A CbO-based Algorithm for Mining Class Relevant Patterns

Hirohisa Seki and Taiki Yamada

Dept. of Computer Science, Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan
seki@nitech.ac.jp, 28114135@stn.nitech.ac.jp

Abstract. In this paper, we consider the problem of mining class relevant patterns from a given context, where each object in the context has a label which is either "positive" (i.e., target-class) or "negative" (i.e., non target-class). Garriga *et al.* studied the notion of *relevancy* in FCA (Formal Concept Analysis). Based on their work, we propose a CbO-based algorithm which, while traversing the search space of closed sets on the positives, performs pruning based on a relevance check; it checks whether or not a pattern (concept) is dominated by another pattern. The pruning method performs two kinds of checks; dominance check between a node and its children nodes, and dominance check between a node and its sibling nodes in a CbO-based search tree. Although extra costs are required for performing the latter check, our experimentally results show that the proposed approach has outperformed the conventional methods in the literature.

1 Introduction

Concise representation is essential in a pattern mining task to handle large amounts of patterns generated during the mining process. Closed patterns and closed itemsets have been extensively studied in the field of FCA (Formal Concept Analysis) [2] and data mining (for example, [14,15]) to reduce the number of patterns without losing their information.

Lavrač *et al.* proposed the theory of *relevance* [10,11], where each object in a given context has a label which is either "positive" (i.e., target-class) or "negative" (i.e., non target-class), and patterns take the form of sets of attributes. A pattern is then considered to be *more relevant than* (or *dominating*) another pattern if it covers at least all positives (i.e., target-class objects) covered by the irrelevant (or dominated) pattern, but no additional negative. Garriga *et al.* [4] have studied the notion of *relevancy* in terms of closed patterns, and relevant patterns are shown to be useful for many classification tasks.

Several methods have been proposed to generate relevant patterns, using the notion of closed patterns [4,12,5]. In particular, the approach in [4] first generates the set of all closed patterns, and then removes from them those irrelevant ones by checking a certain pruning condition. In [5], Grosskreutz first studied an algorithm which is polynomial time in the size of the input and output, and

 Published in Francisco J. Valverde-Albacete, Martin Trnecka (Eds.): Proceedings of the 15th International Conference on Concept Lattices and Their Applications, CLA 2020, pp. 223–234, 2020.

also proposed a memory-efficient divide-and-conquer algorithm which visits a superset of relevant patterns. In the divide-and-conquer algorithm, a pruning criterion based on the dominance check in terms of *negative supports* (*supp*$^-$-check for short) has been introduced.

In this paper, we propose a CbO-based algorithm [8] which, while traversing the search space of closed sets on the positives, performs pruning using two types of the dominance check between patterns: the *supp*$^-$-check and another check, the *siblings-check* for short, between sibling nodes in a CbO-based search tree. Our experimental results on some datasets show the effectiveness of our algorithm; they show that the newly introduced dominance check works well for reducing irrelevant patterns.

The organization of the rest of this paper is as follows. We first summarize some basic notations and definitions of relevant pattern mining in Sect. 2. We then explain our approach to mining relevant patterns from a formal context in Sect. 3, and show some experimental results in Sect. 4. Finally, we give a summary of this work in Sect. 5.

2 Mining Relevant Patterns

2.1 Preliminaries

We use some basic notions of FCA in [2,3]. In FCA, we consider a set G of objects, a set M of attributes and a relation $I \subseteq G \times M$, such that $(g, m) \in I$ if and only if object g has the attribute m. We call such a triple $\mathbb{K} = (G, M, I)$ a *formal context*.

By A' we mean the set of attributes shared by a subset A of objects, and B' the set of objects sharing a subset B of attributes. A concept (A, B) is a maximal objects-attributes correspondence, satisfying $A' = B$ and $B' = A$. A is called the *extent* of the concept, and B its *intent*. Two formal concepts (A_1, B_1) and (A_2, B_2) are ordered by

$$(A_1, B_1) \geq (A_2, B_2) \longleftrightarrow A_1 \supseteq A_2,$$

and form a complete lattice.

We assume that each object has a class label in $\{+, -\}$. Each label is not in M. The set G of all objects are divided into two subsets: the set G^+ of those objects that are labeled by $+$, the *positives* (or the positive examples), and the set G^- of those objects that are labeled by $-$, the *negatives* (or the negative examples).

A *pattern* in $\mathbb{K} = (G, M, I)$ is a subset of the attribute set M. An object g satisfies the pattern P if g contains all attributes in P. The *occurrence set* of P, denoted by $occ(G, P)$, is the set of those objects in G which satisfy P. We simply write it by $occ(P)$, when G is obvious from the context. The *negative occurrence set* of the pattern P, denoted by $occ^-(P)$, is the set of those objects in G^- which satisfy P, i.e., $occ^-(P) = occ(G^-, P)$. The *positive occurrence set* of the pattern P, denoted by $occ^+(P)$, is defined dually, i.e., $occ^+(P) = occ(G^+, P)$.

The *support* of P, denoted by $supp(G, P)$, is the size of $occ(G, P)$. We simply write it by $supp(P)$, when G is obvious from the context. The *negative support* (*positive support*) of the pattern P, denoted by $supp^-(P)$ ($supp^+(P)$), is the size of $occ^-(P)$ ($occ^+(P)$), respectively.

The notion of a *closed* pattern is defined as usual: a pattern P is *closed* when there is no other pattern Q such that $P \subsetneq Q$ and $supp(G, P) = supp(G, Q)$.

Closed patterns can be defined in terms of the following closure operator:

$$\Gamma_G(P) = \{i \in M \mid \forall o \in occ(G, P) \text{ contains attribute } i.\},$$

i.e., the closure $\Gamma_G(P)$ of a pattern P includes all attributes that are present in all objects in G which contain all attributes in P.

2.2 Relevant Patterns

Definition 1 (relevant pattern).

A pattern P is *more relevant than* (or *dominates*) a pattern Q in G iff

- $occ^+(P) \supseteq occ^+(Q)$,
- $occ^-(P) \subseteq occ^-(Q)$.
- $\Gamma(P) \neq Q$.[1]

We call a pattern P *relevant* if there is no other pattern Q more relevant than (or dominating) P. □

The following theorem by Garriga et al. [4] gives a characterization of relevant patterns in terms of the following closure operator:

$$\Gamma^+(P) = \{i \in M \mid \forall o \in occ(G^+, P) \text{ contains attribute } i.\}$$

$\Gamma^+(\cdot)$ is called the *closure on the positives*, and a pattern P is *closed on the positives* if $\Gamma^+(P) = P$.

Theorem 1 (Garriga et al. [4]). Let $\mathcal{R}$ be the set of relevant patterns in a given context. Then, a relevant pattern P satisfies the following conditions:

- P is closed on the positives, and
- there exists no generalization $P_0 \subset P$ such that $P_0 \in \mathcal{R}$ and $supp^-(P_0) = supp^-(P)$.

□

Example 1. Figure 1 shows a small dataset and concepts generated from it. Among them, those concepts shown in a rectangle have relevant patterns with their intents; the relevant patterns are $\emptyset, 0, 1$ and 2. We often simply write a pattern by the concatenation of its attributes instead of a set notation, i.e., 0 instead of $\{0\}$, for example. Similarly for the extent of a concept.

We note that each concept is also closed on the positives. In particular, $supp^-(0) = supp^-(01) = \emptyset$, and pattern 0 is more relevant than (or dominates) pattern 01. □

[1] This condition is due to [5]. It ensures that the members of an equivalence class do not dominate each other circularly.

label		0	1	2
+	0		x	x
+	1	x		x
+	2	x	x	
+	3	x	x	x
-	4			

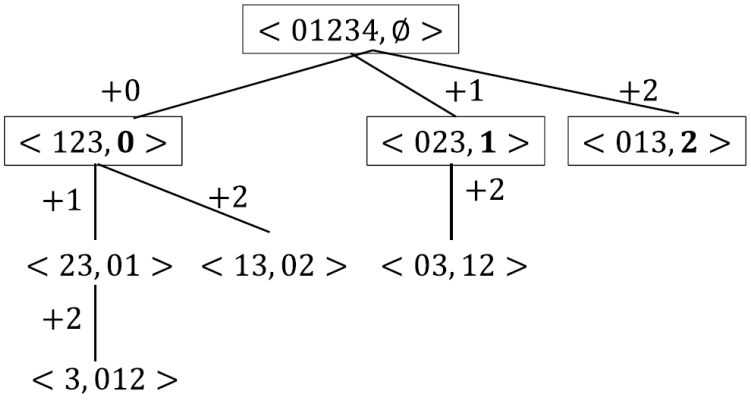

Fig. 1: Example of Closed/Relevant Patterns: Relevant patterns are shown in bold font.

3 Mining Relevant Patterns

In this section, we describe our algorithm for mining relevant patterns. We also explain its application for association rule mining in Section 3.2.

3.1 The Algorithm

Algorithm 1 shows the outline of our algorithm for mining relevant patterns from a formal context. It is based on the Close-by-One (CbO) algorithm by Kuznetsov [8]. We use a version of CbO given in [6,13], which uses a recursive procedure searching for all formal concepts in a depth-first manner. Let $\langle G, M, I\rangle$ be a given formal context, where $G = \{0, 1, \ldots, m-1\}$ and $M = \{0, 1, \ldots, n-1\}$ for some $m, n \geq 0$. Given a concept $\langle A, B\rangle$, $y \in M$ and a set of attributes I_m, function RS-GenerateFrom in Algorithm 1 recursively traverses the search space of closed sets on the positives, a *superset* of all relevant patterns, which are obtained by adding $j \in M$ to B such that $j > y$. I_m is used for a bookkeeping purpose to record those attributes not used during the process of deriving concepts.

The differences of the function with CbO are threefold: (i) we use the closure function Γ^+ (line 10), (ii) we employ the pruning methods based on the dominance check (line 14), and (iii) we perform a modified version of the canonicity

test (or the prefix-preserving test) using I_m (line 13). We call it *the I_m-canonicity test* for short.

For the pruning methods, we perform two kinds of the dominance check; the one is the dominance check between the node $\langle A, B\rangle$ and its child node $N_j = \langle C, D\rangle$, where $D = \Gamma^+(B \cup \{j\})$ (line 10), and the other is the dominance check between N_j and its sibling nodes $N_i = \langle (\Gamma^+(B \cup \{i\}))', \Gamma^+(B \cup \{i\})\rangle$ for some $i > j$ and $i \in D \setminus B$. We refer to the former check as the *$supp^-$-check*, while the latter one is referred to as the *siblings-check* for short. If D is dominated either by B or the intent of N_i, we *skip* the search tree rooted at N_j, and we record attribute j as "marked" and add it to I_m.

We use the set I_m of marked attributes in the I_m-canonicity test (line 13). Since the search tree rooted at a node constructed with each of attributes in I_m is pruned, we ignore those attributes in the canonicity test so that we check whether or not $(B \otimes j) \setminus I_m \neq \emptyset$, where $B \otimes j = (\Gamma^+(B \cup \{j\}) \setminus B) \cap \{0, 1, \ldots, j-1\}$.[2]

Algorithm 1: Generate RelSets

Input: a formal context, *minsup*: a minimun support threshold
Output: *RelSet*: the set of relevant patterns

```
1  RelSet := ∅;
2  call RS-GenerateFrom(⟨(G+)'', (G+)'⟩, 0, ∅);
3  Remove irrelevant patterns in RelSet
4  return RelSet

5  Function RS-GenerateFrom(⟨A, B⟩, y, Im) is
       input: a concept ⟨A, B⟩, an attribute y, a set of attributes Im.
6      add B to RelSet
7      if y > |M| then return
8      for j from y upto |M| do
9          if j ∉ B then
10             D ← Γ+(B ∪ {j})
11             C ← D'
12             if supp(C) < minsup then continue
13             if (B ⊗ j) \ Im ≠ ∅ then continue
14             if supp−(D) = supp−(B) or D is dominated by Γ+(B ∪ {i}) for
                ∃i > j such that i ∈ D \ B then
15                 mark j as "pruned"
16                 Im ← Im ∪ {j}
17                 continue
18             else
19                 Im1 ← Im
20                 call RS-GenerateFrom(⟨C, D⟩, j + 1, Im1)
```

[2] This notation is due to [7].

label		0	1	2	3
+	0	x	x		
-	1	x	x	x	
+	2	x			x
-	3			x	x
+	4	x		x	x
+	5			x	

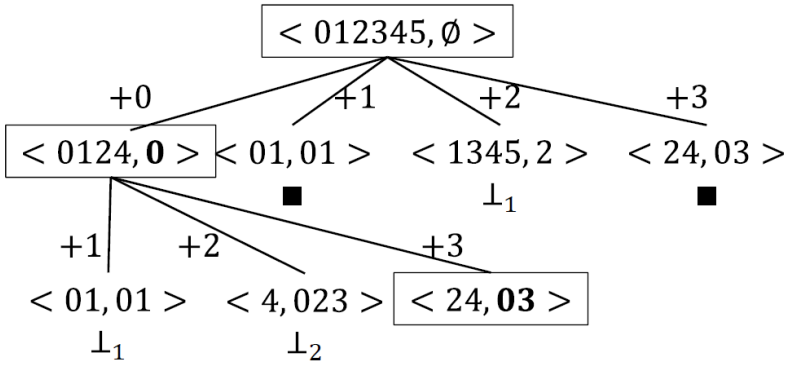

Fig. 2: Example of the Topdown Search of Algorithm 1.

Example 2. Figure 2 shows a dataset and the computation of Algorithm 1 for it. We can represent the computation of function RS-GenerateFrom in Algorithm 1 by a tree; each node represents an invocation of RS-GenerateFrom, and each edge in the tree is labeled by the current value of j which is used to compute a (new) concept (line 8). Leaf nodes denoted by black squares represent computed concepts for which the I_m-canonicity test in line 13 fails. Leaf nodes denoted by $\perp_1$ ($\perp_2$) represent computed concepts for which the $supp^-$-check (the siblings-check) in line 14 fails, respectively. Relevant patterns are $\emptyset, 0$ and 03.

□

As in Figure 2, since the computation of function RS-GenerateFrom in Algorithm 1 can be represented by a tree, we call such a tree a *cPos (search) tree*. We denote by $N_0 \vdash_T N_j$ if N_0 has a descendant node N_j in a cPos tree T. We also write by $N_0 \vdash_{T,I} N_j$ if N_0 has a descendant node N_j in a cPos tree T with the I-canonicity test. We denote simply by $B_0 \vdash_T B_j$ for short, where B_0 (B_j) is the set of attributes of N_0 (N_j), respectively. Similarly for $N_0 \vdash_{T,I} N_j$.

Lemma 1. Let $N_0 = \langle A, B\rangle$ be a node in a cPos search tree T. Suppose that $N_0 \vdash_T N_j = \langle A_j, B_j\rangle$, where $N_j = \langle (\Gamma^+(B \cup \{j\}))', \Gamma^+(B \cup \{j\})\rangle$ ($j \in M$) and

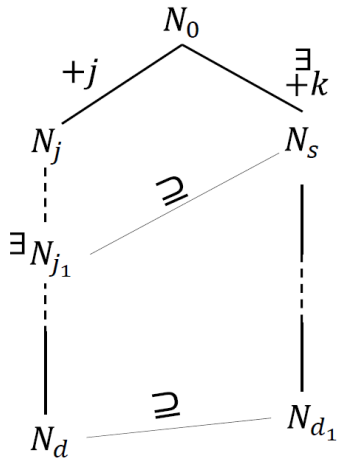

Fig. 3: The CbO-based Search Tree in Lemma 1.

$N_j \vdash_T N_d = \langle A_d, B_d \rangle$ for some node N_d in T. Consider a subset $B_{d_1} \subseteq B_d$ such that $\{y \in B_{d_1} \mid y < j+1\} = \{y \in B_d \mid y < j+1\}$. Then, there exist nodes N_s and N_{d_1} with its intent $\Gamma^+(B_{d_1})$ in T such that N_s is a sibling node of N_j and $N_s \vdash_{T,I_m} N_{d_1}$ for $I_m = B_d \setminus B_{d_1}$.

Proof. Let $k = \min\{i \in B_{d_1} \mid i > j\}$. Then, let N_s be the sibling node of N_j with its intent $B_s = \Gamma^+(B \cup \{k\})$. See Fig. 3. We show that N_s is not pruned by the I_m-canonicity test. Let N_{j_1} be the node in the branch from N_j to N_d in T such that attribute k is first introduced in the branch. For the intent B_{j_1} of N_{j_1}, we have that $B_{j_1} \supseteq B_s$. Since $\Gamma^+(\cdot)$ is monotone, it follows that $\Gamma^+(B \cup \{k\})$ passes the I_m-canonicity test.

Next, we show that there exists node N_{d_1} such that $N_s \vdash_{T,I_m} N_{d_1}$ for $I_m = B_d \setminus B_{d_1}$. We prove it by the induction on the size of $B_{d_1} \setminus B_s$. The base case $B_{d_1} = B_s$ is trivial. For the induction step, let N_{s_1} be the child node of N_s obtained by adding to B_s $k_1 = \min\{i \in B_d \setminus B_s \mid i > k\}$. We can show that $N_s \vdash_{T,I_m} N_{s_1}$ similarly to the above. The remaining proof then follows from the induction hypothesis. □

Theorem 2 (Correctness of Algorithm 1). For a given formal context and a minimum support *minsup*, (i) the set of patterns given by function RS-GenerateFrom in Algorithm 1 includes all the relevant patterns, and (ii) it is a subset of the closed on the positives with the support no less than *minsup*.

Proof. The part (ii) is rather obvious, since the patterns generated by the function are the results of Γ^+. For the part (i), since a relevant pattern is closed on the positives from Theorem 1, we show that the pruning methods, line 14 in the algorithm, do not eliminate any relevant pattern.

Let $N_0 = \langle A, B\rangle$ be a node in a cPos search tree T. Suppose that N_0 has a descendant node $N_d = \langle A_d, B_d\rangle$ in T, i.e., $N_0 \vdash_T N_d = \langle A_d, B_d\rangle$, and $I \subseteq \{y+1, \ldots, |M|-1\}$ $(0 \leq y < |M|)$ is the set of attributes added to each node in the branch br from N_0 to N_d. Let

$I_1 = \{j \in I \mid \mathit{supp}^-(B_1) = \mathit{supp}^-(\Gamma^+(B_1 \cup \{j\}))$ for the intent B_1 of a node in $br\}$, and

$I_2 = \{j \in I \mid N_j$ in br is dominated by a younger sibling node in $T\}$.

Let $I_m = I_1 \cup I_2$. We now consider subset $B_{d_1} = B_d \setminus I_m$. From Lemma 1, we then have that there exists a node N_{d_1} such that $N_0 \vdash_{T,I_m} N_{d_1}$ and its intent is $\Gamma^+(B_{d_1})$.

We note that this derivation from N_0 to N_{d_1} is not pruned in Algorithm 1. Moreover, $\Gamma^+(B_{d_1})$ is more relevant than B_d, since $\mathit{supp}^-(B_d) = \mathit{supp}^-(\Gamma^+(B_{d_1}))$. □

Grosskreutz proposed an output-polynomial time algorithm (Algorithm 2 in [5]) for enumerating relevant patterns. It, however, requires memory with the size of the output (i.e., the set of relevant patterns). To address the problem, the author also proposed a divide-and-conquer algorithm (Algorithm 3 in [5]); it uses less memory, but it visits irrelevant patterns. Algorithm 1 in the current paper is similar to that divide-and-conquer algorithm in that both visit a superset of relevant patterns, and every pattern is visited at most once during the search space traversal. The difference is that Algorithm 3 in [5] employs the supp^--check only, while Algorithm 1 in this paper employs the siblings-check as well as the supp^--check.

Algorithm 1 visits irrelevant patterns; its search space is the set of all closed on the positives in the worst case. The set of all relevant patterns is obtained by filtering in line 3. For that, we use the filtering method in [4].

3.2 Relevant Patterns and Correlation Measures

Since the framework using the support/confidence only generates too many rules, we usually use another measure to find "interesting" ones among the generated rules. *lift*-value is such a measure to find correlated rules; the lift-value of a rule $A \rightarrow c_+$, where A is a pattern and c_+ is a class label, is defined as the ratio of the probability of $P(A \wedge c_+ \mid A)$ to that of $P(c_+)$: $\mathit{lift}(A, c_+) = \frac{P(A \wedge c_+)}{P(A) \cdot P(c_+)}$. Given a contingency table in Table 1, where given m and n are both assumed to be constants, we have that $\mathit{lift}(A, c_+) = \frac{n}{m} \cdot \frac{a}{a+b}$. We use the following property of *lift*-values: for patterns A_1 and A_2,

$$\mathit{lift}(A_1, c_+) \geq \mathit{lift}(A_2, c_+) \leftrightarrow a_1 b_2 \geq a_2 b_1,$$

where a_i (b_i) $(i = 1, 2)$ is the number of *true positives* (*false positives*), respectively. We thus have that, if A_1 is more relevant than A_2, then $\mathit{lift}(A_1, c_+) \geq \mathit{lift}(A_2, c_+)$. Algorithm 1 can be therefore utilized to find association rules with high *lift*-values.

Table 1: Contingency Table for Rule $r{=}A \rightarrow c_+$.

	c_+ is true	c_+ is false	Sum_{row}
A is true	$sup(r){=}a$	b	$sup(A){=}a+b$
A is false	$m-a$	$n-m-b$	$sup(\neg A){=}\ n-(a+b)$
Sum_{col}	$sup(c_+){=}m$	$sup(c_-){=}\ n-m$	n

Table 2: Example Databases: from the UCI Machine Learning Repository.

Dataset	#obj.	#attr.	target class	#pos.
Lenses	24	9	hard	4
			soft	5
			none	15
Mushroom	8124	117	poisonous	3916
Lymphography	148	59	malign lymph	61
Soybean-small	48	72	D3	10

4 Experimental Results

We show in Table 2 some datasets used in our experiments. They are from the UCI Machine Learning [3]. The table shows some properties of each dataset, including the target classes considered and the number of their positive examples.

We have implemented our proposed method by using Java 8 on a PC with an Intel Core i7 processor running at 2.30GHz, 8GB of main memory, working under Windows 10 (64 bit). We have performed the following experiments with fixed $min\_sup = 1$ (i.e., all concepts).

4.1 Experimental Results: the Lenses Dataset

To see the effects of our pruning methods, we present some results for the Lenses dataset in Figure 4. In Algorithm 1, we incorporate into the CbO-based miner for closed on the positives (*cloPos*) two types of pruning methods: the $supp^-$-check and the siblings-check.

The figure shows the number of patterns generated in computing the closed on the positives (cloPos) and that of patterns generated by using the $supp^-$-check only, together with that of patterns generated by using both the $supp^-$-check and siblings-check.

We have tested three cases, varying target classes: hard, soft and non-contact lenses. We have observed that the pruning method by the siblings-check enables us to generate less patterns compared with the cloPos method (i.e., without

[3] http://archive.ics.uci.edu/ml/datasets/statlog+(heart).

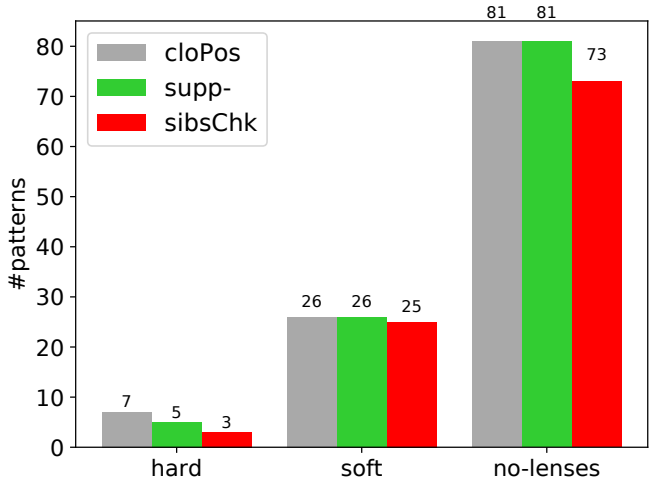

Fig. 4: Comparison in the Lenses Dataset

Table 3: #(Generated Patterns) in Lenses Dataset

	target class		
	hard	soft	non-lense
$\#(clo)$	71	77	107
$\#(cloPos)$	7	26	81
$supp^{-}$-*check*	5	26	81
sibs-check	**3**	**25**	**73**
#(RPs)	3	25	73

pruning) and $supp^{-}$-check only. Table 3 shows more detailed results, including the number of closed patterns ($\#(clo)$) and that of relevant patterns after removing irrelevant ones (#(RPs)) (in line 3 in Algorithm 1).

4.2 Experimental Results on the Other Datasets

In Figure 5, we present some results for the other three datasets in Table 2, Mushroom, Lymphography and the Soybean-small. The figure (left) shows the numbers of generated patterns by Algorithm 1. The figure (right) shows the corresponding execution times in milliseconds. We note that these figures use a logarithmic scale.

We have observed that the effects of the pruning method by the siblings-check on these datasets are more prominent than those on the previous Lenses dataset. The numbers of closed patterns ($\#(clo)$) and closed on the positives ($\#(cloPos)$)

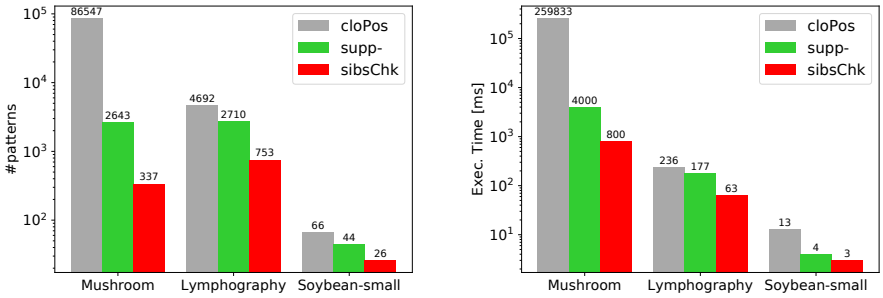

Fig. 5: #(Generated Patterns) and Execution Time for the Mushroom, Lymphography and the Soybean-small Dataset.

Table 4: Results of the Other Datasets in Table 2.

	Mushroom	Lymphography	Soybean-small
$\#(clo)$	145,353	35,905	2,036
$\#(cloPos)$	86,547	4,692	66
$supp^{-}$-*check*	2,643	2,710	44
sibs-check	**337**	**753**	**26**
#(RPs)	187	499	22

in these three datasets are orders of magnitude larger than that of the Lenses dataset. As a result, there are more chances for pruning by the siblings-check to work for reducing the number of irrelevant patterns. The execution times of these datasets are also reduced accordingly.

5 Concluding Remarks

In this paper, we have considered the problem of mining relevant patterns for labeled data from a formal context, especially focusing on the pruning methods for reducing irrelevant patterns. Unlike the divide-and-conquer based method [5] in the literature, we have proposed a CbO-based algorithm; while traversing the search space of closed sets on the positives, it performs pruning using two types of the dominance check between patterns: the $supp^{-}$-check and the siblings-check. We have empirically examined the effectiveness of our algorithm on some datasets, and shown that the effects of the siblings-check on reducing irrelevant patterns have compensated for its extra computational costs.

For future work, sine our algorithm visits a superset of relevant patterns from a formal context, it will be interesting to introduce another dominance check to reduce the search space. Another further work is to apply relevant pattern mining

to association rule mining with non-monotone measures such as *lift* (e.g., [9]). The relationship with *subgroup discovery*, e.g., recent work by Belfodil et al. [1], is also to be studied.

Acknowledgment The authors would like to thank anonymous reviewers for their useful comments on the previous version of the paper. This work is partially supported by JSPS Grant-in-Aid for Scientific Research (C) 18K11432.

References

1. Belfodil, A., Belfodil, A., Bendimerad, A., Lamarre, P., Robardet, C., Kaytoue, M., Plantevit, M.: FSSD - a fast and efficient algorithm for subgroup set discovery. In: 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA). pp. 91–99 (Oct 2019)
2. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1999)
3. Ganter, B., Kuznetsov, S.O.: Formalizing hypotheses with concepts. In: International Conference on Conceptual Structures. pp. 342–356. Springer (2000)
4. Garriga, G.C., Kralj, P., Lavrač, N.: Closed sets for labeled data. Journal of Machine Learning Research 9(Apr), 559–580 (2008)
5. Grosskreutz, H.: Class relevant pattern mining in output-polynomial time. In: SDM. pp. 284–294 (2012)
6. Krajca, P., Outrata, J., Vychodil, V.: Parallel algorithm for computing fixpoints of Galois connections. Annals of Mathematics and Artificial Intelligence 59(2), 257–272 (2010)
7. Krajca, P., Outrata, J., Vychodil, V.: Advances in algorithms based on cbo. In: CLA. vol. 672, pp. 325–337 (2010)
8. Kuznetsov, S.O.: Learning of simple conceptual graphs from positive and negative examples. In: Proc. of the Third European Conference on Principles of Data Mining and Knowledge Discovery. pp. 384–391. PKDD '99, Springer-Verlag, London, UK, UK (1999)
9. Kuznetsov, S., Makhalova, T.: On interestingness measures of formal concepts. Inf. Sci. 442(C), 202–219 (May 2018)
10. Lavrač, N., Gamberger, D., Jovanoski, V.: A study of relevance for learning in deductive databases. J. of Logic Programming 40, 215–249 (Jun 1999)
11. Lavrač, N., Gamberger, D.: Relevancy in constraint-based subgroup discovery. In: Proceedings of the 2004 European Conference on Constraint-Based Mining and Inductive Databases. pp. 243–266. Springer-Verlag, Berlin, Heidelberg (2005)
12. Lemmerich, F., Rohlfs, M., Atzmueller, M.: Fast discovery of relevant subgroup patterns. In: Twenty-Third International FLAIRS Conference (2010)
13. Outrata, J., Vychodil, V.: Fast algorithm for computing fixpoints of galois connections induced by object-attribute relational data. Information Sciences 185(1), 114–127 (2012)
14. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering Frequent Closed Itemsets for Association Rules. In: Proc. ICDT'99, pp. 398–416. Springer (1999)
15. Zaki, M.J.: Mining non-redundant association rules. Data Mining and Knowledge Discovery 9(3), 223–248 (2004)