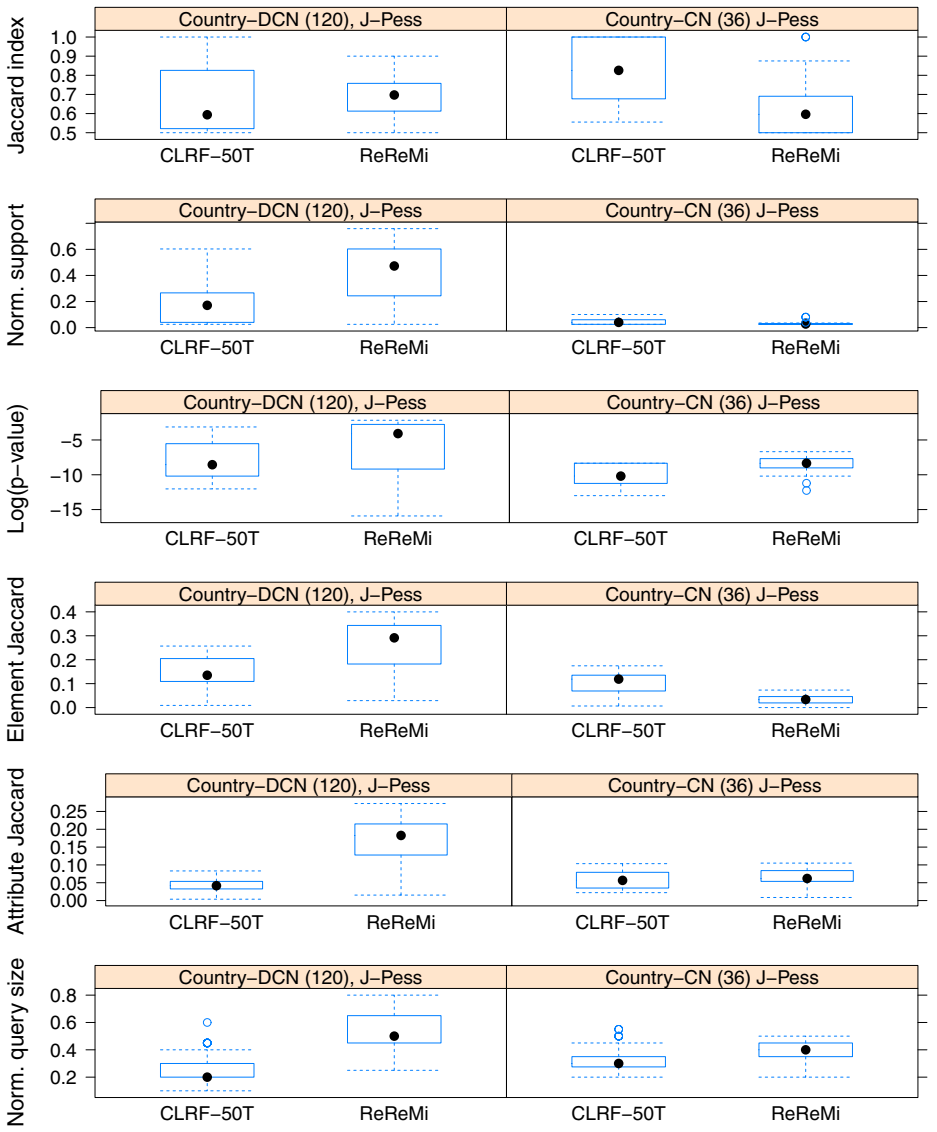

Fig. 7 Comparison of redescription sets produced by CLUS-RM and ReReMi algorithms on the Country dataset. Redescription accuracy is evaluated with the pessimistic Jaccard index

DCN mode while CLUS-RM produced redescription set has $EC = 0.93$ and $AC = 0.59$. In the CN mode, the ReReMi produced redescription set has $EC = 0.98$ and $AC = 0.37$ while CLUS-RM produced redescription set has $EC = 0.95$ and $AC = 0.52$. The redescription sets produced with ReReMi and CLUS-RM have comparable element coverage but the redescription set produced with CLUS-RM has higer attribute coverage.

The comparison results of redescription sets produced by CLUS-RM, Split trees and Layered trees algorithm on the Bio dataset is available in Fig. 9.

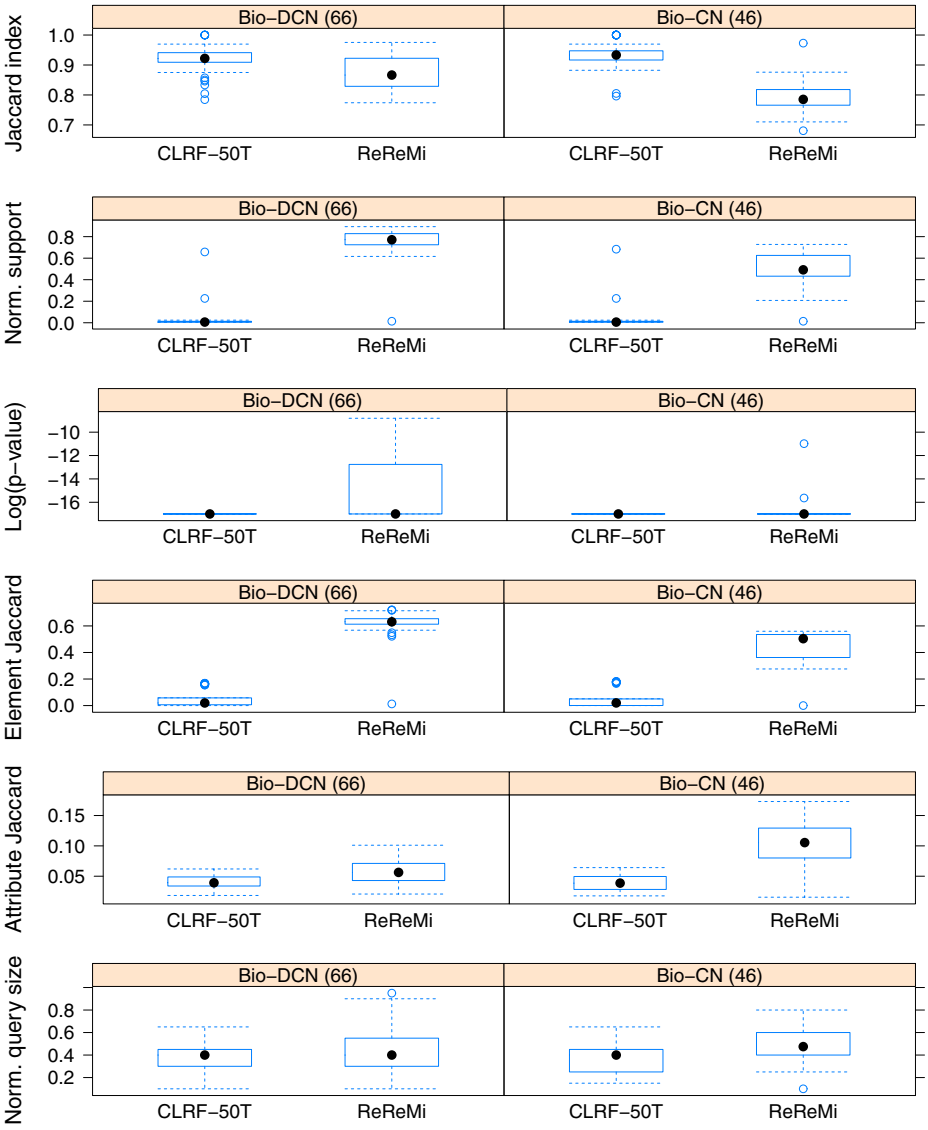

Fig. 8 Comparison of redescription sets produced by CLUS-RM and ReReMi algorithms on the Bio dataset

We perform comparisons of redescription sets created by CLUS-RM, Split trees and Layered trees algorithms by using all logical operators to construct redescriptions. The results suggest that the redescription set produced by CLUS-RM contains redescriptions that tend to have higher accuracy ($p = 2.738 \cdot 10^{-15}$), lower redescription p-value ($p = 0.02165$), lower average element ($p = 2.464 \cdot 10^{-11}$) and attribute ($p = 3.47 \cdot 10^{-16}$) Jaccard index and smaller redescription query size ($p = 0.02877$) compared to the redescription set created by Split trees algorithm. However, CLUS-RM produced redescription set also contains redescriptions that tend to have a smaller support ($p = 1.092 \cdot 10^{-14}$). The CLUS-RM produced redescription set has $EC = 0.75$ and $AC = 0.57$ while the Split trees

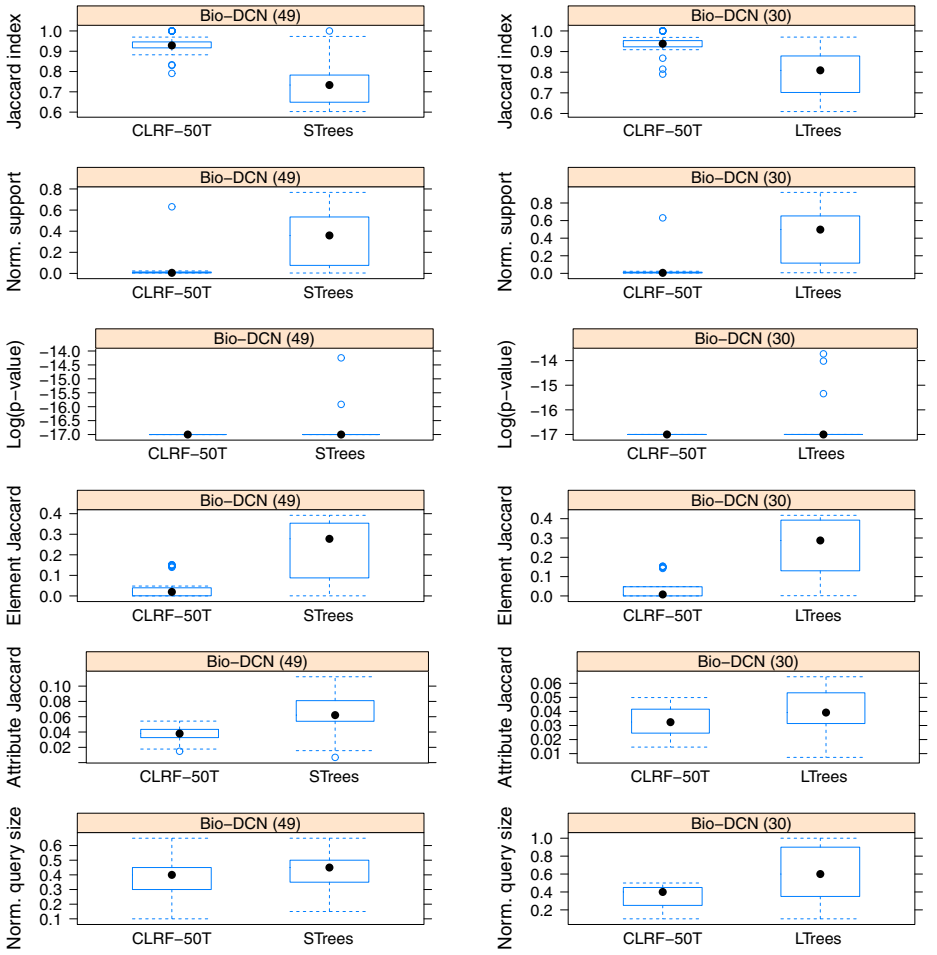

Fig. 9 Comparison of redescription set produced by CLUS-RM with the set produced by Split trees (*left*) and Layered trees (*right*) on the Bio dataset

algorithm produced redescription set with $EC = 0.98$ and $AC = 0.32$. The redescription set produced with CLUS-RM contains redescriptions that tend to have higher accuracy ($p = 2.978 \cdot 10^{-7}$), lower redescription p-value ($p = 0.04076$), lower average element ($p = 5.691 \cdot 10^{-8}$) and attribute ($p = 0.01856$) Jaccard index and smaller redescription query size ($p = 0.0006035$) compared to redescription set created by Layered trees algorithm. The redescription set produced by the CLUS-RM algorithm contains redescriptions that tend to have smaller support than redescriptions contained in the redescription set produced by the Layered tree algorithm ($p = 1.721 \cdot 10^{-8}$). The redescription set produced by CLUS-RM algorithm has $EC = 0.74$ and $AC = 0.45$ while the redescription set produced by Layered trees algorithm has $EC = 1.0$ and $AC = 0.53$.

Lower element coverage in CLUS-RM produced redescription set compared to redescription sets produced by Layered trees and Split trees algorithms is the consequence of a relatively small redescription set size: 49 and 30 redescriptions.

4.4.3 Comparison on the DBLP dataset

The comparison results of redescription sets produced by CLUS-RM, ReReMi and Split trees algorithm on the DBLP dataset is available in Fig. 10. The DBLP dataset is very sparse and it is difficult to produce many highly accurate redescriptions. The results suggest that the redescription set produced by CLUS-RM algorithm contains redescriptions that tend to have higher accuracy ($p < 2.2 \cdot 10^{-16}$) and higher redescription support ($p = 1.23 \cdot 10^{-10}$), smaller redescription query size ($p = 0.005913$) compared to redescription set produced by the ReReMi algorithm when all logical operators were used to create redescriptions. Though, the redescription set produced by CLUS-RM contains redescriptions that tend to have higher redescription p-value ($p = 2.466 \cdot 10^{-14}$) and average element ($p < 2.2 \cdot 10^{-16}$) and attribute ($p < 2.2 \cdot 10^{-16}$) Jaccard index. The redescription set produced by CLUS-RM has $EC = 0.99$ and $AC = 0.06$ while the redescription set produced by ReReMi has $EC = 0.781$ and $AC = 0.326$. Since the authors form examples and attributes in this dataset, one potential explanation for smaller attribute coverage by CLUS-RM is that it concentrated the search to parts of DBLP network that can be described very accurately which constrained the diversity of authors and conferences occuring in redescription queries. This conclusion is also indicated by the higher attribute and element Jaccard index of the produced redescriptions which is visible on all performed experiments on this dataset. When only conjunction operators were used, the redescription set produced by CLUS-RM contains redescriptions that tend to have higher accuracy ($p < 2.2 \cdot 10^{-16}$), though they tend to have smaller support ($p = 2.133 \cdot 10^{-5}$), higher element ($p < 2.2 \cdot 10^{-16}$) and attribute ($p < 2.2 \cdot 10^{-16}$) Jaccard index and larger redescription query size ($p = 9.94 \cdot 10^{-15}$). The redescription set produced by the CLUS-RM algorithm has $EC = 0.044$ and $AC = 0.028$ while redescription set produced by ReReMi has $EC = 0.188 =$ and $AC = 0.044$.

The comparison with the redescription set produced by Split trees algorithm, available in Fig. 10 (right), shows that redescription set produced by CLUS-RM algorithm contains redescriptions that do not have significant difference in accuracy, support and redescription p-value compared to redescriptions contained in the set created by Split trees algorithm. They tend to have redescriptions with smaller query size ($p = 1.145 \cdot 10^{-7}$) but they also tend to have larger average element ($p = 0.0003779$) and attribute ($p = 1.516 \cdot 10^{-8}$) Jaccard index. The redescription set produced by CLUS-RM has $EC = 0.067$ and $AC = 0.028$ while redescription set produced by Split trees has $EC = 0.085$ and $AC = 0.049$.

The results presented in this section lead us to conclude that the CLUS-RM algorithm outperforms other redescription mining approaches in the CN mode with respect to redescription accuracy. With the exception of the DBLP data, it also tends to have smaller average attribute Jaccard index, smaller redescription p-values and smaller query size. In the DCN mode, the approach outperformed other approaches in redescription accuracy, with the exception of the ReReMi algorithm when pessimistic Jaccard index is used to evaluate redescriptions on the Country dataset, and the Split trees on the DBLP dataset, where the difference in accuracy is not statistically significant.

The majority of produced redescriptions by the CLUS-RM contain conjunction and negation operators as opposed to redescriptions produced by other approaches that mostly contain disjunction operators. CLUS-RM uses disjunction operators sparingly by design because it requires redescriptions to have the accuracy larger than the minimal accuracy threshold in order to apply disjunction operator. This disallows CLUS-RM to create different disjunction - based redescriptions that describe unrelated parts of element space (and can have very high accuracy). Such redescriptions are found by ReReMi algorithm which

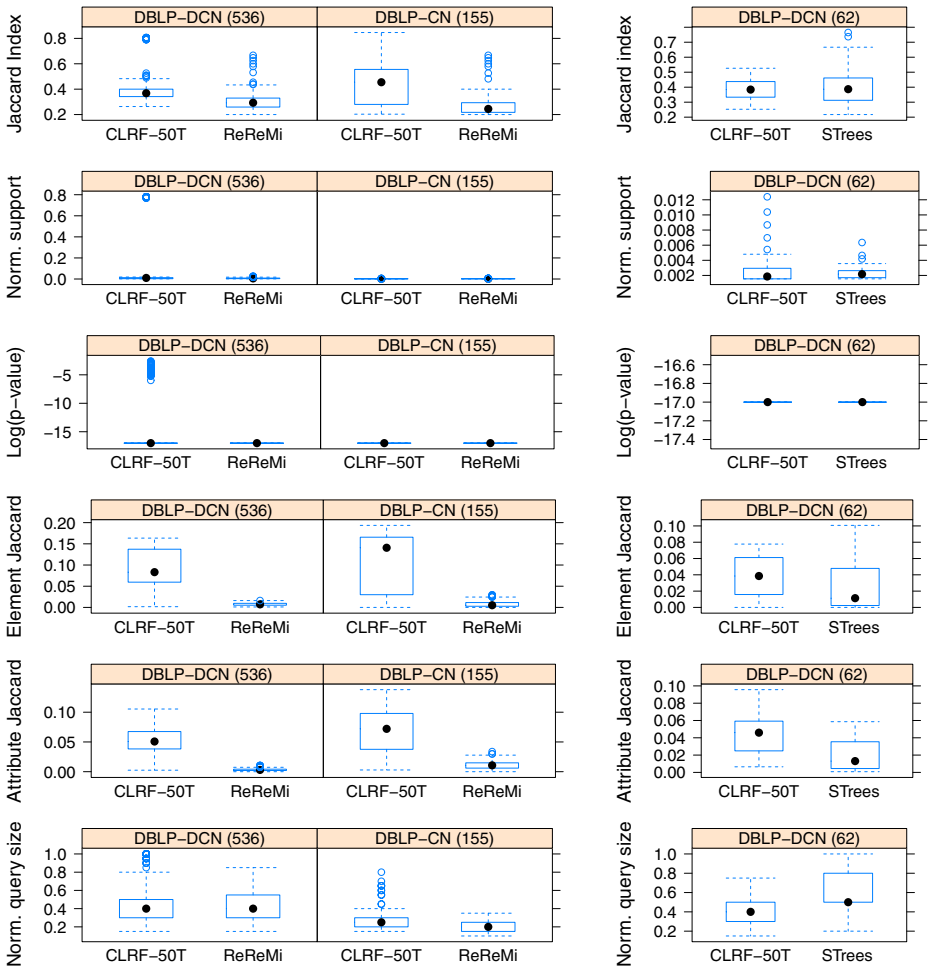

Fig. 10 Comparison of redescription sets produced by CLUS-RM and ReReMi algorithms (*left*), and the CLUS-RM and Split trees algorithm (*right*) on the DBLP dataset

is discussed by Galbrun (2013). This affects the number of highly accurate redescriptions produced by the CLUS-RM compared to other approach in DCN mode.

4.5 Redescription examples

In this section, we present top two redescriptions, by accuracy, found by each approach on the datasets used for evaluation. If there are multiple candidates with the same accuracy we choose redescriptions with shorter query size or smaller p-value. We compare these redescriptions by their structure and quality. The explanation of the meaning of these redescriptions along with a list describing all used attributes in redescription queries of these examples can be seen in Online resource 1.

4.5.1 Examples produced on the Country dataset

Redescriptions presented in Table 5 show that both CLUS-RM and ReReMi managed to find two redescriptions with maximal accuracy 1.0 with both Jaccard index variants. Redescriptions found by CLUS-RM have lower p-value and larger support. Structurally,

Table 5 Examples produced by RM algorithms on the Country dataset

Redescriptions	J	supp	p-value	Algorithm
$0.1 \leq PG \leq 1.1 \wedge 64.4 \leq POP_{15-64} \leq 68.1 \wedge 51.7 \leq CC \leq 171.0$ $0.2 \leq E/I_{41} \leq 2.2 \wedge 0.3 \leq E/I_{61} \leq 4.9 \wedge$ $0.6 \leq E/I_{80} \leq 1.3 \wedge 0.0 \leq E/I_{95} \leq 0.5$	1.0	11	$2.2 \cdot 10^{-12}$	CLUS-RM (J_{qnm})
$7.3 \leq CR\_COV \leq 100 \wedge 15.5 \leq P_{64} \leq 21.1 \wedge -2.4 \leq BAL \leq 14.4 \wedge 73.3 \leq EMSF \leq 91.5 \wedge 6.2 \leq ST \leq 166.6$ $0.0 \leq E/I_{46} \leq 0.95 \wedge 0.7 \leq E/I_{83} \leq 4.3 \wedge$ $17.0 \leq I_{26} \leq 26.0 \wedge 10.0 \leq I_{14} \leq 22.0$	1.0	14	$1.5 \cdot 10^{-13}$	CLUS-RM (J_{qnm})
$1.1 \leq AGR\_F \leq 7.8 \wedge 3.1 \leq M \leq 6.2 \wedge$ $34.1 \leq EIM \leq 49.4$ $1.1 \leq E/I_{92} \leq 2.3 \wedge 9.0 \leq E_{91} \leq 16.0 \wedge$ $0.5 \leq E/I_{20} \leq 1.4 \wedge 1.0 \leq I_{71} \leq 1.0$	1.0	8	$6.4 \cdot 10^{-11}$	CLUS-RM (J_{pess})
$0.1 \leq PG \leq 0.7 \wedge 17.2 \leq P_{64} \leq 20.8 \wedge$ $13.7 \leq RP \leq 32.1$ $3.0 \leq E_{66} \leq 6.0 \wedge 1.0 \leq E/I_{14} \leq 1.1$	1.0	7	$2.4 \cdot 10^{-10}$	CLUS-RM (J_{pess})
$M \leq 95.5 \wedge 75.9 \leq LF \wedge 1.9 \leq PG$ $((94.0 \leq I_{97} \leq 99.0 \wedge E/I_{69} \leq 0.03) \vee 31.0 \leq E_{22} \leq 34.0 \vee 5.382 \leq E/I_{43} \leq 7.813) \wedge$ $1.0 \leq E_{37}$	1.0	10	$6.3 \cdot 10^{-12}$	ReReMi (J_{qnm})
$M \leq 6.2 \wedge 15.3 \leq P_{64} \leq 17.8169$ $(2.0 \leq I_{82} \leq 2.0 \vee 0.7 \leq E/I_{85} \leq 1.6) \wedge$ $1.0 \leq E_{76} \leq 3.0 \wedge 0.5 \leq E/I_{25} \leq 1.4 \wedge 0.4 \leq E/I_{71} \leq 1.8$	1.0	13	$3.4 \cdot 10^{-13}$	ReReMi (J_{qnm})
$21.6 \leq RP \leq 37.5 \wedge 74.2 \leq CR\_COV \wedge$ $45.0 \leq LF \leq 53.5$ $69.0 \leq E_{13} \leq 86.0 \wedge I_{33} \leq 0.0 \wedge 0.1 \leq E/I_{38} \leq 1.0 \wedge 0.3 \leq E/I_{66} \leq 6.9$	1.0	6	$9.7 \cdot 10^{-10}$	ReReMi (J_{pess})
$85.7 \leq LM \wedge 40.8 \leq P_{14} \leq 43.5 \wedge 2.5 \leq PG \leq 3.3$ $4.0 \leq E_{72} \leq 11.0 \wedge 1.0 \leq I_{60} \wedge 3.0 \leq I_{66} \leq 4.0 \wedge 1.1 \leq E/I_{45}$	1.0	5	$4.6 \cdot 10^{-9}$	ReReMi (J_{pess})

redescriptions created with CLUS-RM contain only conjunction operators, while two redescriptions produced by ReReMi contain complex queries containing conjunction and disjunction operators. This makes CLUS-RM produced redescriptions easier to understand. Redescriptions produced by CLUS-RM contain longer queries describing countries by using general country information (the first presented query in the pair) while ReReMi produced redescriptions contain longer queries describing countries by their trading patterns (the second presented query in the pair).

Redescriptions produced by CLUS-RM mostly describe European countries. Top two most accurate redescriptions produced by ReReMi and presented in Table 5 are not as homogeneous as redescriptions produced by CLUS-RM with respect to location of described

Table 6 Examples produced by RM algorithms on the Bio dataset

Redescriptions	J	supp	p-value	Algorithm
$\neg(-8.3 \leq t_4^{\sim})$ PB	0.95	36	0.0	CLUS-RM (Bio)
$-8.4 \leq t_{11}^{\sim} \leq -5.98 \wedge 6.6 \leq t_6^{\sim} \leq 11.1 \wedge$ $78.9 \leq p_7 \leq 104.2 \wedge 14.4 \leq t_7^{+} \leq 18.8$ $\neg M \wedge B \wedge EWV \wedge W \wedge LS \wedge NB$	1.0	15	0.0	CLUS-RM (Bio)
$4.7 \leq t_3^{+} \leq 19.8$ $\neg M \vee C$	0.88	1726	0.0	CLUS-RM (Bio)
$-11.9 \leq t_3^{+} \leq -7.3$ PB	0.97	36	0.0	ReReMi (Bio)
$((((-12.2 \leq t_1^{-} \vee t_7^{+} \leq 13.5 \vee -12.2 \leq t_2^{\sim} \leq$ $-11.8 \vee 13.3 \leq t_8^{\sim} \leq 13.8 \vee -2.4 \leq t_{11}^{\sim} \leq$ $-1.7 \vee -7.6 \leq t_{12}^{\sim} \leq -7.5) \wedge -12.6 \leq t_3^{-} \leq$ $-11.0) \vee 1.2 \leq t_4^{+} \leq 1.2) \wedge -2.4 \leq t_3^{+} \leq$ $-1.5) \vee -6.5 \leq t_2^{+} \leq -6.4 \vee -4.58 \leq t_4^{\sim} \leq$ $-4.55 \vee 12.5 \leq t_6^{\sim} \leq 12.5$ $\neg GRBV \wedge \neg W$	0.98	2294	0.0	ReReMi (Bio)
$(64.8 \leq p_{10}^{\sim} \wedge p_8^{\sim} \leq 2.2) \vee (34.4 \leq p_4^{\sim} \wedge p_9^{\sim} \leq$ $14.9 \wedge 2.2 \leq p_8^{\sim})$ CSM	1.0	10	0.0	Spl. Trees (Bio)
$(-16.7 \leq t_3^{\sim} \wedge t_3^{\sim} \leq -11.2)$ PB	0.97	36	0.0	Spl. Trees (Bio)
$16.6 \leq t_7^{+} \vee (16.6 \leq t_7^{+} \wedge 10.8 \leq t_9^{+})$ $(LW \wedge \neg AF) \vee (LW \wedge AF \wedge EH) \vee$ $(\neg LW \wedge \neg AF)$	0.97	2370	$9.5 \cdot 10^{-15}$	Lay. Trees (Bio)
$t_3^{+} \leq -7.0$ PB	0.95	36	0.0	Lay. Trees (Bio)

countries. They describe countries located in Africa, Asia and Europe. A detailed description of the meaning of these redescriptions along with the interpretation and confirmations from domain knowledge can be seen in Section S2.1 of the Online resource 1.

4.5.2 Examples produced by RM algorithms on the Bio dataset

Table 6 contains top two redescriptions (by accuracy) produced by each approach on the Bio dataset.

The exact locations used as examples in this dataset can be seen in Galbrun (2013), p. 50, Figure 6.1. On this dataset, we provide one additional redescription created by CLUS-RM as example of a redescription with large support. This redescription contains disjunction and negation operators. However, the top two redescriptions created by CLUS-RM contain only conjunction and negation operators. Presented redescriptions created by ReReMi and Layered trees contain all three logical operators and redescription examples created by Split trees algorithm contain conjunctions and disjunctions.

Redescriptions, presented in Table 6, with large support (>1000 locations) are very uninformative because they contain negations of mammal species inhabiting geographical locations. All four algorithms discovered redescription describing habitats in Europe of the Polar Bear but use temperature in different months to provide information about the weather conditions on these locations. A detailed description of the meaning of these redescriptions

Table 7 Examples produced by RM algorithms on the DBLP dataset

Redescriptions	J	supp	p-value	Algorithm
$\neg HICSS \wedge ISWC \wedge \neg ICWS \wedge EC-Web$ $AM \wedge CS \wedge DO$	0.83	10	0.0	CLUS-RM (DBLP)
$SEBD \wedge SIGMOD \wedge LPNMR$ $TE \wedge GG$	0.67	10	0.0	CLUS-RM (DBLP)
$SEBD \wedge LPNMR \wedge SIGMOD$ $TE \wedge GT$	0.67	10	0.0	ReReMi (DBLP)
$EC-Web \wedge ISWC$ $DO \wedge SH$	0.65	11	0.0	ReReMi (DBLP)
$(ITCC \wedge ISWC \wedge \neg EC-Web) \vee$ $(\neg HICSS \wedge ISWC \wedge EC-Web)$ CS	0.76	13	0.0	Spl. Trees (DBLP)
$(ICIP(1) \wedge OTMW \wedge \neg EC-Web) \vee$ $(\neg HICSS \wedge ISWC \wedge EC-Web)$ SH	0.74	14	0.0	Spl. Trees (DBLP)
$ISWC \wedge \neg HICSS \wedge \neg SIGIR$ $(SH \wedge CS \wedge \neg AG) \vee (\neg SH \wedge YSA)$	0.85	11	0.0	Lay. Trees (DBLP)
$ITCC \wedge SEBD \wedge \neg WETICE$ $(GM \wedge \neg AMa) \vee (\neg GM \wedge EM))$	0.81	13	0.0	Lay. Trees (DBLP)

along with the interpretation and confirmations from domain knowledge can be seen in Section S2.2 of the Online resource 1.

4.5.3 Examples produced by RM algorithms on the DBLP dataset

Table 7 contains top two redescriptions (by accuracy) produced by each approach on the DBLP dataset.

Structurally, CLUS-RM and ReReMi produced the most understandable redescriptions containing only conjunction and (CLUR-RM) negation operators. Split trees and Layered trees use all operators to create redescriptions which requires analysing queries in parts to understand the relationship of parts of a query to the corresponding part of redescription support which it describes.

A detailed description of the meaning of these redescriptions along with the interpretation and confirmations from domain knowledge can be seen in Section S2.3 of the Online resource 1.

We can see from the example tables (Tables 6 and 7) that the most accurate redescriptions created by the approaches on the Bio and the DBLP dataset have large similarities. All four approaches found a set of locations corresponding to a habitat of a Polar Bear on the Bio dataset but used different climate indicators to describe the weather on these locations. All algorithms discovered very similar sets of co-authors using slightly different authors and conferences in redescription queries.

5 Conclusions

This work introduces a novel redescription mining algorithm which optimizes a redescription set of user suggested size. The algorithm is based on multi-target predictive clustering trees, which allows using element coverage by rules constructed on one view as targets for the construction of rules from the other view. One predictive clustering tree is used to create rules that are employed to guide the search, while additional random forest of predictive clustering trees is used to construct rules that increase redescription accuracy and diversity. Produced redescriptions incrementally improve the redescription set by using a predefined set of criteria (the Jaccard index, the p-value, the element and the attribute Jaccard index and the exclusive coverage). The ability to construct many different redescriptions and use them to optimize a redescription set differentiates the approach from currently proposed solutions and enables removing some user-defined constraints from the redescription mining process which is a desirable property (Galbrun 2013). The most important constraint not required by our approach is the Jaccard index threshold which is often determined by experimentation. Moreover, our approach expands the redescription set in very conservative manner which reflects the goal to present accurate and understandable redescription set of a user suggested size. Using random forest as the augmentation model decreases attribute redundancy and increases overall redescription accuracy in the output redescription sets in majority of experiments. It also increases the number of produced highly accurate redescriptions.

The results of algorithm comparisons show that our approach outperforms other approaches with respect to redescription accuracy when disjunction operators are not used in redescription construction. When all operators are used, CLUS-RM outperforms other approaches in majority of comparisons with respect to redescription accuracy. The final redescription sets contain redescriptions with smaller support, though the overall element and attribute coverage is comparable to other approaches. In general, CLUS-RM creates

many different redescriptions of various support which can be obtained by increasing minimal support constraint or increasing the redescription set size.

We have demonstrated the advantages of our approach over current state of the art methods and provided exhaustive analysis that shows it creates many complementary redescriptions to those produced by currently proposed approaches. The produced redescriptions by our approach are structurally different, mostly containing conjunction operators in redescription queries and using disjunction operators only to improve accuracy of those redescriptions with accuracy above some predefined threshold. This increases understandability and eliminates (if high enough threshold is defined) creation of redescriptions describing unrelated parts of element space. Finally, we show that among top two redescriptions by accuracy, our approach has comparable performance with respect to other approaches. Among the example redescriptions, several redescriptions produced by different approaches have a large similarity in described elements and contain related queries. This is especially visible on the DBLP and the Bio dataset.

Acknowledgments The authors would like to acknowledge the European Commission's support through the MAESTRA project (Gr. no. 612944), the MULTIPLEX project (Gr.no. 317532), the InnoMol project (Gr. no. 316289), and support of the Croatian Science Foundation (Pr. no. 9623: Machine Learning Algorithms for Insightful Analysis of Complex Data Structures).

References

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on management of data* (pp. 207-216). Washington: D.C.

Bickel, S., & Scheffer, T. (2004). Multi-View Clustering. In *Proceedings of the 4th IEEE international conference on data mining, 19–26, Washington*.

Blockeel, H. (1998). *Top-down induction of first order logical decision trees. Phd thesis, Katholieke Universiteit Leuven, Department of Computer Science*.

Bringmann, B., & Zimmermann, A. (2007). The chosen few: on identifying valuable patterns. In *Proceedings of the 7th IEEE international conference on data mining* (pp. 63–72). Omaha.

Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J.D., & Yang, C. (2000). Finding interesting associations without support pruning. In *ICDE, 489–499*.

DBLP dataset (2010). http://dblp.uni-trier.de/db.

Galbrun, E. (2013). Methods for Redescription mining. Phd thesis, University of Helsinki.

Galbrun, E., & Kimmig, A. (2014). Finding relational redescriptions. *Machine Learning*, 225–248.

Galbrun, E., & Miettinen, P. (2012a). From black and white to full color: extending redescription mining outside the Boolean world. *Statistical Analysis and Data Mining*, 284–303.

Galbrun, E., & Miettinen, P. (2012b). Siren an interactive tool for mining and visualizing geospatial redescriptions. *KDD*, 1544–1547.

Galbrun, E., & Miettinen, P. (2012c). A Case of Visual and Interactive Data Analysis: Geospatial Redescription Mining. Instant Interactive Data Mining Workshop @ ECML-PKDD.

Gallo, A., Miettinen, P., & Mannila, H. (2008). Finding subgroups having several descriptions: algorithms for redescription mining. In *Proceedings of the SIAM international conference on data mining* (pp. 334–345). Georgia: Atlanta.

Gamberger, D., & Lavrač, N. (2002). Expert-guided subgroup discovery: methodology and application. *Journal of Artificial Intelligence Research*, *17*, 501–527.

Gamberger, D., Mihelčić, M., & Lavrač, N. (2014). Multilayer clustering, a discovery experiment on country level trading data. In *Proceedings of the 17th international conference on discovery science* (pp. 87–98). Slovenia: Bled.

Giacometti, A., Li, D.H., Marcel, P., & Soulet, A. (2014). 20 Years of pattern mining: a bibliometric survey. *SIGKDD Explor. Newsl.*, 41–50.

Han, J., Cheng, H., Xin, D., & Yan, X. (2007). Frequent pattern mining, current status and future directions. *Data Mining and Knowledge Discovery*, *15*, 55–86.

Hijmans, R.J., Cameron, S., Parra, L., Jones, P., & Jarvis, A. (2005). Very high resolution interpolated climate surfaces for global land areas. *International Journal of Climatology*, *25*, 1965–978. www.worldclim.org.

Knobbe, A.J., & Ho, E.K.Y. (2006). Pattern teams. In *Proceedings of the 10th european conference on principles and practice of knowledge discovery in databases* (pp. 577–584). Germany: Berlin.

Kocev, D.K., Vens, C., Struyf, J., & Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, 817–833.

Lavrač, N., Kavšek, B., Flach, P., & Todorovski, Lj. (2004). Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, *5*, 153–188.

Mihelčić, M., Džeroski, S., Lavrač, N., & Šmuc, T. (2015a). Redescription mining with multi-label predictive clustering trees. In *Proceedings of the 4th workshop on new frontiers in mining complex patterns* (pp. 86–97). Portugal: Porto.

Mihelčić, M., Džeroski, S., Lavrač, N., & Šmuc, T. (2015b). Redescription mining with multi-target predictive clustering trees (2015b). In *New frontiers in mining complex patterns - 4th international workshop, NFMCP 2015, held in conjunction with ECML-PKDD 2015, porto, Portugal, September 7, 2015, Revised Selected Papers*, (Vol. 9607 pp. 125–143).

Mitchell-Jones, A.J., Amori, G., Bogdanowicz, W., Krystufe, B., Reijnders, P., Spitzenberger, F., Stubbe, M., Thissen, J., Vohralik, V., & Zima, J. (1999). *The atlas of european mammals*. London: Academic Press. www.european-mammals.org.

Mooney, C.H., & Roddick, J.F. (2013). Sequential pattern mining – approaches and algorithms. *ACM Computing Surveys*, *45*(2).

Parida, L., & Ramakrishnan, N. (2004). Redescription mining: structure theory and algorithms. In *Proceedings of the 20th national conference on artificial intelligence* (pp. 837–844). Pennsylvania: Pittsburgh.

Piccart, B. (2012). Algorithms for multi-target learning. Phd thesis, Katholieke Universiteit Leuven.

Ramakrishnan, N., Kumar, D., Mishra, B., Potts, M., & Helm, R.F. (2004). Turning CARTwheels: an alternating algorithm for mining redescriptions. In *Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 266–275). Seattle, WA: ACM.

Stojanova, D., Ceci, M., Appice, A., & Džeroski, S. (2012). Network regression with predictive clustering trees. *Data Mining and Knowledge Discovery*, 378–413.

UNCTAD Database, http://unctadstat.unctad.org/EN/.

van Leeuwen, M., & Galbrun, E. (2015). Association discovery in two-view data. *IEEE Transactions on Knowledge and Data Engineering*, *27*, 3190–3202.

World bank database, http://data.worldbank.org/.

Zaki, M.J., & Ramakrishnan, N. (2005). Reasoning about sets using redescription mining. In *Proceedings of the 11th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 364–373). Chicago, Illinois: ACM.

Zinchenko, T. (2014). Redescription mining over non-binary data sets using decision trees. Masters thesis, Universität des Saarlandes.