Fast and Memory-Efficient Discovery of the Top-k Relevant Subgroups in a Reduced Candidate Space

Henrik Grosskreutz and Daniel Paurat

Fraunhofer IAIS, Schloss Birlinghoven, 53754 St. Augustin, Germany
henrik.grosskreutz@iais.fraunhofer.de,
daniel.paurat@iais-extern.fraunhofer.de

Abstract. We consider a modified version of the top-k subgroup discovery task, where subgroups dominated by other subgroups are discarded. The advantage of this modified task, known as relevant subgroup discovery, is that it avoids redundancy in the outcome. Although it has been applied in many applications, so far no efficient exact algorithm for this task has been proposed. Most existing solutions do not guarantee the exact solution (as a result of the use of non-admissible heuristics), while the only exact solution relies on the explicit storage of the whole search space, which results in prohibitively large memory requirements.

In this paper, we present a new top-k relevant subgroup discovery algorithm which overcomes these shortcomings. Our solution is based on the fact that if an iterative deepening approach is applied, the relevance check – which is the root of the problems of all other approaches – can be realized based solely on the best k subgroups visited so far. The approach also allows for the integration of admissible pruning techniques like optimistic estimate pruning. The result is a fast, memory-efficient algorithm which clearly outperforms existing top-k relevant subgroup discovery approaches. Moreover, we analytically and empirically show that it is competitive with simpler approaches which do not consider the relevance criterion.

1 Introduction

In applications of local pattern discovery tasks, one is typically interested in obtaining a small yet meaningful set of patterns. The reason is that resources for post-processing of the patterns are typically limited, both if the patterns are manually reviewed by human experts, or if they are used as input of a subsequent data-mining step, following a multi-step approach like LeGo [11].

Reducing the number of raw patterns to a subset of manageable size can be done using different approaches: one is to use a *quality function* to assess the value of the patterns, and to discard all but the k highest-quality patterns. This is known as the "top-k" approach. It can be further subdivided depending on the pattern type and the quality function considered. In this paper, we consider the case where the data has a binary label, the quality function accounts for

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part I, LNAI 6911, pp. 533–548, 2011.

the support in the different classes, and the patterns have the form of itemsets. This setting is known as top-k *supervised descriptive rule discovery*, *correlated pattern mining* or *subgroup discovery* [5]. In the following, we will stick with the expression *subgroup discovery* and with the terminology used in this community.

Restricting to the top-k patterns (or subgroups, in our specific case) is not the only approach to reduce the size of the output. A different line of research aims at the identification and removal of patterns which are of little interest compared to other patterns, (cf. [6,4]). This idea is formalized using constraints based on the interrelation between patterns. A particularly appealing approach along this line is the theory of relevance [14,7]. The idea of this approach, which applies only to binary labeled data, is to remove all patterns that are *dominated* (or *covered*) by another pattern. Here, a pattern is considered as dominating another pattern if the dominating pattern covers at least all *positives* (i.e. target-class individuals) covered by the dominated pattern, but no additional *negative*.

The theory of relevance not only allows to get rid of multiple equivalent descriptions (as does the theory of closed sets), but also of trivial specializations which provide no additional insight over their generalizations. Due to this advantage, relevance has been used as a filtering criterion in several subgroup discovery applications [14,12,2]. In many settings, however, the number of *relevant* subgroups is still far larger than desired. In this case, a nearby solution is to combine it with the top-k approach.

Up to now, however, no satisfying solution has been developed for the task of *top-k relevant subgroup discovery*. Most algorithms apply non-admissible pruning heuristics, with the result that high-quality subgroups can be overlooked (e.g. [14,16]). The source of these problems is that relevance is a property not defined locally, but with respect to the set of *all* other subgroups. The only non-trivial algorithm which provably finds the exact solution to this task is that of Garriga et al. [7]. This approach is based on the insight that all relevant subgroups must be *closed on the positives*; moreover, the relevance of a closed-on-the-positive can be determined based solely on the information about all closed-on-the-positives. This gives rise to an algorithmic approach which exhaustively traverses all closed-on-the-positives, *stores them*, and relies on this collection to distinguish the relevant subgroups from the irrelevant closed-on-the-positives. Obviously, this approach suffers from the drawback that a potentially very large number of subgroups has to be *stored in memory*. For complex datasets, the high memory requirements are a much more severe problem than the runtime. Another drawback of this approach is that it does not allow for admissible pruning techniques based on a dynamically increasing threshold [22,17,8].

In this paper, we make the following contributions:

- We analyze existing top-k relevant subgroup discovery algorithms and show how all except for the memory-demanding approach of [7] fail to guarantee an exact solution;
- We present a simple solution to the relevance check which requires an amount of memory only *linear* in k and the number of features. An additional advantage of this approach is that it can easily be combined with admissible pruning techniques;

– Thereupon we present a new, memory-efficient top-k relevant subgroup discovery algorithm. We demonstrate that it is faster than the only existing exact algorithm (Garriga et al. [7]), while avoiding the memory issues of the latter. Moreover, we show that our approach is competitive with *all* existing exhaustive subgroup discovery approaches, in particular with simpler algorithms which do not consider the relevance criterion.

The remainder of the paper is structured as follows: after reviewing basic definitions in Section 2, we illustrate the task of relevant subgroup discovery in Section 3. Successively, we present our new approach and analyze its properties in Section 4, before we present empirical results in Section 5.

2 Preliminaries

In this section, we will define the task of subgroup discovery, review the theory of relevance and discuss its connection to closure operators.

2.1 Subgroup Discovery

Subgroup discovery [10] aims at discovering descriptions of interesting subportions of a dataset. We assume all records $d_1, \ldots, d_m$ of the dataset to be described by a set of n binary features $(f_1(d_i), \ldots, f_n(d_i)) \in \{0,1\}^n$. A *subgroup description* sd is a subset of the feature set, i.e. $sd \subseteq \{f_1, \ldots, f_n\}$. In the following, we will sometimes simply write *subgroup* to refer to a subgroup description. A data record d satisfies sd if $f(d) = 1$ for all $f \in sd$, that is, subgroup descriptions are interpreted conjunctively. Thus, we sometimes use the notation $f_{i_1}\&, \ldots \& f_{i_k}$ instead of $\{f_{i_1}, \ldots, f_{i_k}\}$. Finally, $DB[sd]$ denotes the set of records $d \in DB$ of a database DB satisfying a subgroup description sd.

The interestingness of a subgroup description sd in the context of a database DB is measured by a *quality function* q that assigns a real-valued quality $q(sd, DB)$ to sd. The quality functions usually combine the size of the subgroup and its unusualness with respect to a designated target variable, the *class* or *label*. In this paper, we only consider the case of *binary* labels, that is, the label of a record d is a special feature $class(d)$ with range $\{+, -\}$. Some of the most common quality functions for binary labeled data are of the form:

$$|DB[sd]|^a \cdot \left(\frac{|TP(DB, sd)|}{|DB[sd]|} - \frac{|TP(DB, \emptyset)|}{|DB|} \right) \tag{1}$$

where $TP(DB, sd) := \{d \in DB[sd] \mid class(d) = +\}$ denotes the true positives of the subgroup sd, a is a constant such that $0 \leq a \leq 1$, and $TP(DB, \emptyset)$ simply denotes *all* positives in the dataset. The family of quality functions characterized by Equation 1 includes some of the most popular quality functions: for $a = 1$, it is order equivalent to the Piatetsky-Shapiro quality function [10] and the weighted relative accuracy WRACC [13], while for $a = 0.5$ it corresponds to the binomial test quality function [10].

2.2 Optimistic Estimate Pruning

A concept closely related to quality functions is that of an *optimistic estimate* [22]. This is a function that provides a bound on the quality of a subgroup description *and of all its specializations*. Formally, an optimistic estimator for a quality function q is a function oe mapping a database DB and a subgroup description sd to a real value such that for all DB, sd and *specializations* $sd' \supseteq sd$, it holds that $oe(DB, sd) \geq q(DB, sd')$. Optimistic estimates allow to drastically improve the performance of subgroup discovery by means of *pruning* [17,8].

2.3 The Theory of Relevance

The theory of relevance [15,14] is aimed at eliminating irrelevant patterns, resp. subgroups. A subgroup sd_{irr} is considered as irrelevant if it is *dominated* (or *covered*) by another subgroup sd in the following sense:

Definition 1. *The subgroup sd_{irr} is* dominated *by the subgroup sd in database DB iff. (i) $TP(DB, sd_{irr}) \subseteq TP(DB, sd)$ and (ii) $FP(DB, sd) \subseteq FP(DB, sd_{irr})$.*

Here, TP is defined as in 2.1, while $FP(DB, sd) = \{c \in DB[sd] \mid class(c) = -\}$ denotes the false positives.

2.4 Closure Operators and Their Connection to Relevance

As shown by Garriga et al. [7], the notion of relevance can be restated in terms of the following mapping between subgroup descriptions:

$$\Gamma^+(X) := \{f \mid \forall d \in TP(DB, X) : f[d] = 1\}. \tag{2}$$

Γ^+ is a *closure operator*, i.a. a function defined on the power-set of features $\mathcal{P}(\{f_1, \ldots, f_n\})$ such that for all $X, Y \in \mathcal{P}(\{f_1, \ldots, f_n\})$, (i) $X \subseteq \Gamma(X)$ (extensivity), (ii) $X \subseteq Y \Rightarrow \Gamma(X) \subseteq \Gamma(Y)$ (monotonicity), and (iii) $\Gamma(X) = \Gamma(\Gamma(X))$ (idempotence) holds. The fixpoints of Γ^+, i.e. the subgroup descriptions sd_{rel} such that $sd_{rel} = \Gamma^+(sd_{rel})$, are precisely the *closed-on-the-positives* mentioned earlier. The main result in [7] is that

Proposition 1. *The space of relevant patterns consists of all patterns sd_{rel} satisfying the following: (i) sd_{rel} is closed on the positives, and (ii) there is no generalization $sd \subsetneq sd_{rel}$ closed on the positives such that $|FP(sd)| = |FP(sd_{rel})|$.*

The connection between relevancy and closure operators is particularly interesting because closure operators have extensively been studied in the area of closed pattern mining (cf. [19]). However, unlike here in closed pattern mining the closure operator is defined solely on the *support*, without accounting for labels. The fixpoints of the closure operator based on the support are called *closed* patterns. Many algorithms have been developed to traverse the fixpoints of some *arbitrary* closure operator, e.g. LCM [21]. Making use of this fact, Garriga et al. [7] have proposed a simple two-step approach to find the relevant patterns: first, find and store all closed-on-the-positives; second, remove all dominated closed-on-the-positives using Proposition 1. We will refer to this approach as CPosSd.

3 Relevant Subgroup Discovery

As motivated in the introduction, we are interested in the following task:

Task 1. ***Top-k Relevant Subgroup Discovery*** *Given a database DB, a quality function q, and an integer $k > 0$, find a set of subgroup descriptions R of size k, such that*

- *all subgroup descriptions in R are relevant wrt. DB, and*
- *all subgroup descriptions not in R either have a quality no higher than $\min_{sd \in R} q(DB, sd)$, or are dominated by some subgroup description in R.*

We will now illustrate this task using a simple example, before we show how existing pruning approaches result in incorrect results.

3.1 An Illustrative Example

Our example database, shown in Table 1, describes opinion polls. There is one record for every participating person. Beside the class, *Approval*, there are four features that characterize the records: *Children:yes* and *Children:no* indicates whether or not the participant has children; *University* indicates that the participant has a university degree, and finally *High Income* indicates an above-average income. To keep the example simple, we have not included features describing the negative counterparts of the last two features.

This simple example dataset induces a lattice of candidate subgroup descriptions, containing a total of 16 nodes. These include several redundant descriptions (like *University* and *HighIncome & University*), which are avoided if we consider the sub-space of *closed* subgroups. Figure 1 visualizes this space, thereby representing both the type and the WRACC quality:

- *type:* The visualization distinguishes between subgroup descriptions which are *closed*, *closed on the positives* and *relevant*. Every node represents a *closed* subgroup; those *closed on the positives* are rendered with a double border; finally, *relevant* subgroups are rendered using a rectangular shape.
- *quality:* The color intensity of a node corresponds to the quality of the subgroup: higher-qualities correspond to more intense gray shades.

Table 1. Example: An simple opinion poll dataset

Approval	Children=yes	Children=no	University	High Income
+		✓	✓	✓
+	✓		✓	✓
+	✓			
-		✓		✓
-	✓			
-	✓			
-		✓		

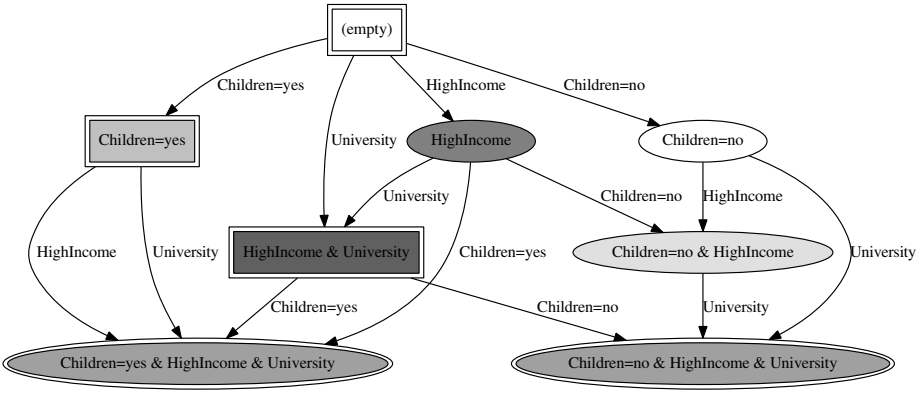


Fig. 1. Subgroup lattice for the example. Relevant subgroups are highlighted

The figure illustrates that high-quality subgroups need not be relevant: For example, the two subgroups *Children:yes & HighIncome & University* and *Children:no & HighIncome & University* have high quality but are irrelevant, as they are merely a fragmentation of the relevant subgroup *HighIncome & University*.

3.2 Existing Approaches, Challenges and Pitfalls

The existing approaches can briefly be divided into two classes: extensions of classical algorithms which apply some kind of pruning, and the closed-on-the-positives approach of Garriga et al. [7].

Pruning-based Approaches. State-of-the-art top-k subgroup discovery algorithm do not traverse the whole space of candidate patterns but apply pruning to reduce the number of patterns effectively visited (cf. [8,3,18]). The use of such techniques results in a dramatic reduction of the execution time and is an indispensable tool for fast exhaustive subgroup discovery [17,8,3].

To better understand the issues that can arise, we first briefly review the concept of a *dynamically increasing quality threshold* used during optimistic estimate pruning. Recall that classical subgroup discovery algorithms traverse the space of candidate subgroup descriptions, collecting the best subgroups. Once at least k subgroups have been collected, only subgroups are of interest whose quality *exceeds* that of the k-th best subgroup visited so far. The quality of the k-th subgroup thus represents a threshold, which *increases monotonically* in a *dynamic* fashion during traversal of the search space. Combined with an optimistic estimator (see Section 2.2), this dynamic threshold can allow pruning large parts of the search space.

The use of a dynamic threshold is of key importance in optimistic estimate pruning, as it is impossible to calculate a suitable threshold beforehand. Figure 1

illustrates the efficiency of this approach: if top-1 subgroup discovery is performed in this example, then the dynamic threshold allows pruning *all* but the direct children of the root node. Unfortunately, storing only the k best subgroups visited and using the quality of the k-best subgroup as dynamic threshold is problematic in *relevant* subgroup discovery:

1. If a relevant subgroup is visited which dominates more than one subgroup so far collected, then all dominated subgroups have to be removed. This can have the effect that when the computation ends, the result queue erroneously contains less than k subgroups.
2. If the quality threshold is increased to the quality of the k-th subgroup in the result queue, but the queue contains non-relevant subgroups, then some relevant subgroups can erroneously be pruned.

The two above problems can be observed in our example scenario. Issue 1 arises if we search for the top-2 subgroups in Figure 1, and the nodes are visited in the following order: *Children=yes*, *Children:yes & HighIncome & University*, *Children:no & HighIncome & University* and *High Income & University*. When the computation ends, the result will only contain *High Income & University*, but miss the second relevant subgroup, *Children=yes*. Issue 2 arises if *Children:yes & HighIncome & University* and *Children:no & HighIncome & University* are added to the queue before *Children=yes* is visited: the effect is that the minimum quality threshold is increased to a level that will incorrectly prune *Children=yes*.

The above issues are the reason why most existing algorithms, like BSD [16], do not guarantee an exact solution for the task of top-k relevant subgroup discovery. This is problematic both because the outcome will typically be of less value, and because it is not uniquely determined; in fact, the outcome can differ among implementations and possibly even among execution traces. This effect is amplified if a beam search is applied (e.g. [14]), where the subgroups considered are not guaranteed not to be dominated by some subgroup outside the beam.

Approaches based on the closed-on-the-positives. The paper of Garriga et. al. [7] is the first that proposes a non-trivial approach to *correctly* solve the relevant subgroup discovery task. The authors investigate the relation between closure operators (cf. [19]) and relevance, and show that the relevant subgroups are a subset of the subgroups closed on the positives. While the focus of the paper is on structural properties and not on computational aspects, the authors also propose the simple two-step algorithm CPosSd described in Section 2.4. The search space considered by this algorithm — the closed-on-the-positives — is a subset of the closed subgroups, thus it operates on a smaller candidate space than all earlier approaches. The downside is that it does not account for optimistic estimate pruning, and, probably more seriously, that it has very high memory requirements, as the whole set of closed-on-the-positives has to be stored.

4 An Iterative Deepening Approach

We aim at a solution that:

1. Avoids the high memory requirements of CPosSd [7];
2. Considers a reduced search space, namely the closed on the positives;
3. Applies pruning based on a dynamically increasing quality threshold.

The last bullet implies that we need a way to efficiently determine whether a subgroup is relevant or not *the moment it is visited*. Proposition 1 tells us that this can be done if a pattern is guaranteed to be visited only once each of its generalizations have been visited. One straightforward solution is thus to traverse the candidate space in a general-to-specific way. While this traversal strategy slightly differs from a breadth-first-traversal, it has the same drawbacks, in particular that the memory requirements can be linear in the size of the candidate space. In the worst case, this results in memory requirements exponential in the number of features, which is clearly problematic.

To avoid the above memory issue, we build our solution upon an *iterative deepening* depth-first traversal of the space of closed-on-the-positives. Iterative deepening depth-first search is well known to have more convenient memory requirements [20], and moreover it ensures that whenever a subgroup description is visited, all its generalizations have been visited before.

4.1 A Relevance Check Based On The Top-k Subgroups Visited

One challenge remains, namely devising a memory-efficient way to test the relevance of a newly visited pattern. Obviously, we cannot store all generalizations of every subgroup in memory (and simply apply Proposition 1): as every pattern can have exponentially many generalizations, this would again result in memory issues. Instead, our solution is based on the following observation:

Proposition 2. *Let DB be a dataset, q a quality function of the form of Equation 1 (with* $0 \leq a \leq 1$*) and minQ some real value. Then, the relevance of any closed-on-the-positive sd with quality* $\geq minQ$ *can be computed from the set*

$$G^* = \{sd_{gen} \subsetneq sd \,|\, sd_{gen} \text{ is relevant in } DB \text{ and } q(DB, sd_{gen}) > minQ\}$$

of all generalizations of sd with quality $\geq minQ$*. In particular, sd is irrelevant if and only if there is a relevant subgroup* sd_{gen} *in* G^* *with same* negative support*, where the negative support of a pattern sd is defined as* $|FP(DB, sd)|$*.*

The above proposition tell us that we can perform the relevance check based *only* on the top-k relevant subgroups visited so far: The iterative deepening traversal ensures that a pattern *sd* is only visited once all generalizations have been visited; so if the quality of the newly visited pattern *sd* exceeds that of the k-best subgroup visited so far, then the set of the best k relevant subgroups visited includes all generalizations of *sd* with higher quality – that is, a superset of the set G^* mentioned in Proposition 2; hence, we can check the relevance

of *sd*. On the other hand, if the quality of *sd* is lower than that of the k-best subgroup visited, then we don't care about its relevance anyways.

To prove the correctness of Proposition 2, we first present two lemmas:

Lemma 3. *If a closed-on-the-positive sd_{irr} is irrelevant, i.e. if there is a generalization $sd \subsetneq sd_{irr}$ closed on the positives with the same negative support as sd_{irr}, then there is also at least one* relevant *generalization $sd_{rel} \subsetneq sd_{irr}$ with the same negative support.*

Proof. Let N be the set of all closed-on-the-positives generalizations of sd_{irr} with the same negative support as *sd*. There must be at least one sd_{rel} in N such that none of the patterns in N is a generalization of sd_{rel}. From Proposition 1, we can conclude that sd_{rel} must be relevant and dominates sd_{irr}. □

Lemma 4. *If a pattern sd_{rel} dominates another pattern sd_{irr}, then sd_{rel} has higher quality than sd_{irr}.*

Proof. We have that $|DB[sd_{rel}]| \geq |DB[sd_{irr}]|$, because sd_{rel} is a subset of sd_{irr} and support is antimonotonic. Thus, to show that sd_{rel} has higher quality, it is sufficient to show $|TP(DB, sd_{rel})|\,/|DB[sd_{rel}]| > |TP(DB, sd_{irr})|\,/|DB[sd_{irr}]|$. From Proposition 1, we can conclude that sd_{rel} and sd_{irr} have the same number of false positives; let F denote this number. Using F, we can restate the above inequality as $|TP(DB, sd_{rel})|\,/(|TP(DB, sd_{rel})| + F) > |TP(DB, sd_{irr})|\,/(|TP(DB, sd_{irr})| + F)$. All that remains to show is thus that $|TP(DB, sd_{rel})| > |TP(DB, sd_{irr})|$. By definition of relevance, $|TP(DB, sd_{rel})| \geq |TP(DB, sd_{irr})|$, and because sd_{rel} and sd_{irr} are different and closed on the positives, the inequality must be strict, which completes the proof. □

Based upon these lemmas, it is straightforward to prove Proposition 2:

Proof. We first show that if *sd* is irrelevant, then there is a generalization in G^* with the same negative support. From Lemma 3 we know that if *sd* is irrelevant, then there is at least one *relevant* generalization of *sd* with same negative support dominating *sd*. Let sd_{gen} be such a generalization. Lemma 4 implies that $q(DB, sd_{gen}) \geq q(DB, sd) \geq minQ$, hence sd_{gen} is a member of the set G^*.

It remains to show that if *sd* is relevant, then there is no generalization in G^* with same negative support. This follows directly from Proposition 1. □

4.2 The Algorithm

Algorithm 1 shows the pseudo-code for the approach outlined above. The main program is responsible for the iterative deepening. The actual work is done in the procedure `findSubgroupsWithDepthLimit`, which traverses the space of closed-on-the-positives in a depth-first fashion using a *stack* (aka LIFO data structure). Thereby, it ignores (closed-on-the-positive) subgroups longer than the length limit, and it avoids multiple visits of the same node using some standard technique like the *prefix-preserving property test* [21]. Moreover, the function applies standard optimistic estimate pruning and dynamic quality threshold adjustment. The relevance check is done in line 6, relying on Proposition 2.

Algorithm 1. Iterative Deepening Top-k RelevantSD (*ID-Rsd*)

Input : integer k and database DB over features $\{f_1; ...; f_n\}$
Output : the top-k relevant subgroups

main:
1: **var** *result*= queue with maximum capacity k (initially empty)
2: **var** *minQ*= 0
3: **for** *limit*= 1 to n **do**
4: `findSubgroupsWithDepthLimit`(*result*, *limit*)
5: **return** *result*

procedure `findSubgroupsWithDepthLimit`(*result*, *limit*):
1: **var** stack = new stack initialized with root node
2: **while** *stack* not empty **do**
3: **var** *next*= pop from *stack*
4: **if** *next*'s optimistic estimate exceeds *minQ* and its length does not exceeds *limit* **then**
5: add all successor patterns of *next* to *stack* (avoiding multiple visits)
6: **if** *next* has quality above *minQ* and is not dominated by any $p' \in$ *result* **then**
7: add *next* to *result*
8: update *minQ* if possible

4.3 Complexity

We will now turn to the complexity of our algorithm. Let n denote the number of features in the dataset and m the number of records. The memory complexity is $O(n^2 + kn)$, given that the maximum recursion depth is n, the maximum size of the result queue is k, and every subgroup description has length $O(n)$.

Let us now consider the runtime complexity. For every node visited we compute the quality, test for relevance and consider at most n augmentations. The quality computation can be done in $O(nm)$, while the relevance check can be done in $O(kn)$. The computation of the successors in Line 5 involves the execution of n closure computations, which amounts to $O(n^2 m)$. Altogether, the cost-per-node is thus $O(n^2 m + kn)$. Finally, the number of nodes considered is obviously bounded by $O(|\mathcal{C}_p|\, n)$, where $\mathcal{C}_p$ is the set of closed-on-the-positives and the factor n is caused by the iterative deepening approach.[1]

Table 2 compares the runtime and space complexity of our algorithm with CPosSd. Moreover, we show the complexity of classical and closed subgroup discovery algorithms. Although these algorithms solve a *different*, simpler task, it is interesting to observe they do *not* have a lower complexity. The expression

[1] In case the search space has the shape of a tree, the number of nodes visited by an iterative deepening approach is well-known to be proportional to the size of the tree. Here, however, a tree-shape is not guaranteed, which is why we use the more loose bound involving the additional factor n.

Table 2. Complexity of the different subgroup discovery approaches

Algorithm	Memory	Runtime	Pruning
ID-Rsd	$O(n^2 + kn)$	$O(\lvert\mathcal{C}_p\rvert\,[n^3m + n^2k])$	yes
CPosSd	$\Theta(n\,\lvert\mathcal{C}_p\rvert)$	$\Theta(\lvert\mathcal{C}_p\rvert\,n^2m)$	no
Classical SD	$O(n^2 + kn)$	$O(\lvert\mathcal{S}\rvert\,nm)$	yes
Closed SD	$O(n^2 + kn)$	$O(\lvert\mathcal{C}\rvert\,n^2m)$	yes

$\mathcal{S}$ used in the table denotes set of all subgroup descriptions, while $\mathcal{C}$ denote the set of closed subgroups.

Let us consider the figures in more detail, starting with the memory complexity: Except for CPosSd, all approach can apply depth-first-search (possibly iterated) and thus have moderate memory requirements. In contrast, CPosSd has to collect all closed-on-the-positives, each of which has a description of length n. Please note that no pruning is applied, meaning that $n\,|\mathcal{C}_p|$ is not a loose upper bound for number of nodes stored in memory, but the precise indication — which is why we use the Θ-notation in the table. As the number of closed-on-the-positives can be exponential in n, this approach can quickly become unfeasible.

Let us now turn to the runtime complexity. First, let's compare the runtime complexity of our approach with classic resp. closed subgroup discovery algorithms. Probably the most important difference is that they operate on different spaces. While otherwise the complexity of our approach is higher by a linear factor (resp. quadratic, compared to classic subgroup discovery), the space we consider, i.e. the closed-on-the-positives $\mathcal{C}_p$, can be *exponentially* smaller than the one considered by the other approaches (i.e. $\mathcal{C}$, respectively its superset $\mathcal{S}$). This is illustrated by the following family of datasets:

Proposition 5. *For all $n \in \mathbb{N}^+$, there is a dataset DB_n of size $n+1$ over n features such that the ratio of* closed *to* closed-on-the-positives *is $O(2^n)$.*

Construction 1. *We define the dataset $DB_n = d_1, \ldots, d_n, d_{n+1}$ over the $n+1$ features $f_1, \ldots, f_n$, class as*

$$f_j(d_i) = \begin{cases} 0, & \text{if } i = j \\ 1, & \text{otherwise} \end{cases} \qquad \text{and } class(d_i) = -$$

for $i = 1, ..., n$ and $d_{n+1} = (1, \ldots, 1, +)$.

In these datasets, every non-empty subgroup description is closed and has positive quality. The total number of closed subgroups is thus $2^n - 1$, while there is only one closed-on-the-positives, namely $\{f_1 \ldots f_n\}$. □

Finally, compared to CPosSd, we see that in worst-case our iterative deepening approach causes an additional factor of n (the second term involving k is not much of a problem, as in practice k is relatively small). For large datasets, this disadvantage is however clearly outweighed by the reduction of the memory

footprint. Moreover, as we will show in the following section, in practice this worst-case seldom happens: on real datasets, our approach is mostly not slower than CPosSd, but instead much faster (due to it's use of pruning).

5 Experimental Results

In this section we empirically compare our new relevant subgroup discovery algorithm with existing algorithms. In particular, we considered the following two questions:

- How does our algorithm perform compared to CPosSd?
- How does our algorithm perform compared to classical and closed subgroup discovery algorithms?

We will not investigate and quantify the advantage of the relevant subgroups over standard or closed subgroups, as the value of the relevance criterion on similar datasets has been demonstrated elsewhere (cf. [7]).

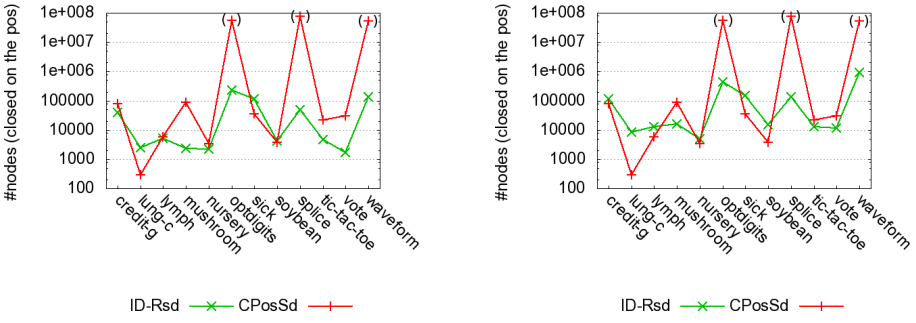
5.1 Implementation and Setup

We implemented our algorithm in JAVA, using conditional datasets but no sophisticated data structures like fp-trees [9] or bitsets [16]. As minor optimization, during the iterative deepening the length limit is increased in a way that length limits for which no patterns exist are skipped (this is realized by keeping track, in every iteration, of the length of the shortest pattern expanded exceeding the current length limit).

Dataset	target class	# rec.	# feat.
credit-g	bad	1000	58
lung-cancer	1	32	159
lymph	mal_lymph	148	50
mushroom	poisonous	8124	117
nursery	recommend	12960	27
optdigits	9	5620	64
sick	sick	3772	66
soybean	brown-spot	638	133
splice	EI	3190	287
tic-tac-toe	positive	958	27
vote	republican	435	48
waveform	0	5000	40

Fig. 2. Datasets

In the following investigation, we use a dozen datasets from the UCI Machine Learning Repository [1], which are presented along with their most important properties in Figure 2. All numerical attributes where discretized using minimal entropy discretization. We run the experiments using two quality functions: the binomial test quality function and the WRACC quality. For pruning, we used the tight optimistic estimate from [8] for the WRACC quality, while for binomial test quality we used the function $\sqrt{|TP(DB, sd)|} \cdot (1 - \frac{|TP(DB,\emptyset)|}{|DB|})$, which can be verified to be a tight optimistic estimate using some basic maths. These optimistic estimates were used in all implementations to make sure that the results are comparable. The experiments were run on a Core2Duo 2.4 GHz PC with 4 GB of RAM.



(a) Bin test quality, k=10 (b) Bin test quality, k=100

Fig. 3. Number of nodes considered during relevant subgroup discovery (brackets indicate memory issues)

5.2 Comparison with CPosSd

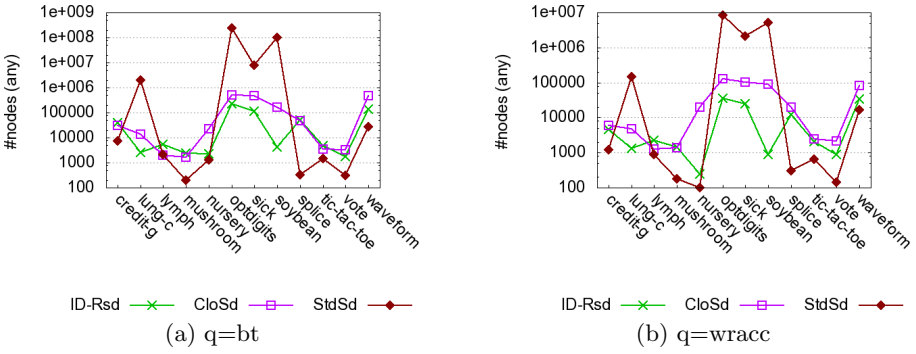
In this paragraph, we compare our algorithm with Garriga et al.'s. In order to abstract from the implementation, we compare the number of visited nodes, rather than the runtime or the exact amount of memory used.

First, in Figure 3 we show the number of nodes considered by our algorithm ("ID-Rsd") and by the other approach ("CPosSd"). We used the binomial test quality, and distinguished between two values for k (10 and 100); the results for other quality function are comparable and omitted for space reasons. The figure shows that for three datasets ('optdigits', 'splice' and 'waveform'), the number of nodes considered by CPosSd was almost 100 millions. As the algorithm CPosSd has to keep all visited nodes in memory, the computation failed: our machine run out of memory for these three datasets. The number of nodes plotted in Figure 3 was obtained by merely *counting* the number of nodes traversed (instead of computing the top-k relevant subgroups). This illustrates that the memory footprint of CPosSd is often prohibitive.

In our approach, on the other hand, there is no need to keep all visited patterns in memory, and hence all computations succeeded. Moreover, in total our approach considers way less nodes than CPosSd. The overall reduction in the number of nodes visited amounts to roughly two orders of magnitude (please note that we used a logarithmic scale in the figures).

5.3 Comparison with Other Subgroup Miners

Next, we compared our algorithm with subgroup miners that solve a different but related task, namely *classical* subgroup discovery and *closed* subgroup discovery. As representative algorithms, we used DpSubgroup [8] and the depth-first closed subgroup miner from [4], which is essentially an adaptation of LCM [21] to the task of subgroup discovery. We remark that these algorithms are also



(a) q=bt (b) q=wracc

Fig. 4. Number of nodes considered by (non-relevant) SD algorithms (k=10)

Table 3. Total num. nodes visited, and percentage compared to StdSd (k=10)

	q=bt			q=wracc		
	StdSd	CloSd	Id-Rsd	StdSd	CloSd	Id-Rsd
total # nodes	346 921 363	1 742 316	590 068	15 873 969	459 434	120 967
percentage (vs. StdSd)	100%	0.5%	0.17%	100%	2.9%	0.76%
total Runtime (sec)	2 717	286	118	147	100	45
percentage	100%	10.5%	4.4%	100%	68%	30%

representative for approaches like the algorithms SD and BSD discussed in Section 3.2, which apply some ad-hoc and possibly incorrect relevance filtering, but otherwise operate on the space of all subgroup descriptions.[2]

Figure 4 shows the number of nodes considered if k is set to 10 and the binomial test, respectively the WRACC quality function is used. Again, we use a logarithmic scale. The results for $k = 100$ are similar and omitted for space reasons. Please note that for our algorithm ("ID-Rsd"), all nodes are closed-on-the-positives, while for the closed subgroup discovery approach ("CloSd") they are closed and for the classic approach ("StdSd") they are arbitrary subgroup descriptions.

The results differ strongly depending on the characteristics of the data. For several datasets, our approach results in a decrease of the number of nodes considered. The difference to the classical subgroup miner DpSubgroup is particularly apparent, where it often amounts to several orders of magnitude.

There are, however, several datasets where our algorithm traverses more nodes than the classical approaches. Again, the effect is particularly considerable when compared with the classical subgroup miner. Beside the overhead caused by the multiple iterations, one reason for this effect is that the quality of the k-th pattern found differs for the different algorithms: For the relevant subgroup algorithm, the k-best quality tends to be lower, because this approach suppresses

[2] Note that BSD becomes faster if its relevance check is disabled [16].

high-quality but irrelevant subgroups. One could argue that it would be more fair to use a larger k-value for the non-relevant algorithms, as their output contains more redundancy.

Overall, our algorithm is competitive (or, somewhat faster) than the other approaches, as the aggregated figures in Table 3 show. Although the costs-per-node are lower for classical subgroup discovery than for the other approaches, overall this does not compensate for the much larger number of nodes traversed.

6 Conclusions

In this paper, we have presented a new algorithm for the task of top-k relevant subgroup discovery. The algorithm is the first that finds the top-k relevant subgroups based on a traversal of the closed-on-the-positives, while avoiding the high memory requirements of the approach of Garriga et al. [7]. Moreover, it allows for the use of optimistic estimate pruning, which reduces the fraction of closed-on-the-positives effectively considered.

The central idea of our algorithm is the memory-efficient relevance test, which allows getting along with only the information about the k best patterns visited so far. Please note that restricting the candidate space to (a subset of) the closed-on-the-positives not only reduces the size of the search space: it also ensures the correctness of our memory-efficient relevance check. The best k patterns can only be used to determine the relevance of a new high-quality patterns if all patterns visited are closed-on-the-positive subgroups – not if we were to consider arbitrary subgroup descriptions. The restriction to closed-on-the-positives is thus a prerequisite for the correctness of our approach.

The new algorithm performs quite well compared to existing approaches. It not only avoids the high memory requirements of the approach of Garriga et al. [7], but also clearly outperforms this approach. Moreover, it is competitive with the existing *non-relevant* top-k subgroup discovery algorithms. This is particularly remarkable as it produces more valuable patterns than those simpler approaches.

Acknowledgments. Part of this work was supported by the German Science Foundation (DFG) under 'GA 1615/1-1' and by the European Commission under 'ICT-FP7-LIFT-255951'.

References

1. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
2. Atzmueller, M., Lemmerich, F., Krause, B., Hotho, A.: Towards Understanding Spammers - Discovering Local Patterns for Concept Characterization and Description. In: Proc. of the LeGo Workshop at ECML-PKDD (2009)
3. Atzmueller, M., Lemmerich, F.: Fast subgroup discovery for continuous target concepts. In: Rauch, J., Raś, Z.W., Berka, P., Elomaa, T. (eds.) ISMIS 2009. LNCS, vol. 5722, pp. 35–44. Springer, Heidelberg (2009)

4. Boley, M., Grosskreutz, H.: Non-redundant subgroup discovery using a closure system. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009. LNCS, vol. 5781, pp. 179–194. Springer, Heidelberg (2009)
5. Bringmann, B., Nijssen, S., Zimmermann, A.: Pattern based classification: a unifying perspective. In: LeGo Worskhop Colocated with ECML/PKDD (2009)
6. Bringmann, B., Zimmermann, A.: The chosen few: On identifying valuable patterns. In: ICDM (2007)
7. Garriga, G.C., Kralj, P., Lavrač, N.: Closed sets for labeled data. J. Mach. Learn. Res. 9, 559–580 (2008)
8. Grosskreutz, H., Rüping, S., Wrobel, S.: Tight optimistic estimates for fast subgroup discovery. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 440–456. Springer, Heidelberg (2008)
9. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD Conference, pp. 1–12 (2000)
10. Klösgen, W.: Explora: A multipattern and multistrategy discovery assistant. In: Advances in Knowledge Discovery and Data Mining, pp. 249–271 (1996)
11. Knobbe, A., Cremilleux, B., Fürnkranz, J., Scholz, M.: From local patterns to global models: The lego approach to data mining. In: From Local Patterns to Global Models: Proceedings of the ECML/PKDD-2008 Workshop (2008)
12. Kralj, P., Lavrač, N., Zupan, B., Gamberger, D.: Experimental comparison of three subgroup discovery algorithms: Analysing brain ischemia data. In: Information Society, pp. 220–223 (2005)
13. Lavrac, N., Kavsek, B., Flach, P., Todorovski, L.: Subgroup discovery with CN2-SD. Journal of Machine Learning Research 5(Feb), 153–188 (2004)
14. Lavrac, N., Gamberger, D.: Relevancy in constraint-based subgroup discovery. In: Constraint-Based Mining and Inductive Databases (2005)
15. Lavrac, N., Gamberger, D., Jovanoski, V.: A study of relevance for learning in deductive databases. J. Log. Program. 40(2-3), 215–249 (1999)
16. Lemmerich, F., Atzmueller, M.: Fast discovery of relevant subgroup patterns. In: FLAIRS (2010)
17. Morishita, S., Sese, J.: Traversing itemset lattice with statistical metric pruning. In: PODS (2000)
18. Nijssen, S., Guns, T., De Raedt, L.: Correlated itemset mining in roc space: a constraint programming approach. In: KDD, pp. 647–656 (2009)
19. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. Inf. Syst. 24(1), 25–46 (1999)
20. Russell, S.J., Norvig, P.: Artificial Intelligence: a modern approach, 2nd International edn. Prentice Hall, Englewood Cliffs (2003)
21. Uno, T., Asai, T., Uchida, Y., Arimura, H.: An efficient algorithm for enumerating closed patterns in transaction databases. In: Discovery Science (2004)
22. Wrobel, S.: An algorithm for multi-relational discovery of subgroups. In: Komorowski, J., Żytkow, J.M. (eds.) PKDD 1997. LNCS, vol. 1263. Springer, Heidelberg (1997)