Depth-first Traversal over a Mirrored Space for Non-redundant Discriminative Itemsets

Yoshitaka Kameya and Hiroki Asaoka

Dept. of Information Engineering, Meijo University
1-501 Shiogama-guchi, Tenpaku-ku, Nagoya 468-8502, Japan
ykameya@meijo-u.ac.jp

Abstract. Discriminative pattern mining is known under the names of subgroup discovery, contrast set mining, emerging pattern mining, etc. and has been intensively studied for the last 15 years. Based on the sophisticated techniques developed so far (e.g. branch-and-bound search, minimum support raising, and redundancy elimination including the use of closed patterns), this paper proposes an efficient exact algorithm for finding top-k discriminative patterns that are not redundant and would be of value at a later step in prediction or knowledge discovery. The proposed algorithm is unique in that it conducts depth-first search over enumeration trees in a mirrored form of conventional ones, and by this design we can keep compact the list of candidate top-k patterns during the search and consequently high the minimum support threshold. Experimental results with the datasets from UCI Machine Learning Repository clearly show the efficiency of the proposed algorithm.

Keywords: discriminative pattern mining, top-k mining, minimum support raising, closed itemsets, dual-monotonicity, suffix enumeration trees.

1 Introduction

Discriminative pattern mining is known under the names of subgroup discovery [21], contrast set mining [1], emerging pattern mining [3], supervised descriptive rule discovery [11], cluster grouping [23], and so on and has been intensively studied for the last 15 years. The obtained discriminative patterns can be used to characterize a particular class c of interest, or to build more precise classifiers. One current issue in discriminative pattern mining is to deal with the redundancy among the obtained patterns. For example, suppose that we are performing top-k mining and a pattern $\{\mathsf{A}\}$ is significantly relevant to a class c of interest. Then, the patterns including A, such as $\{\mathsf{A},\mathsf{B}\}$, $\{\mathsf{A},\mathsf{C}\}$ and $\{\mathsf{A},\mathsf{B},\mathsf{C}\}$, also tend to be relevant to c, and would occupy the list of the final top-k patterns. So we hope to find a more informative collection of top-k patterns by eliminating such redundancy in a reasonable way. For instance, let us introduce a constraint called *productivity* [20]. Then, if a pattern $\{\mathsf{A},\mathsf{C},\mathsf{D}\}$ is more relevant to the class c of interest than another pattern $\{\mathsf{A},\mathsf{C}\}$, we consider that there is something meaningful in the combination of A, C and D, but if the former is less relevant than the latter, the former can be considered redundant and removed.

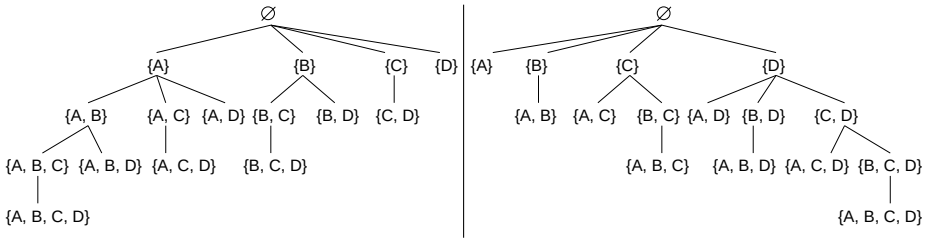

Fig. 1. A prefix enumeration tree (left) and a suffix enumeration tree (right).

In the literature of frequent/discriminative pattern mining, we have conventionally used *prefix enumeration trees* (e.g. [1, 2, 19]), illustrated in Fig. 1 (left) which include items A, B, C and D ordered as $\mathsf{A} \prec \mathsf{B} \prec \mathsf{C} \prec \mathsf{D}$. This paper, on the other hand, proposes to use *suffix enumeration trees* [10], illustrated in Fig. 1 (right), as a mirrored form of prefix enumeration trees. In a prefix (resp. suffix) enumeration tree, the parent of each node (pattern)[1] is its immediate prefix (resp. suffix). It is less known that when branching ascendingly w.r.t. $\prec$, FP-Growth [8] implicitly runs over suffix enumeration trees. The merit of using suffix enumeration trees comes from a property that *when a node $\boldsymbol{x}$ is visited in a depth-first (and left-to-right) search, all of $\boldsymbol{x}$'s sub-patterns have already been visited.* In the suffix enumeration tree in Fig. 1, when visiting $\{\mathsf{A}, \mathsf{C}, \mathsf{D}\}$, we have already visited $\{\mathsf{A}\}$, $\{\mathsf{C}\}$, $\{\mathsf{D}\}$, $\{\mathsf{A}, \mathsf{C}\}$, $\{\mathsf{A}, \mathsf{D}\}$ and $\{\mathsf{C}, \mathsf{D}\}$, but this is not the case in prefix enumeration trees. This property on the visiting order makes efficient the tests on the set-inclusion-based constraints among patterns.

Based on the observation above, we propose an efficient exact algorithm for finding top-k non-redundant discriminative patterns under two set-inclusion-based constraints among patterns. One is closedness, which has been studied in frequent pattern mining [15, 19], and the other is productivity illustrated before. We can say that the proposed algorithm has two technical contributions. First, it adopts a new, relaxed condition called *dual-monotonicity*, which is desired to be satisfied by the scores on relevance. Indeed, dual-monotonicity is shown to play a crucial role in various aspects of the proposed algorithm. Second, the proposed algorithm introduces a mirrored version of prefix-preserving closure extension in LCM [19] in order to traverse over a search space like a suffix enumeration tree. We show formally and empirically that this mirrored operation successfully keeps compact the list of candidate top-k patterns during the search and consequently high the minimum support threshold.

The remainder of this paper is outlined as follows. Section 2 describes dual-monotonicity together with the concepts and the techniques that have been developed in the literature of discriminative pattern mining. Section 3 presents our proposed method. Some experimental results are reported in Section 4, and lastly Section 5 concludes the paper.

[1] We only consider enumeration trees which have a one-to-one map between the nodes in a tree and the possible patterns. So we refer to a node by its corresponding pattern.

2 Dual-monotonicity of relevance scores

2.1 Preliminaries

First, we introduce some notations. We consider a dataset $\mathcal{D} = \{t_1, t_2, \ldots, t_N\}$ of size N, where t_i is a transaction, a set of items. The set of all items appearing in $\mathcal{D}$ is denoted by $\mathcal{X}$. Also each transaction belongs to one of pre-defined classes $\mathcal{C}$, and let c_i be the class of transaction t_i. A pattern $\boldsymbol{x}$ is a subset of $\mathcal{X}$, and let $\mathcal{P}$ be the set of all possible patterns. We say that $\boldsymbol{x}$ matches a transaction t_i when $\boldsymbol{x} \subseteq t_i$. Depending on the context, we interchangeably denote a pattern as a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, as a set $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$, or as a conjunction $\boldsymbol{x} = (x_1 \wedge x_2 \wedge \ldots \wedge x_n)$. Besides, we introduce some total order $\prec$ among items, and by default, we place the items in a transaction/pattern following $\prec$. Transaction/patterns can be ordered in a lexicographical way w.r.t. $\prec$.

The probabilities treated in this paper are all empirical ones, i.e. they are computed from the statistics on the dataset $\mathcal{D}$. First, we define some subsets of $\mathcal{D}$: $\mathcal{D}_c = \{i \mid c_i = c, 1 \leq i \leq N\}$, $\mathcal{D}(\boldsymbol{x}) = \{i \mid \boldsymbol{x} \subseteq t_i, 1 \leq i \leq N\}$ and $\mathcal{D}_c(\boldsymbol{x}) = \{i \mid c_i = c, \boldsymbol{x} \subseteq t_i, 1 \leq i \leq N\}$, where $c \in \mathcal{C}$ is the class of interest. A joint probability $p(c, \boldsymbol{x})$ is then obtained as $|\mathcal{D}_c(\boldsymbol{x})|/N$. Also we use a symbol $\neg$ for negation, e.g. we have $\mathcal{D}_{\neg c} = \mathcal{D} \setminus \mathcal{D}_c$, $p(c, \neg\boldsymbol{x}) = |\mathcal{D}_c \setminus \mathcal{D}_c(\boldsymbol{x})|/N$, $p(\neg c, \boldsymbol{x}) = |\mathcal{D}(\boldsymbol{x}) \setminus \mathcal{D}_c(\boldsymbol{x})|/N$, and so on. Using joint probabilities, marginal probabilities and conditional probabilities are computed, e.g. we obtain $p(\boldsymbol{x}) = p(c, \boldsymbol{x}) + p(\neg c, \boldsymbol{x})$, $p(c) = p(c, \boldsymbol{x}) + p(c, \neg\boldsymbol{x})$ or $p(c \mid \boldsymbol{x}) = p(c, \boldsymbol{x})/p(\boldsymbol{x})$.

2.2 Relevance Scores

As stated before, we seek for k patterns that are relevant to a class c of interest. For that, we first adopt a relevance score R_c, a function from $\mathcal{P}$ to $\mathbb{R}$ (the set of real numbers). The relevance measured by $R_c(\boldsymbol{x})$ can also be regarded as interestingness of a class association rule $\boldsymbol{x} \Rightarrow c$, where $\boldsymbol{x}$ is a pattern. Among dozens of relevance scores proposed so far [4, 10, 11], we adopt *F-score* $\mathrm{F}_c(\boldsymbol{x}) = 2p(c \mid \boldsymbol{x})p(\boldsymbol{x} \mid c)/(p(c \mid \boldsymbol{x}) + p(\boldsymbol{x} \mid c)) = 2p(c, \boldsymbol{x})/(p(c) + p(\boldsymbol{x}))$, which is a popular measure in information retrieval or evaluation of classifiers.

Now consider a class association rule $\boldsymbol{x} \Rightarrow c$ applied to the original dataset $\mathcal{D}$. Then, true positive rate (TPR) is written as $p(\boldsymbol{x} \mid c) = p(c, \boldsymbol{x})/p(c)$ in our notation, and called the *positive support* of $\boldsymbol{x}$ for class c. Similarly, false positive rate (FPR) is written as $p(\boldsymbol{x} \mid \neg c)$ and called the *negative support*. The ROC space [4, 14] is then formed by TPR (as y-axis) and FPR (as x-axis) and each pattern $\boldsymbol{x}$ is located as a point (v, u) in the ROC space, where u and v respectively indicate TPR and FPR of $\boldsymbol{x}$. For brevity, 'support' means positive support unless explicitly noted. Since we seek for the patterns relevant to a particular class c, we are only interested in the patterns $\boldsymbol{x}$ such that $p(\boldsymbol{x} \mid c) \geq p(\boldsymbol{x} \mid \neg c)$ or equivalently $p(c \mid \boldsymbol{x}) \geq p(c)$.

In the literature, the *convexity* of the relevance score has been exploited in branch-and-bound pruning [7, 13, 14, 23]. Recently Nijssen et al. introduced a property called *zero diagonal convexity* [14]: $R_c(\boldsymbol{x})$ is zero diagonal convex iff

$R_c(\boldsymbol{x})$ is convex and reaches its minimum in all the points on the diagonal TPR = FPR in the ROC space. Several popular relevance scores such as the Fisher score, information gain, Gini index, χ^2, support difference are zero diagonal convex [14]. Furthermore, zero diagonal convexity can be relaxed as a new condition called *dual-monotonicity*, which is defined as follows:

Definition 1. *Let R_c be a relevance score for a class c of interest. Then, R_c is dual-monotonic iff $R_c(\boldsymbol{x})$ is monotonically increasing w.r.t. $p(\boldsymbol{x} \mid c)$ and monotonically decreasing w.r.t. $p(\boldsymbol{x} \mid \neg c)$ wherever $p(\boldsymbol{x} \mid c) \geq p(\boldsymbol{x} \mid \neg c)$.* □

Recall here that $R_c(\boldsymbol{x})$ is a function of TPR $u = p(\boldsymbol{x} \mid c)$ and FPR $v = p(\boldsymbol{x} \mid \neg c)$. Interestingly, dual-monotonicity includes two out of Piatetsky-Shapiro's three conditions desired for relevance scores [4, 17].[2] Besides, dual-monotonicity obviously holds if $R_c(\boldsymbol{x})$ satisfies zero diagonal convexity. In contrast, for F-score, dual-monotonicity holds while convexity does not. As we will see, dual-monotonicity plays a crucial role in various aspects of the proposed algorithm.

2.3 Redundancy Elimination with Set-inclusion-based Constraints

As stated in the introduction, elimination of redundant patterns is necessary for having more informative results. In this paper, we focus on redundancy elimination based on set-inclusion-based constraints among patterns. One popular technique for this is to use the closedness constraint, and we additionally introduce a generalized version of set-inclusion-based constraint called *productivity*.

To explain the closedness constraint, we first introduce a closure operator Γ such that $\Gamma(\boldsymbol{x}, \mathcal{D}) = \bigcap_{t \in \mathcal{D}(\boldsymbol{x})} t$, where $\mathcal{D}$ is the transactions and $\boldsymbol{x}$ is some pattern. Here $\Gamma(\boldsymbol{x}, \mathcal{D})$ is called a *closure* of $\boldsymbol{x}$ w.r.t. $\mathcal{D}$. A closed pattern is then a pattern $\boldsymbol{x}$ such that $\boldsymbol{x} = \Gamma(\boldsymbol{x}, \mathcal{D})$. Each closed pattern $\boldsymbol{x}$ is the maximal pattern in an equivalence class $[\boldsymbol{x}] = \{\boldsymbol{x}' \mid \mathcal{D}(\boldsymbol{x}) = \mathcal{D}(\boldsymbol{x}')\} = \{\boldsymbol{x}' \mid \boldsymbol{x} = \Gamma(\boldsymbol{x}', \mathcal{D})\}$ and seen as a representative of $[\boldsymbol{x}]$. Since the size of $[\boldsymbol{x}]$ can be exponential, focusing only on closed patterns often leads to a significant reduction of the search space.

Next, let us consider a situation for discriminative pattern mining. Let c be a class of interest, $\mathcal{D}_c$ the transactions that belong to c, and $\boldsymbol{x}$ some pattern. Also let $\boldsymbol{x}^* = \Gamma(\boldsymbol{x}, \mathcal{D}_c)$. We further note that $\mathcal{D}_c(\boldsymbol{x}^*) = \mathcal{D}_c(\boldsymbol{x})$ since $\boldsymbol{x}^*$ and $\boldsymbol{x}$ are in the same equivalence class $[\boldsymbol{x}]$, and $\mathcal{D}'(\boldsymbol{x}^*) \subseteq \mathcal{D}'(\boldsymbol{x})$ for any transactions $\mathcal{D}'$ since $\boldsymbol{x}^*$ is the maximal pattern in $[\boldsymbol{x}]$. Then, under a dual-monotonic relevance score R_c, we have $R_c(\boldsymbol{x}^*) \geq R_c(\boldsymbol{x})$ since $p(\boldsymbol{x}^* \mid c) = p(\boldsymbol{x} \mid c)$ (from $\mathcal{D}_c(\boldsymbol{x}^*) = \mathcal{D}_c(\boldsymbol{x})$) and $p(\boldsymbol{x}^* \mid \neg c) \leq p(\boldsymbol{x} \mid \neg c)$ (from $\mathcal{D}_{\neg c}(\boldsymbol{x}^*) \subseteq \mathcal{D}_{\neg c}(\boldsymbol{x})$) [5, 18]. Now, interestingly, it is also justified from the viewpoint of the relevance score R_c to focus only on the closed patterns obtained by the closure operator to $\mathcal{D}_c$, and such closed patterns are often called the *closed-on-the-positives*. Hereafter we abbreviate $\Gamma(\boldsymbol{x}, \mathcal{D}_c)$ as $\Gamma_c(\boldsymbol{x})$ and called it the *closure of $\boldsymbol{x}$ on the positives*.

[2] The remaining one is that $R_c(\boldsymbol{x}) = 0$ when $p(\boldsymbol{x} \mid c) = p(\boldsymbol{x} \mid \neg c)$, i.e. $R_c(\boldsymbol{x})$ reaches zero in all the points on the diagonal TPR = FPR [4, 17].

In addition, we introduce another set-inclusion-based constraints called productivity, whose original version is defined with confidence (i.e. $R_c(\boldsymbol{x})$ is fixed as $p(c \mid \boldsymbol{x})$) [2, 20]. Productivity is defined as follows:[3]

Definition 2. *Let c be a class of interest. Then, for a pair of patterns $\boldsymbol{x}$ and $\boldsymbol{x}'$ in $\mathcal{P}$, $\boldsymbol{x}$ is* weaker than *$\boldsymbol{x}'$ iff $\boldsymbol{x} \supset \boldsymbol{x}'$ and $R_c(\boldsymbol{x}) \leq R_c(\boldsymbol{x}')$. A pattern $\boldsymbol{x}$ is* productive *iff $\boldsymbol{x}$ is not weaker than any sub-pattern of $\boldsymbol{x}$.* □

In the literature [5, 6, 12], a pattern $\boldsymbol{x}$ is said to be *dominated* by another pattern $\boldsymbol{x}'$ iff $\mathcal{D}_c(\boldsymbol{x}) \subseteq \mathcal{D}_c(\boldsymbol{x}')$ and $\mathcal{D}_{\neg c}(\boldsymbol{x}) \supseteq \mathcal{D}_{\neg c}(\boldsymbol{x}')$, and a pattern $\boldsymbol{x}$ is *relevant* iff $\boldsymbol{x}$ is not dominated by any other pattern. Garriga et al. [5] derived a condition equivalent to this relevance: *a pattern $\boldsymbol{x}$ is relevant iff $\boldsymbol{x}$ is closed on the positives and there is no generalization $\boldsymbol{x}' \subset \boldsymbol{x}$ closed on the positives such that $\mathcal{D}_{\neg c}(\boldsymbol{x}') = \mathcal{D}_{\neg c}(\boldsymbol{x})$*. Here it is straightforward to show that under a dual-monotonic relevance score R_c, productivity implies relevance in the sense above (i.e. productivity is a tighter constraint) among the patterns closed on the positives. From this observation, in this paper, we aim to find top-k productive closed-on-the-positives.

2.4 Branch-and-Bound Pruning in Top-k Mining

Suppose that we conduct a branch-and-bound search for top-k patterns under a dual-monotonic relevance score R_c. Also consider an anti-monotonic upper bound $\overline{R}_c(\boldsymbol{x})$ of $R_c(\boldsymbol{x})$ of a pattern $\boldsymbol{x}$. Then, if it is found that $\overline{R}_c(\boldsymbol{x}) < R_c(\boldsymbol{z})$, where $\boldsymbol{z}$ is the pattern with the k-th greatest score, we can safely prune the subtree rooted by $\boldsymbol{x}$. This pruning exploits the anti-monotonicity of $\overline{R}_c$, which guarantees $R_c(\boldsymbol{x}') \leq \overline{R}_c(\boldsymbol{x}') \leq \overline{R}_c(\boldsymbol{x}) < R_c(\boldsymbol{z})$ for any super-pattern $\boldsymbol{x}'$ of $\boldsymbol{x}$.

Several previous methods obtain the upper bound by considering the most optimistic scenario. Since $R_c(\boldsymbol{x})$ is dual-monotonic, by definition $R_c(\boldsymbol{x})$ is monotonically increasing (resp. decreasing) w.r.t. $p(\boldsymbol{x} \mid c)$ (resp. $p(\boldsymbol{x} \mid \neg c)$), and both $p(\boldsymbol{x} \mid c)$ and $p(\boldsymbol{x} \mid \neg c)$ are anti-monotonic w.r.t. pattern-inclusion. Thus, the most optimistic scenario when extending $\boldsymbol{x}$ into $\boldsymbol{x}'$ is that $p(\boldsymbol{x}' \mid c)$ remains $p(\boldsymbol{x} \mid c)$ and $p(\boldsymbol{x}' \mid \neg c)$ turns to be zero. So a general heuristic for obtaining an upper bound $\overline{R}_c(\boldsymbol{x})$ is to substitute $p(\boldsymbol{x} \mid \neg c) := 0$ into the definition of $R_c(\boldsymbol{x})$.[4] After having the upper bound $\overline{R}_c(\boldsymbol{x})$ where $p(\boldsymbol{x} \mid \neg c)$ is constant at zero, $\overline{R}_c(\boldsymbol{x})$ is always anti-monotonic w.r.t. pattern-inclusion thanks to the dual-monotonicity of R_c. For example, F-score is defined as $\mathrm{F}_c(\boldsymbol{x}) = 2p(c \mid \boldsymbol{x})p(\boldsymbol{x} \mid c)/(p(c \mid \boldsymbol{x}) + p(\boldsymbol{x} \mid c))$, so we obtain its upper bound as $\overline{\mathrm{F}}_c(\boldsymbol{x}) = 2p(\boldsymbol{x} \mid c)/(1 + p(\boldsymbol{x} \mid c))$. The above heuristic is applicable to any dual-monotonic relevance scores.[5]

Then, we translate branch-and-bound pruning into minimum support raising [9]. In top-k mining, we usually use an ordered list of candidate patterns,

[3] Weakness in this definition is also called *strong dominance* in the context of relevant explanation in Bayesian networks [22].

[4] Equivalent substitutions are also possible: $p(c \mid \boldsymbol{x}) := 1$, $p(\neg\boldsymbol{x} \mid \neg c) := 1$, and so on.

[5] However, for some scores whose upper bounds are always high, there would be no chance of pruning. For instance, a relevance score called growth rate [3] $\mathrm{GR}_c(\boldsymbol{x}) = p(\boldsymbol{x} \mid c)/p(\boldsymbol{x} \mid \neg c)$ goes into infinity when substituting $p(\boldsymbol{x} \mid \neg c) := 0$.

called the *candidate list* in this paper, and insert the patterns found in the middle of the search into the list in the descending order of the relevance score. A pattern $\boldsymbol{x}$ cannot stay in the final top-k patterns if $\overline{\mathrm{F}}_c(\boldsymbol{x}) = 2p(\boldsymbol{x} \mid c)/(1 + p(\boldsymbol{x} \mid c)) < \mathrm{F}_c(\boldsymbol{z})$, where $\boldsymbol{z}$ is the k-th pattern. Here we rewrite this condition as: $p(\boldsymbol{x} \mid c) < \mathrm{F}_c(\boldsymbol{z})/(2 - \mathrm{F}_c(\boldsymbol{z})) = U_c(\boldsymbol{z})$. When the relevance score is dual-monotonic, this rewriting is always valid.[6] Here $U_c(\boldsymbol{z})$ works as a threshold for the support of $\boldsymbol{x}$ and leads to minimum support raising. That is, starting with a small value (e.g. $\sigma_{\min} := 1/|\mathcal{D}_c|$), the minimum support is repeatedly updated by $\sigma_{\min} := \max\{U_c(\boldsymbol{z}), \sigma_{\min}\}$ during the search. Thanks to the translation above, we can inherit fast frequent pattern mining algorithms like FP-Growth [8].

2.5 Search Strategies under Productivity

In this subsection, we describe search strategies for handling the productivity constraint in top-k branch-and-bound search. As stated before, top-k mining often uses a candidate list L that stores the current top-k candidate patterns. By the nature of minimum support raising, a new minimum support is set heavily depending on the k-th greatest score in L, and thus the effect of minimum support raising is rather limited if we relax the size limit of L and add the patterns that are not truly productive patterns. For example, consider the prefix enumeration tree in Fig. 1, where $\boldsymbol{x} = \{\mathsf{A}, \mathsf{C}, \mathsf{D}\}$ is visited before $\boldsymbol{x}' = \{\mathsf{A}, \mathsf{D}\}$ in a depth-first search. When visiting $\boldsymbol{x}$, it is inevitable to add $\boldsymbol{x}$ into L since it is uncertain whether $\boldsymbol{x}$ is weaker than $\boldsymbol{x}'$ at the moment. Contrastingly, we should keep the candidate list L as compact as possible, i.e. wherever possible we should filter out immediately the patterns that are not truly productive patterns.

Fortunately, if a pattern $\boldsymbol{x}$ is guaranteed to be visited after all sub-patterns of $\boldsymbol{x}$ have been visited, we can filter out $\boldsymbol{x}$ immediately if $\boldsymbol{x}$ is weaker than some existing pattern in the candidate list L. Typically breadth-first search, where shorter patterns are visited earlier, enables this filtering. Also for the same purpose, recent work by Grosskreutz et al. [6] proposes a memory-efficient method based on iterative deepening where 'depth' is the maximum size of patterns to be found. Furthermore, in this paper, following RP-Growth [10], we adopt memory-efficient depth-first traversal over a search space like suffix enumeration trees illustrated in the introduction. We take this strategy because the overhead of iterative deepening seems not ignorable in finding long patterns.

Furthermore, we also inherit aggressive pruning from RP-Growth. This pruning is based on an extended notion of weakness defined as follows:

Definition 3. *Let c be a class of interest, and $\boldsymbol{x}$, $\boldsymbol{x}'$ be a pair of patterns in $\mathcal{P}$. Then, $\boldsymbol{x}$ is prunably weaker than $\boldsymbol{x}'$ iff $\boldsymbol{x} \supset \boldsymbol{x}'$ and $\overline{R}_c(\boldsymbol{x}) \leq R_c(\boldsymbol{x}')$.* □

If a pattern $\boldsymbol{x}$ is prunably weaker than some pattern $\boldsymbol{x}'$ in the current candidate list, any super-pattern of $\boldsymbol{x}$ is also weaker than $\boldsymbol{x}'$, and thus we can safely prune the subtree rooted by $\boldsymbol{x}$ in finding top-k productive patterns.

[6] For many scores, it is possible to have $U_c(\boldsymbol{z})$ in closed form. A typical exception is the case with information gain, since it includes the entropy function in its definition. In such a case, the threshold in the right hand side should be numerically solved.

3 The Proposed Method

3.1 Suffix-Preserving Closure Extension

Based on the concepts and the techniques in Section 2, from now, we propose an efficient exact algorithm for top-k productive closed-on-the-positives. We have seen that, via minimum support raising, branch-and-bound pruning can be plugged into frequent closed pattern mining algorithms like LCM [19]. LCM is the first algorithm that introduces *prefix-preserving closure extension* (PPC extension, for short) for avoiding duplicate visits to a pattern. So we aim to run LCM over suffix enumeration trees, by introducing a mirrored version of PPC extension, called *suffix-preserving closure extension* (SPC extension).

Definition 4. *Let c be a class of interest, $\mathcal{D}_c$ the transactions in c, $\boldsymbol{x}$ a pattern in $\mathcal{P}$, x an item in $\mathcal{X}$ and $\Sigma_x(\boldsymbol{x}) = \boldsymbol{x} \cap \{x' \mid x \prec x'\}$. Then, the suffix-preserving closure extension of a pattern $\boldsymbol{x}$ by an item x is defined as $\boldsymbol{x}^* = \Gamma_c(\{x\} \cup \boldsymbol{x})$ such that (i) $x \notin \boldsymbol{x}$, (ii) $x \prec \mathrm{core}(\boldsymbol{x})$ and (iii) $\Sigma_x(\boldsymbol{x}^*) = \Sigma_x(\boldsymbol{x})$, where $\mathrm{core}(\boldsymbol{x})$ is the maximum item x w.r.t. $\prec$ such that $\mathcal{D}_c(\boldsymbol{x} \cap \{x' \mid x \preceq x'\}) = \mathcal{D}_c(\boldsymbol{x})$.* □

Here $\Sigma_x(\boldsymbol{x})$ is the suffix of $\boldsymbol{x}$ starting from the successor of x, and Condition (iii) says that such a suffix must be preserved between the original pattern and the extended one. To handle core$(\cdot)$ procedurally, a useful property is known [19]: *in an SPC extension $\boldsymbol{x}^* = \Gamma_c(\{x\} \cup \boldsymbol{x})$, $\mathrm{core}(\boldsymbol{x}^*)$ is exactly the added item x.* In this paper, for each pattern $\boldsymbol{x}$, we consider to record all of such added items in a chain of SPC extensions producing $\boldsymbol{x}$ and call them the *core items* in $\boldsymbol{x}$.

To illustrate, let us consider nine example transactions shown in Fig. 2 (top-left), each belongs to one of two classes $\{+, -\}$. Here we are interested in class $+$. Then, we have a total order $\mathsf{A} \prec \mathsf{B} \prec \mathsf{C} \prec \mathsf{E} \prec \mathsf{D}$ as a descending order of F-score in Fig. 2 (bottom-left). Fig. 2 (top-right) is an enumeration tree obtained by the exhaustive applications of SPC extension, where, at each branch, the core items to be added are chosen in the ascending order w.r.t. $\prec$. Such a tree is hereafter called an *SPC enumeration tree*. In Fig. 2, the core items are underlined, and among them, the doubly underlined ones are the core items which are added last. For example, given an empty pattern $\boldsymbol{x} = \emptyset$, we apply SPC extension by item C to the example transactions and obtain $\boldsymbol{x}^* = \Gamma_c(\{\mathsf{C}\} \cup \emptyset) = \{\mathsf{A}, \mathsf{C}\}$. In this case, C is a (the last) core item in $\{\mathsf{A}, \mathsf{C}\}$, while A is not.

3.2 Properties Related to SPC Extension

In this subsection, we show some key properties related to SPC extension in finding top-k productive closed-on-the-positives. Before that, we first obtain a tree like Fig. 2 (bottom-right) by extracting core items from an SPC enumeration tree, and call it a *core enumeration tree*. Let T and T_{core} be an SPC enumeration tree and its core enumeration tree, respectively. It is then easy to show that for a node (pattern) $\boldsymbol{x}$ in T_{core}, the corresponding node in T is its closure on the positives, i.e. $\Gamma_c(\boldsymbol{x})$. Also we see that T and T_{core} are isomorphic from the way of adding core items. For example, for the node $\{\mathsf{B}, \mathsf{D}\}$ in the core enumeration tree

Class	Transactions
+	{A, B, D, E}
+	{A, B, C, D, E}
+	{A, C, D, E}
+	{A, B, C}
+	{B}
−	{A, B, D, E}
−	{B, C, D}
−	{C, D, E}
−	{B, D, E}

Item x	$p(x \mid +)$	$p(x \mid -)$	$\mathrm{F}_+(x)$
A	0.8	0.25	0.780
B	0.8	0.75	0.627
C	0.6	0.50	0.571
D	0.6	1.00	0.462
E	0.6	0.75	0.511

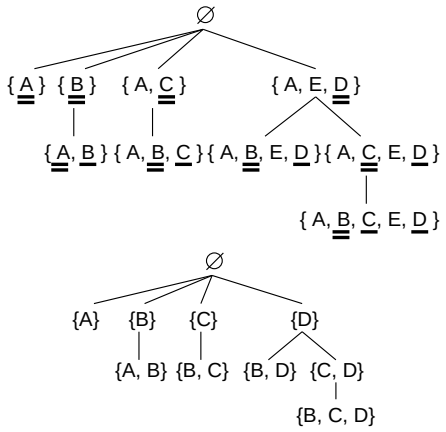

Fig. 2. Example transactions (top-left), F-scores of items (bottom-left), the enumeration tree with SPC extension (top-right) and its core enumeration tree (bottom-right).

in Fig. 2, we have $\{\mathsf{A}, \mathsf{B}, \mathsf{E}, \mathsf{D}\}$ in the SPC enumeration tree in Fig. 2. Besides, a core enumeration tree is a suffix enumeration tree with some subtrees being removed. Indeed, Fig. 2 (bottom-right) is the suffix enumeration tree in Fig. 1 with several leaves ($\{\mathsf{A}, \mathsf{C}\}$, $\{\mathsf{A}, \mathsf{D}\}$, etc.) being removed. This property comes from Condition (ii) of SPC extension (we are only allowed to add, as a new core item, a predecessor of the last core item) together with the fact that the core items to be added at each branch are chosen ascendingly w.r.t. $\prec$.

From the observations above, we will prove a couple of propositions that justify depth-first search with SPC extension (i.e. depth-first search over an SPC enumeration tree) in finding top-k productive closed-on-the-positives. We write $\boldsymbol{x} \sqsubset \boldsymbol{x}'$ iff $\boldsymbol{x}$ is visited before $\boldsymbol{x}'$ is visited, and note that for two distinct patterns in an SPC enumeration tree T, the corresponding patterns in T's core enumeration tree T_{core} are distinct, and vice versa (since T and T_{core} are isomorphic).

Proposition 1. *Let $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$ be two distinct patterns in an SPC enumeration tree T. Also let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively, be the corresponding patterns in T's core enumeration tree T_{core}. Then, if $\boldsymbol{x}_1 \subset \boldsymbol{x}_2$, we have $\boldsymbol{x}_1^* \subset \boldsymbol{x}_2^*$ and $\boldsymbol{x}_1^* \sqsubset \boldsymbol{x}_2^*$.* □

Proof. First, from $\boldsymbol{x}_1^* = \Gamma_c(\boldsymbol{x}_1)$ and $\boldsymbol{x}_2^* = \Gamma_c(\boldsymbol{x}_2)$, it is easy to see that $\boldsymbol{x}_1 \subset \boldsymbol{x}_2 \Rightarrow \boldsymbol{x}_1^* \subset \boldsymbol{x}_2^*$ since $\boldsymbol{x}_1 \subseteq \boldsymbol{x}_2 \Rightarrow \Gamma_c(\boldsymbol{x}_1) \subseteq \Gamma_c(\boldsymbol{x}_2)$ from the monotonicity of the closure operator [15]. We then have $\boldsymbol{x}_1 \subset \boldsymbol{x}_2 \Rightarrow \boldsymbol{x}_1 \sqsubset \boldsymbol{x}_2$ since any core enumeration tree is a part of a suffix enumeration tree. Also $\boldsymbol{x}_1^* \sqsubset \boldsymbol{x}_2^* \Leftrightarrow \boldsymbol{x}_1 \sqsubset \boldsymbol{x}_2$ holds since T and T_{core} are isomorphic. It immediately follows that $\boldsymbol{x}_1 \subset \boldsymbol{x}_2 \Rightarrow \boldsymbol{x}_1^* \sqsubset \boldsymbol{x}_2^*$. □

Proposition 2. *Let $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$ be two distinct patterns in an SPC enumeration tree T. Also let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively, be the corresponding patterns in T's core enumeration tree T_{core}. Also suppose that $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are not subsets of each other. Then, $\boldsymbol{x}_1^* \subset \boldsymbol{x}_2^* \Rightarrow \boldsymbol{x}_1^* \sqsubset \boldsymbol{x}_2^*$.* □

Proof. We prove this by contraposition. First assume that $\boldsymbol{x}_1^* \supset \boldsymbol{x}_2^*$ and equivalently $\boldsymbol{x}_1 \supset \boldsymbol{x}_2$. Also consider the common suffix $\boldsymbol{y}$ of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Furthermore, let x (resp. x') be the maximum item w.r.t. $\prec$ in $\boldsymbol{x}_1 \setminus \boldsymbol{y}$ (resp. $\boldsymbol{x}_2 \setminus \boldsymbol{y}$). Then, we necessarily have $x' \prec x$ since $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are not subsets of each other, and $\boldsymbol{x}_2 \sqsubset \boldsymbol{x}_1$. From Condition (iii) of SPC extension, $\Gamma_c(\{x'\} \cup \boldsymbol{y})$ never contains x, and accordingly neither does $\boldsymbol{x}_2^*$ $(= \Gamma_c(\boldsymbol{x}_2))$. On the contrary, $\boldsymbol{x}_1^*$ $(= \Gamma_c(\boldsymbol{x}_1))$ always contains x since x is its core item. Therefore, $\boldsymbol{x}_1^*$ can never be a subset of $\boldsymbol{x}_2^*$. Now we have $\boldsymbol{x}_1^* \subset \boldsymbol{x}_2^* \Rightarrow \boldsymbol{x}_1^* \sqsubset \boldsymbol{x}_2^*$. □

Proposition 3. *When a pattern $\boldsymbol{x}$ is visited in a depth-first search with SPC extension, all of $\boldsymbol{x}$'s sub-patterns have already been visited.* □

Proof. It is sufficient to show that $\boldsymbol{x}_1^* \subset \boldsymbol{x}_2^* \Rightarrow \boldsymbol{x}_1^* \sqsubset \boldsymbol{x}_2^*$, where $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$ be two distinct patterns in an SPC enumeration tree, say T. Here we first assume that $\boldsymbol{x}_1^* \subset \boldsymbol{x}_2^*$. Also let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively, be the corresponding patterns in T's core enumeration tree T_{core}. Then, we consider three exhaustive cases: (1) $\boldsymbol{x}_1 \supset \boldsymbol{x}_2$, (2) $\boldsymbol{x}_1 \subset \boldsymbol{x}_2$, and (3) $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are not subsets of each other. The first case is incompatible with the assumption $\boldsymbol{x}_1^* \subset \boldsymbol{x}_2^*$ since $\boldsymbol{x}_2 \subset \boldsymbol{x}_1 \Rightarrow \boldsymbol{x}_2^* \subset \boldsymbol{x}_1^*$ from Proposition 1. In the second case, we have $\boldsymbol{x}_1^* \sqsubset \boldsymbol{x}_2^*$ again from Proposition 1. In the last case, we also have $\boldsymbol{x}_1^* \sqsubset \boldsymbol{x}_2^*$ from Proposition 2. □

Now, as described in Section 2.5, we can filter out a new pattern $\boldsymbol{x}$ immediately if $\boldsymbol{x}$ is weaker than some existing pattern in the candidate list. So the size of the list is kept being k except when there are ties at the bottom.[7]

3.3 Algorithm Description

In this subsection, we present the proposed algorithm for finding top-k productive closed-on-the-positives. First, we introduce four global variables c, k, $\sigma_{\min}$ and L, which stand for the class of interest, the number of patterns to be output, the minimum support and the candidate list, respectively. The values of c and k are given by the user. On the other hand, $\sigma_{\min}$ and L are respectively initialized as $1/|\mathcal{D}_c|$ and $\emptyset$. The total order $\prec$ among items is considered as a descending order of the relevance score R_c in use.[8] The central procedure is Grow, shown in Algorithm 1, and we call Grow($\emptyset$) to run the algorithm. After termination, the final top-k patterns are stored in the candidate list L.

Given the current pattern $\boldsymbol{x}$, Grow($\boldsymbol{x}$) works as follows. First, we compute the set B of items that satisfy Conditions (i) and (ii) of SPC extension (Line 1). Note that B contains all possible items if $\boldsymbol{x} = \emptyset$. Then, we try to branch by each item x in B ascendingly w.r.t. $\prec$ (Line 2). We create a new pattern $\boldsymbol{x}^*$ (Line 4) with pruning by the minimum support (Line 3) and by Condition (iii) of SPC extension (Line 5). After computing the relevance score of $\boldsymbol{x}^*$ (Line 6), we add

[7] Of course, such ties do not affect the minimum support threshold to be raised.

[8] This total ordering has a preferable side-effect that, as is seen in Fig. 1 (right) and Fig. 2 (top-right), we try the combinations of promising items earlier in a suffix/SPC enumeration tree and hence the minimum support tends to be raised quickly.

Algorithm 1 GROW($\boldsymbol{x}$)

Require: $\boldsymbol{x}$: the current pattern
1: $B := \{x \mid x \notin \boldsymbol{x}$ and x is a predecessor of the last core item added into $\boldsymbol{x}\}$
2: **for each** $x \in B$ enumerated in the ascending order of $\prec$ **do**
3: **if** $p(\{x\} \cup \boldsymbol{x} \mid c) < \sigma_{\min}$ **then continue**
4: $\boldsymbol{x}^* := \Gamma_c(\{x\} \cup \boldsymbol{x})$
5: **if** $\Sigma_x(\boldsymbol{x}^*) \neq \Sigma_x(\boldsymbol{x})$ **then continue**
6: Compute $R_c(\boldsymbol{x}^*)$
7: **if** $\boldsymbol{x}^*$ is not weaker than any patterns in L **then**
8: Insert $\boldsymbol{x}^*$ into L following the descending order of R_c
9: **if** $|L| \geq k$ **then**
10: Remove the patterns with the score below the k-th pattern's score from L
11: $\sigma_{\min} := \max\{U_c(\boldsymbol{z}), \sigma_{\min}\}$, where $\boldsymbol{z}$ is the k-th pattern in L
12: **end if**
13: **end if**
14: Call GROW($\boldsymbol{x}^*$) **if** $\boldsymbol{x}$ is not prunably weaker than any patterns in L
15: **end for**

$\boldsymbol{x}^*$ into the candidate list L if $\boldsymbol{x}^*$ is not weaker than any existing pattern in L (Lines 7–13). If L is full, we replace with $\boldsymbol{x}^*$ the patterns that are less relevant than $\boldsymbol{x}^*$ (Line 10) and update the minimum support (Line 11). Here $U_c(\boldsymbol{z})$ is a new threshold based on $\boldsymbol{z}$'s score (Section 2.4). Finally, for $\boldsymbol{x}^*$ having passed the filter on prunable weakness, we call GROW recursively (line 14).

4 Experimental results

We conducted an experiment to confirm the efficiency of the proposed algorithm. The datasets are originally collected in UCI Machine Learning Repository and we used a preprocessed version available from `http://dtai.cs.kuleuven.be/CP4IM/datasets/`. The statistics are summarized in Table 1.

In the experiment, we compare the search strategies described in Section 2.5, i.e. breadth-first search (BFS), depth-first search over a prefix/suffix enumeration tree (DFS(Prefix)/DFS(Suffix)) and iterative deepening (ID) recently proposed by Grosskreutz et al. [6].[9] We specified $k = 10$ as the number of productive closed-on-the-positives to be found. F-score is used as the relevance score. The result is shown in Fig. 3. The y-axis indicates the number of visited nodes, which equals the number of times the closure operator was applied. The measurements at the y-axis are presented in logarithmic scale. From Fig. 3, we first see that DFS(Suffix) (the proposed method) outperforms DFS(Prefix) for most of datasets. In particular, DFS(Prefix) works poorly for german-credit. In addition, as stated

[9] We also implemented an optimization described in Section 5.1 of [6], where we skip the depths for which no patterns exist by keeping track the length of the shortest pattern exceeding the current depth at each iteration. All implementations are written in Java, and for comparison, we did not use elaborate data structures like FP-trees but pseudo-projection-style databases [16].

Table 1. Statistics on the datasets. "#Trs." indicates the number of transactions.

Dataset	#Trs.	#Items	Dataset	#Trs.	#Items	Dataset	#Trs.	#Items
anneal	812	93	hypothyroid	3,247	88	splice-1	3,190	287
audiology	216	148	kr-vs-kp	3,196	73	tic-tac-toe	958	27
australian-credit	653	125	lymph	148	68	vote	435	48
german-credit	1,000	112	mushroom	8,124	119	zoo-1	101	36
heart-cleveland	296	95	primary-tumor	336	31			
hepatitis	137	68	soybean	630	50			

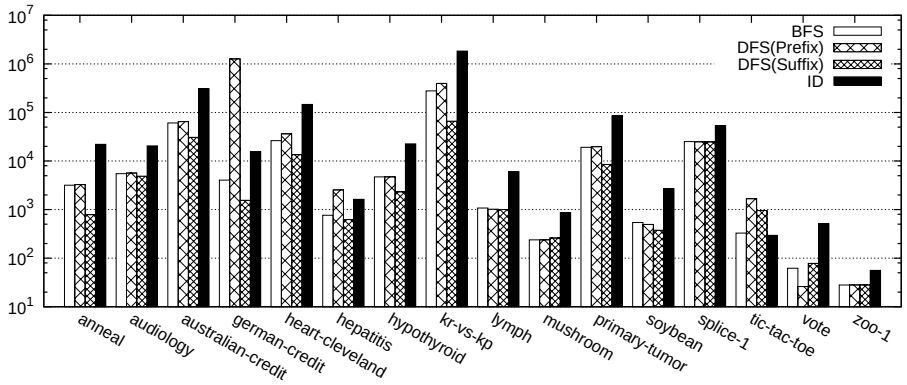

Fig. 3. The result of a comparison among search strategies.

before, the overhead of iterative deepening is not ignorable in this experiment. On comparison between BFS and DFS(Suffix), we can say that for some small datasets like tic-tac-toe, BFS works better than DFS(Suffix), but for the datasets where the search is costly, DFS(Suffix) outperforms BFS (remind here that the y-axis is in logarithmic scale). Totally, the proposed method DFS(Suffix) stably runs fast in comparison with the other search strategies.

5 Conclusion

In this paper, we proposed an efficient exact algorithm for finding top-k discriminative patterns that are not redundant and would be of value at a later step in prediction or knowledge discovery. Redundancy among discriminative patterns are eliminated by two set-inclusion-based constraints, closedness and productivity. Such constraints are efficiently tested with suffix-preserving closure extension under dual-monotonic relevance scores. We showed formally and empirically that the proposed algorithm successfully keeps compact the list of candidate top-k patterns during the search and consequently high the minimum support threshold. In future, we would like to extend the proposed algorithm for more complex data such as sequences.

Acknowledgments. This work was supported in part by JSPS KAKENHI Grant Number 24700141.

References

1. Bay, S.D., Pazzani, M.J.: Detecting group differences: mining contrast sets. Data Mining and Knowledge Discovery 5, 213–246 (2001)
2. Bayardo, R., Agrawal, R., Gunopulos, D.: Constraint-based rule mining in large, dense databases. Data Mining and Knowledge Discovery 4, 217–240 (2000)
3. Dong, G., Li, J.: Efficient mining of emerging patterns: discovering trends and differences. In: Proc. of KDD-99. pp. 43–52 (1999)
4. Fürnkranz, J., Gamberger, D., Lavrač, N.: Foundations of Rule Learning. Springer (2012)
5. Garriga, G.C., Kralj, P., Lavrač, N.: Closed sets for labeled data. J. of Machine Learning Research 9, 559–580 (2008)
6. Grosskreutz, H., Paurat, D.: Fast and memory-efficient discovery of the top-k relevant subgroups in a reduced candidate space. In: Proc. of ECML/PKDD-11. pp. 533–548 (2011)
7. Grosskreutz, H., Rüping, S., Wrobel, S.: Tight optimistic estimates for fast subgroup discovery. In: Proc. of ECLM/PKDD-08. pp. 440–456 (2008)
8. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proc. of SIGMOD-00. pp. 1–12 (2000)
9. Han, J., Wang, J., Lu, Y., Tzvetkov, P.: Mining top-K frequent closed patterns without minimum support. In: Proc. of ICDM-02. pp. 211–218 (2002)
10. Kameya, Y., Sato, T.: RP-growth: top-k mining of relevant patterns with minimum support raising. In: Proc. of SDM-12. pp. 816–827 (2012)
11. Kralj Novak, P., Lavrač, N., Webb, G.I.: Supervised descriptive rule discovery: a unifying survey of contrast set, emerging pattern and subgroup mining. J. of Machine Learning Research 10, 377–403 (2009)
12. Lavrač, N., Gamberger, D., Jovanoski, V.: A study of relevance for learning in deductive databases. J. of Logic Programming 40, 215–249 (1999)
13. Morishita, S., Sese, J.: Traversing itemset lattices with statistical metric pruning. In: Proc. of PODS-00. pp. 226–236 (2000)
14. Nijssen, S., Guns, T., De Raedt, L.: Correlated itemset mining in ROC space: a constraint programming approach. In: Proc. of KDD-09. pp. 647–656 (2009)
15. Pasquier, N., Bastide, Y., Taouli, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Proc. of ICDT-99. pp. 398–416 (1999)
16. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proc. of ICDE-01. pp. 215–224 (2001)
17. Piatetsky-Shapiro, G.: Discovery, analysis, and presentation of strong rules. In: Knowledge Discovery in Databases, pp. 229–248. AAAI Press (1991)
18. Soulet, A., Crémilleux, B., Rioult, F.: Condensed representation of emerging patterns. In: Proc. of PAKDD-04. pp. 127–132 (2004)
19. Uno, T., Asai, T., Uchida, Y., Arimura, H.: An efficient algorithm for enumerating closed patterns in transaction databases. In: Proc. of DS-04. pp. 16–31 (2004)
20. Webb, G.I.: Discovering significant patterns. Machine Learning 68, 1–33 (2007)
21. Wrobel, S.: An algorithm for multi-relational discovery of subgroups. In: Proc. of PKDD-97. pp. 78–87 (1997)
22. Yuan, C., Lim, H., Lu, T.C.: Most relevant explanation in Bayesian networks. J. of Artificial Intelligence Research 42, 309–352 (2011)
23. Zimmermann, A., De Raedt, L.: Cluster grouping: from subgroup discovery to clustering. Machine Learning 77, 125–159 (2009)