Fast Discovery of Relevant Subgroups using a Reduced Search Space

Henrik Grosskreutz and Daniel Paurat
Fraunhofer IAIS
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
{henrik.grosskreutz,daniel.paurat}@iais.fraunhofer.de

October 11, 2010

Abstract

We consider a modified version of the local pattern discovery task of subgroup discovery, 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 was considered in many applications, so far no efficient and exact algorithm for this task has been proposed. One particular problem is that the correctness is not guaranteed if the standard pruning approach is applied. In this paper, we devise a new algorithm based on two ideas: For one, we use the theory of closed sets for labeled data to reduce the candidate space; for another we introduce a special search space traversal which allows the use of optimistic estimate pruning while guaranteeing the correctness of the solution. We show that although our algorithm solves a more valuable task than other (classical) approaches, it outperforms all existing subgroup discovery algorithms.

1 Introduction

Subgroup discovery [12, 23] is a local pattern discovery task that searches for interesting sub-populations in labeled data. A typical application is medical data analysis (e.g. [11]), where the label represents some medical condition, e.g. dementia, and a subgroup is a conjunction of features characterizing a sub-population with high positive rate. Unlike in classical association rule mining, in subgroup discovery all subgroups are ranked by a *quality function* which takes into account *both* the support of the subgroup and its distributional unusualness. Based on this ranking, subgroup discovery algorithms search for the top-k subgroups, which are presented to the domain experts for manual examination, or are forwarded to the next step of a complex data-mining process [13, 6].

One important issue in subgroup discovery is the avoidance of *redundancy* in the output [7, 5]. Redundant output is undesirable both because it wastes the human experts precious time (in case of manual examination), and because it can push valuable patterns out of the top-k subgroups. In the case of binary labeled data—the most common setting—a particularly appealing remedy to this issue is the theory of relevance [16, 8]. The idea is to remove all subgroups that are *dominated* (or *covered*) by another subgroup, in the sense that the dominating subgroup covers at least all *positives* (i.e. target-class individuals) in the dominated subgroup, but no additional *negative* (i.e. non target-class individual). The theory of relevance not only allows to get rid of multiple equivalent descriptions for the same subgroup, but also of trivial specializations which provide no additional insight over their generalizations.

Due to these advantages, relevance has been used as a filtering criterion in several applications [16, 14, 2] and can already be considered a standard approach. Up to now, however, no algorithm has been developed that *efficiently* and *exactly* solves the task of relevant subgroup discovery. Although several ad-hoc solutions have been proposed that extend a classical algorithm with a relevance check (e.g. [16, 18]), these cannot guarantee the correctness of the result. The problem is that relevance is a property not defined locally, but with respect to the set of *all* other subgroups. As a result, approaches which traverse the space of candidate subgroups and only keep track of the highest-quality subgroups thereby visited can run into situations where they erroneously add an irrelevant subgroup to their result queue, with the effect that relevant subgroups are missing in the output (we will elaborated on these pitfalls in Sections 1.1 and 3.2).

The only non-trivial algorithm which computes a provably correct solution is that of Garriga et al. [8]. This approach exhaustively collects a superset of the relevant subgroups, and removes every subgroup dominated by some other member of that collection. This is only feasible because the superset considered, the set of so-called *closed-on-the-positives* (which are characterized by some special closure operator), is much smaller than the space of *all* subgroup descriptions. Still, the approach suffers from the drawback that a potentially very large number of patterns has to be traversed *and* collected. This differs from all efficient (non-relevant) subgroup discovery algorithms, which rely on pruning techniques [23, 19, 9] to reduce the number of candidates effectively considered and only collect k subgroups during traversal.

In this paper, we present an algorithm that demonstrates that an *exact* solution to the task of relevant subgroup discovery does not need to be slower than classical subgroup discovery. On the contrary, the restriction to relevant subgroups provides an opportunity to considerably speedup the execution, without sacrificing the correctness guarantee for the result. Our approach relies on the work from Garriga et al. [8], which showed that the relevant subgroups are a subset of the closed-on-the-positives (which in turn are a subset of all subgroups). Our main contribution is to show that using a special traversal strategy on the closed-on-the-positives, we can efficiently verify the relevance of a subgroup the very moment it is visited. Based thereupon, we devise the first correct algorithm that combines a traversal on the closed-on-the-positives with *optimistic estimate pruning* [23], a branch-and-bound technique which is known to result in a dramatic reduction of the execution time [19, 9, 3].

This combination distinguishes our approach from all existing algorithms, which can be categorized with respect to the search space they consider and the pruning strategy they apply. Figure 1 visualizes these two dimensions: the horizontal axis lists the different search spaces, while the vertical axis distinguishes between approaches that prune and others that don't. In the following section, we review related work and categorize it according to these dimensions. The remainder of the paper is structured as follow: In Section 2, we review basic definitions, before we illustrate, in Section 3, the task of relevant subgroup discovery and the pitfalls that can arise. Successively, in Section 4 we present our solution and show its correctness, before we show, in Section 5, that it has a superior runtime in most settings.

1.1 Related Work

The classical subgroup discovery algorithms [12, 16, 4] search for the best subgroups in the space of *all* valid subgroup descriptions. Moreover, they do not prune the candidate space, hence in the categorization of Figure 1 they belong in the upper left area. The algorithms differ in the data structures they use and in their search space traversal strategy (some apply heuristics like beam search which strongly reduce

		search space		
		all subgroup descriptions	closed subgroups	closed-on-the-positives
pruning	none	Explora [12], SD [16], SD-Map [4]		CPosSd[8]
	optim. estim. pruning	DpSubgroup [9], SD-Map* [3], BSD [18]	imr [5]	*Our approach*

Figure 1: Classification of different subgroup discovery algorithms wrt. to the search space they consider and the pruning technique they apply

the execution time, at the price that the correctness of the result is no longer guaranteed).

These classical approaches are outperformed by algorithms like DpSubgroup [9] or SD-Map* [3], which consider the same search space but apply optimistic estimate pruning. These approaches belong to the lower left area of our figure. We remark that lately, so-called 4-support bounds have been proposed, which provide improved estimates [20]; for the sake of simplicity, however, in this paper we will only consider the traditional optimistic estimates.

The algorithm imr [5] adapts the theory of closed itemsets [21, 22] to subgroup discovery, with the motivation to avoid multiple equivalent descriptions of the same subgroup in the output. Thus, it solves a different, more valuable task than the earlier algorithms. It makes use of a closure operator to directly traverse the closed subgroup descriptions, which are a subset of all subgroup descriptions. This is combined with optimistic estimate pruning, which results in a reduction of the execution time on many datasets. In our categorization, this approach belong to the center bottom area.

Equivalent descriptions are not the only kind of redundancy. For example, the top-k subgroups could contain trivial specializations of other subgroups in the result set. Different approaches have been proposed to get rid of such redundancies. A particularly principled approach to this issue is the theory of relevance [17, 16], which unlike other approaches (e.g. the weighted covering approach [15]) is parameter-free and has a clear semantics. Due to these advantages, several authors have equipped subgroup discovery algorithms with a relevance constraint [16, 18, 8]. This extends the classical task to the (more valuable) task of *relevant* subgroup discovery.

Unfortunately, this task is not straightforward to solve. The reason is that relevance is not a property that can be checked locally, but which is defined with respect to *all other subgroups*. Due to this problem, most existing algorithms do not solve the problem exactly: The algorithm SD [16] combines relevance with beam search, which has the effect that it cannot guarantee that the computed subgroups are not dominated by some subgroup outside the beam. The algorithm BSD [18] introduces a bitset representation of the data to speedup the search for the relevant subgroups. Otherwise, it performs an exhaustive traversal, keeping track only of the k highest-quality subgroups visited so far. Some of them can turn out to be irrelevant later. The effect is that although BSD guarantees that all subgroups in the result set are relevant, some relevant high-quality subgroups may be missing. That is, both algorithms do not guarantee an exact solution. As for the search space considered, both algorithms operate on the set of all subgroup descriptions. Thus, in this respect they do not improve over the classical algorithms—in fact, the algorithm BSD becomes *faster* if the relevance check is disabled [18]. In our categorization, these algorithms thus belong into the left areas (top or bottom, depending on their use of pruning).

The work of Garriga et. al. [8] 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. [21]) and relevance, and show that the relevant subgroups are a subset of the subgroups *closed* on the *positively labeled data*. While the focus of the paper is on structural properties and not on computational aspects, the authors also propose a simple two-step algorithm based on this insight. 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 that there is no easy way to use pruning in this approach. In our categorization, it thus belongs to the top right area.

To the best of our knowledge, no approach exists that takes advantage of both optimistic estimate pruning and the theory of closed sets for labeled data.

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 [12] 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. $sd \subseteq \{f_1, \ldots, f_n\}$. 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 $\{f_{i_1}, \ldots, f_{i_k}\}$. The *subgroup extension* described by sd in a database $DB = \{d_1, \ldots, d_m\}$, denoted by $DB[sd]$, is the set of records $d \in DB$ that satisfy 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's extension 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 $\{+, -\}$. The most common quality functions for binary labeled data are of the form:

$$(2.1) \qquad |DB[sd]|^a \cdot \left(\frac{|TP(DB, sd)|}{|DB[sd]|} - \frac{|TP(DB, \emptyset)|}{|DB|} \right)$$

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 2.1 includes some of the most popular quality functions: for $a = 1$, it is order equivalent to the Piatetsky-Shapiro quality function [12] and the weighted relative accuracy WRACC [15], while for $a = 0.5$ it corresponds to the binomial test quality function [12]. Based on these definitions, the task of *subgroup discovery* is defined as follows:

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

$$\forall sd: \; sd \notin G \Rightarrow q(DB, sd) \leq \min_{sd \in G} q(DB, sd).$$

A concept closely related to quality functions is that of an *optimistic estimate* [23]. 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* [19, 9], a topic we will elaborate on in Section 3.2.

2.2 The Theory of Relevancy

The theory of relevance [17, 16] is aimed at eliminating irrelevant patterns. 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.*

- $TP(DB, sd_{irr}) \subseteq TP(DB, sd)$ *and*
- $FP(DB, sd) \subseteq FP(DB, sd_{irr})$.

Here, TP is defined as in 2.1, while $FP(DB, sd) = \{c \in DB[sd] \mid class(c) = -\}$. The above notion allows to distinguish between *irrelevant* and *relevant* subgroups, the former being those dominated by a different subgroup. The idea of the theory of relevance is to restrict consideration to the relevant subgroups, because for each of the other subgroups there is a dominating relevant subgroup which is more valuable.

2.3 Closure Operators and their Connection to Relevance

As shown by Garriga et al. [8], the notion of relevance can be restated in terms of the following mapping between subgroup descriptions:

$$\Gamma^+(X) := \{f \mid \forall d \in TP(DB, X) : f[d] = 1\}. \quad (2.2)$$

Γ^+ is a *closure operator*, i.a. a function defined on the powerset of features $\mathcal{P}(\{f_1, \dots, f_n\})$ such that for all $X, Y \in \mathcal{P}(\{f_1, \dots, f_n\})$, (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 [8] is that

Proposition 1. *The space of relevant patterns consists of all patterns sd_{rel} satisfying the following:*

- *sd_{rel} is closed on the positives, and*
- *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. [21]). However, unlike here in closed pattern mining the closure operator is defined solely on the *support*, without accounting for labels:

$$\Gamma_{sup}(X) = \{f \mid \forall d \in DB[X] : f[d] = 1\}.$$

The fixpoints of this closure operator are simply called *closed* patterns. Many algorithms have been developed to traverse the fixpoints of some *arbitrary* closure operator, e.g. LCM [22]. Making use of this fact, Garriga et al. have proposed a simple two-step approach to find the relevant patterns, to which we will refer as CPosSd.

3 Relevant Subgroup Discovery

As motivated in the introduction, we are not interested in classical subgroup discovery but in the following task:

Task 2. ***Top-k Relevant Subgroup Discovery*** *Given a database DB, a quality function q, and an integer $k > 0$, find a set of subgroup descriptions G of size k, such that*

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

We aim at an algorithm that efficiently solves this task by taking advantage of two techniques:

- *Reduced candidate space:* Instead of considering the space of all valid subgroup descriptions, we want to only consider subgroup descriptions which are closed on the positives. This space can efficiently be traversed, as it is characterized by the closure operator Γ^+ from Equation 2.2;
- *Optimistic estimate pruning*: To reduce the number of subgroups effectively considered, we want to additionally apply optimistic estimate pruning.

Although the general idea is straightforward, the realization raises subtle issues. In fact, the naive application of optimistic estimate pruning can result in incorrect results. To make the problem clear, we will use a simple example dataset, illustrate the different concepts (i.e. relevance, closure, closure on the positives, and optimistic estimate pruning), and show examples where the naive use of optimistic estimate pruning results 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.

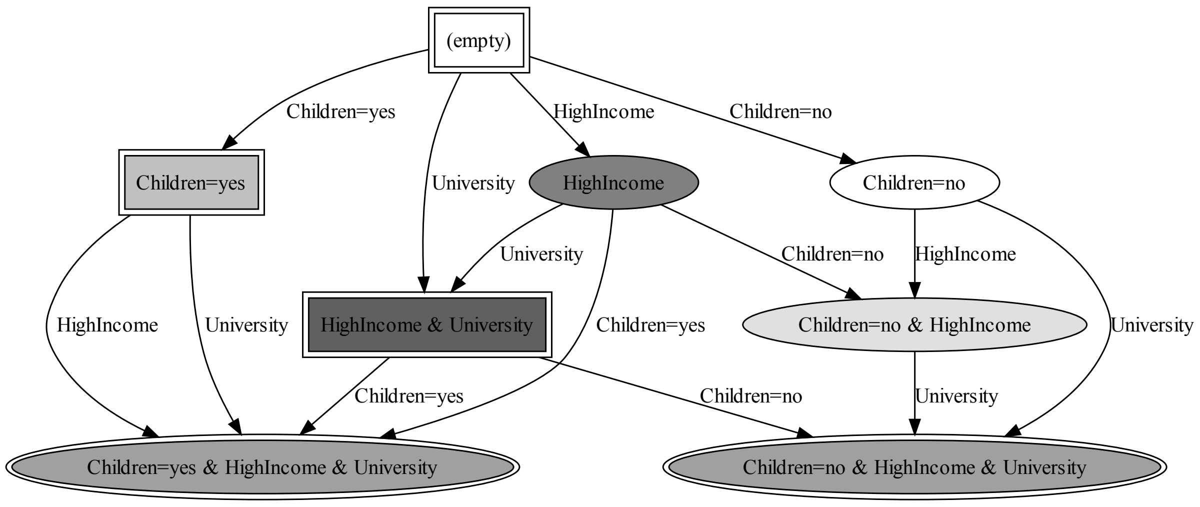


Figure 2: Concept lattice of the exemplary election dataset. Relevant subgroups are highlighted.

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

Table 1: Example: An simple opinion poll dataset

This simple example dataset induces a lattice of candidate subgroup descriptions, containing a total of 16 nodes. These include several redundant representations: for example, both *University* and *HighIncome & University* describe the same extension. Such redundancies are avoided if we consider the sub-space of *closed* subgroups. Figure 2 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 fill-color of a node corresponds to the quality of the subgroup: higher-qualities correspond to more intense gray shades.

The figure illustrates several facts: First, the set of relevant subgroups is a subset of the subgroups closed on the positives, which in turn is a subset of the set of closed subgroups [8]; Second, relevant subgroups need not have a high quality, and neither do high-quality subgroups need to be relevant. For example, the subgroup *(empty)* is relevant but has a low quality—it is simply not interesting, even though it is not dominated. On the other hand, the two subgroups *Children:yes & HighIncome & University* and *Children:no & HighIncome & University* have a high quality, but they are irrelevant as they are merely a fragmentation of the relevant subgroup *HighIncome & University*. This is why we are interested in both relevance *and* high quality, as precised by Task 2. Table 2 shows which of the top-8 subgroup descriptions are closed, respectively relevant. It thereby illustrates that relevant subgroup discovery provides a more dense representation of the interesting patterns in the dataset.

3.2 Challenges and Pitfalls We will now turn to the issues that arise if optimistic estimate pruning is to be applied during relevant subgroup discovery. First, we 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 de-

Subgroup description	Classic sd	Closed sd	Relevant sd	WRACC Quality
HighIncome & University	1st	1st	1st	0.16
University	2nd			0.16
HighIncome	3rd	2nd		0.10
Children=yes & HighIncome & University	4th	3rd		0.08
Children=yes & University	5th			0.08
Children=no & HighIncome & University	6th	4th		0.08
Children=no & HighIncome	7th			0.08
Children=yes	8th	5th	2nd	0.04

Table 2: Subgroups, closed subgroups and relevant subgroups in the example dataset

scriptions, 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. The idea of optimistic estimate pruning is to prune all branches of the search space whose optimistic estimate does not exceed that threshold.

The use of a dynamic threshold is of key importance in optimistic estimate pruning, as it is impossible to calculate a suitable threshold beforehand—the only save option would be to use a trivial threshold like 0. On the other hand, if we search the top-1 subgroup in the space of closed subgroups visualized in Figure 2, the dynamic threshold can allow pruning all but the direct children of the root node: Once these four nodes have been visited, the threshold is set to the quality of subgroup *HighIncome & University*, which allows pruning the whole rest of the search space (because no node has an optimistic estimate that exceeds this quality).

While optimistic estimate pruning based on the best k subgroups visited so far is correct for classical subgroup discovery, in the case of *relevant* subgroup discovery the following problems can occur:

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 which should be part of the result 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 2, 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.*

These issues can arise no matter whether the traversal is based on the closed-on-the-positives, the closed subgroups, or all subgroup descriptions. They are precisely the reason why algorithms like BSD [18] do not guarantee correct results. Of course, one can keep track of *all* subgroups visited and *never* increase the threshold (essentially, this is the approach of [8]). However, without dynamically increasing threshold, optimistic estimate pruning cannot be effectively applied, and the price in terms of an increased execution time can be tremendous.

4 Our Approach

If we wish to efficiently perform optimistic estimate pruning, using the quality of the k-th subgroup in the queue as dynamic threshold, then we clearly need a way to efficiently determine whether a subgroup is relevant or not *at the moment it is visited.* We will now present a traversal strategy which allows doing so, and an algorithm built thereupon.

4.1 The Length-Aware Traversal Strategy If we restrict ourselves to the space of subgroups which are closed on the positives, then a subgroup can only be dominated by a *generalization*, as shown in [8]. This proposition can be extended as follows:

Lemma 2. *Let DB be a dataset and q a quality function of the form of Equation 2.1 (with $0 \leq a \leq 1$). A subgroup sd_{irr} closed on the positives is irrelevant if and only if there exists a relevant subgroup sd_{rel} such that (i) $sd_{rel} \subsetneq sd_{irr}$, (ii) $|FP(DB, sd_{rel})| = |FP(DB, sd_{irr})|$ and (iii) $q(DB, sd_{rel}) > q(DB, sd_{irr})$.*

Proof. It is shown in [8] that sd_{irr} is irrelevant iff. (i) and (ii) hold. Thus we only need to show if it exists, sd_{rel} must have higher quality than sd_{irr}. It holds that $|DB[sd_{rel}]| \geq |DB[sd_{irr}]|$, because sd_{rel} is a subset of sd_{irr}. 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 (ii), it follows that this inequality is equivalent to $TP(DB, sd_{rel})/(|TP(DB, sd_{rel})| + F) > TP(DB, sd_{irr})/(|TP(DB, sd_{irr})| + F)$, where F denotes the common number of false positives. All that remains to show is thus $|TP(DB, sd_{rel})| > |TP(DB, sd_{irr})|$. 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. □

The above lemma is the foundation for our new relevant subgroup discovery algorithm. The idea is to make sure that when a subgroup description is visited, then *all its generalizations have been visited before.* This allows to realize an efficient relevance check based on a queue that stores the top-k relevant subgroups visited so far: when a new subgroup is visited whose quality exceeds that of the k-best relevant subgroup, then this subgroup is relevant *if and only if* it is not dominated by any generalization in the queue. This follows from the above lemma and the observation that if such a subgroup sd_{rel} exists, then it must have been visited before and must still be in the queue (as we will prove in Section 4.3).

Thus, to devise an efficient relevance test we need to ensure that the subgroups are traversed in a generalization-aware order. Although at first it might seem that a breadth-first-search will ensure that a specialization cannot be visited before any of its generalizations, this is not true. The reason is that the parent-child relation is defined wrt. the closure operator Γ^+, which can have the effect that two children of the same parent have different description length; some children can even be generalizations of others. For example, in Figure 2 the children of the root node are *Children=yes*, *HighIncome&University* and *Children=no&HighIncome&University*; the third child is clearly a specialization of the second child (please remember that if we use Γ^+, the search in Figure 2 consists only of the nodes with a double border).

Algorithm 1 Top-k Relevant Subgroup Discovery

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

1:  init toVisit = priorityQueue ordered by length
2:  init resultQ = priorityQueue ordered by quality
3:  init generated = empty set
4:  init $\theta = -\infty$

5:  toVisit.enqueue($\Gamma^+(\emptyset)$)

6:  while not toVisit.isEmpty() do
7:    next= toVisit.dequeueShortest()
8:    if calcQ(next) $> \theta$ and checkIsRel(next, resultQ) then
9:      resultQ.insert(next)
10:     if resultQ.length()>k then
11:       resultQ.removeLast()
12:     if resultQ.length()=k then
13:       $\theta$= resultQ.getKthQuality()
14:   for all direct specializations s of next do
15:     cs= $\Gamma^+(s)$
16:     if calcOE(cs) $> \theta$ and cs$\notin$ generated then
17:       generated.add(cs)
18:       toVisit.enqueue(cs)
19: return resultQ

20: function checkIsRel(cs, resultQ): boolean
21:   if $\exists g \in$ resultQ :
22:         $g \subsetneq cs \wedge |FP(g)| = |FP(cs)| \wedge q(g) > q(cs)$
23:   then return false else return true
```

4.2 The Algorithm Based on the above considerations, our algorithm applies a special traversal strategy that ensure that whenever a node is visited, all generalizations have been visited before. Technically, this is realized by replacing the FIFO used in standard breadth-first-search by a priority queue which orders the nodes by their length, and always returns a subgroup of minimum description length.

Algorithm 1 shows the corresponding pseudo-code. The length-aware priority queue is referenced by the variable *toVisit*, which is used to orchestrate the traversal. The queue is initially with a single node, namely the closure-on-the-positives of the empty subgroup description. Beside this variable, the algorithm uses a second priority queue, *resultQ*, to collect the k highest-quality relevant subgroups visited. Additionally, the set *generated* is used to avoid multiple visits by keeping track of the nodes already generated. Finally, the algorithm uses the threshold variable θ, initialized to $-\infty$.

The traversal of the search space is done in the while-loop starting in Line 6. The queue *toVisit* is used to determine the (minimum-length) subgroup to be considered next. Its quality is calculated and compared with the threshold; if it exceeds the latter, the relevance is tested, using the function `checkIsRel`. This function, defined in Line 20, essentially tests the conditions from Lemma 2. If the subgroup passes the relevance test, it is added to the *resultQ* and the threshold is increased (if possible). Afterwards, the closed-on-the-positive specializations are considered. If their optimistic estimate is sufficient and they have not already been generated, they are added to *toVisit*.

4.3 Properties In this section, we will prove the correctness of our algorithm and consider some interesting properties.

4.3.1 Correctness As first step, we show that the search space traversal strategy ensures that generalizations are considered before their specializations:

Lemma 3. *If the relevance check is invoked for a subgroup description sd (in Line 8), then each of sd's closed-on-the-positive generalizations have been dequeued (in Line 7) at some earlier time of execution.*

Proof. First, we show that for each *sd* considered, all its closed-on-the-positive generalizations were *en*queued (in Line 18) at some earlier time.

(By contradiction:) Assume that there is a subgroup description $sd_{gen} \subsetneq sd$ violating this proposition, i.e. which has not been enqueued before the relevance check was invoked on *sd*. Neither sd_{gen} nor *sd* can be the closure of the empty set (this would directly lead to a contradiction). Thus, there must exist a common ancestor of these two descriptions which has been dequeued before *sd*. More precisely, there must be a sequence $sd_1, ..., sd_n$ such that sd_{i+1} is a "child" of sd_i (i.e. sd_{i+1} is the positive closure of a direct specialization of sd_i), $sd_n = sd_{gen}$, and *sd* was generated from sd_1 (by a sequence of invocations of the lines 14 and 15). By construction, sd_1 has been dequeued before *sd*, while by assumption $sd_n = sd_{gen}$ has not be enqueued before *sd*. Thus, there must exists an index $i*$ such that sd_{i*} has been dequeued before *sd*, while sd_{i*+1} has not been dequeued before *sd*. However, sd_{i*+1} must have been *en*queued before *sd* was *de*queued. This is because sd_{i*+1} is a child of sd_{i*} (which has been dequeued), and sd_{i*+1}'s optimistic estimate is $\geq q(sd)$ (because $sd_{i*} \subset sd$) and hence greater than θ. But then, sd_{i*+1} must still have been in the queue when *sd* was dequeued. This is a contradiction, because $sd_{i*+1} \subsetneq sd_{gen} \subsetneq sd$ and hence $length(sd_{i*+1}) < length(sd)$.

Finally, if every closed-on-the-positives generalization has been *en*queued before the relevance check for *sd* is invoked, then they must also all have been *de*queued before, because their description is shorter than *sd*'s. □

This peculiarity of the traversal strategy ensures the correctness of our algorithm:

Theorem 4. *Algorithm 1 correctly solves Task 2 ("top-k relevant subgroup discovery").*

Proof. To prove the overall correctness, it suffices to show the correctness of the function `checkIsRel`. Otherwise, Algorithm 1 performs the same traversal and pruning as other algorithms, and can thus be shown to be correct using the same arguments (cf. [5]).

The correctness of `checkIsRel` needs only to hold for subgroups *cs* of quality $> \theta$. Here, the correctness follows from Lemma 2 and Lemma 3. If *cs* is irrelevant, then there exists a relevant generalization with higher quality and same negative support (Lemma 2). This generalization must have been visited earlier during the execution (Lemma 3). The generalization must still be enqueued in *resultQ*, because its quality exceeds θ. Thus, if *cs* is irrelevant then there is a generalization satisfying the conditions in Line 22.

The inverse implication also holds: If *cs* is relevant, then by Lemma 2 there cannot be a subgroup satisfying the conditions in Line 22. □

4.3.2 Runtime Complexity We will now turn to the runtime complexity of our algorithm. Let n denote the number of features in the dataset and m the number of records. For every node visited (i.e. dequeued in Line 7), we compute the quality, test for relevance and consider at most n augmentations. The quality computation and the relevance check can be done in $O(nm)$ (for a fixed k). For each augmentation, we compute the closure and check if the result was already generated. The closure computation has complexity $O(nm)$, while the checks can be performed in $O(n)$ using a prefix tree. Finally, the costs for the enqueue/dequeue operations are logarithmic in the size of the queue, the latter being $< 2^n$. Altogether, the complexity is thus $O(|\mathcal{C}_p|\, n^2 m)$, where $\mathcal{C}_p$ is the set of closed-on-the-positives visited.

For comparison, let us consider the complexity of classical and closed subgroup discovery algorithms. The former have complexity $O(|\mathcal{S}|\, nm)$, where $\mathcal{S}$ is the set of subgroups visited, while the latter are $O(|\mathcal{C}|\, n^2 m)$, where $\mathcal{C}$ is the set of closed subgroups visited (cf. [5]). This means that the cost-per-node of classical approaches can be smaller by a linear factor, while for closed subgroups they are the same. On the other hand, the number of nodes considered can strongly differ: the set of all subgroups can be exponentially larger than the set of closed subgroups [5]. A similar reduction can occur when we move from the closed subgroups to those closed on the positives:

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

Construction 1. *We construct the dataset $DB_n = d_1, \ldots, d_n, d_{n+1}$ over the n features $f_1, \ldots, f_n$ according to the scheme illustrated in the following table:*

Id	*Label*	f_1	f_2	...	f_n
d_1	-	0	1	...	1
d_2	-	1	0	...	1
...	-	1	1	...	1
d_n	-	1	1	...	0
d_{n+1}	+	1	1	...	1

That is, the first n records have label "-" and all features except the one in the diagonal have value 1; the last record has label "+" and all its features have value 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 fixpoint of Γ^+, namely $\{f_1 \ldots f_n\}$. □

4.3.3 Length Limits

A common practice in subgroup discovery is to restrict the search depth to reduce the runtime. Adding such a restriction to our algorithm requires only minor modifications. Essentially, one has to make sure that a subgroup description is only enqueued if it satisfies the length limit.

The effect of such a modification is, obviously, that some longer subgroups may be missing in the result set. Given that relevance is a concept that is defined with respect to *all* other subgroups, however, it is not obvious that such a length limit will preserve the property of the algorithm to only output relevant subgroups.

Fortunately, the output consists only of relevant subgroups, even if a length limit is applied. The reason is that a subgroup visited via a traversal based on Γ^+ can only be dominated by its *generalizations*. These all have a shorter description, and must thus have been considered beforehand.

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 existing relevant subgroup discovery algorithms?

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
sick	sick	3772	66
soybean	brown-spot	638	133
splice	EI	3190	287
tic-tac-toe	positive	958	27
vote	republican	435	48

Table 3: Datasets

- 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 the other approaches, as this issue has been addressed elsewhere (cf. [8]).

5.1 Setup

Our implementation is basically a JAVA version of Algorithm 1. It uses a prefix tree to implement *generated*, and a binary heap to implement the priority queue *toVisit*. We use conditional datasets, but no sophisticated data structures like fp-trees [10] or bitsets [18]. As minor optimization, every term whose addition to a subgroup results in an optimistic-estimate-prunable subgroup is marked as prunable in all its children. The implementation can be downloaded under `http://g3cko.org/SubgroupDiscovery/RelSD.zip`.

We performed our investigation using ten datasets from the UCI Machine Learning Repository [1], which are presented along with their most important properties in Table 3. 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 [9] 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 an Intel Core2Duo 2.4 GHz PC with 4 GB of RAM.

5.2 Comparison with relevant subgroup miners

Although several algorithms exist which tackle the task of relevant subgroup discovery, only the approach of Garriga et al. [8] solves this task exactly; All other existing approaches do not guarantee the correctness of the result, as discussed in Section 1.1. Thus, in this paragraph we only compare our algorithm with Garriga et al.'s. To ensure comparability of the results, we used a JAVA reimplementation of their two-step algorithm, which suppresses all but the k highest-quality relevant subgroups in a post-processing step.

Figure 3a shows the number of nodes considered by our algorithm ("RelSd") and by the other approach ("CPosSd") if the binomial test quality is used, and k is set to 10. Figure 3b shows the corresponding results if the WRACC quality is used instead. As expected, the plots confirm that the number of closed-on-the-positives considered by our algorithm is considerably reduced, sometimes by several orders of magnitude (note that the figures are plotted using a logarithmic scale). The number of nodes considered has a roughly linear impact on the runtime, which will be considered in detail in Section 5.4.

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 [9] and imr [5] which are also implemented in JAVA. We remark that these algorithms are also representative for approaches like the algorithms SD and BSD discussed in Section 1.1, which apply some ad-hod and possibly incorrect relevance filtering, but otherwise operate on the space of all subgroup descriptions.[1]

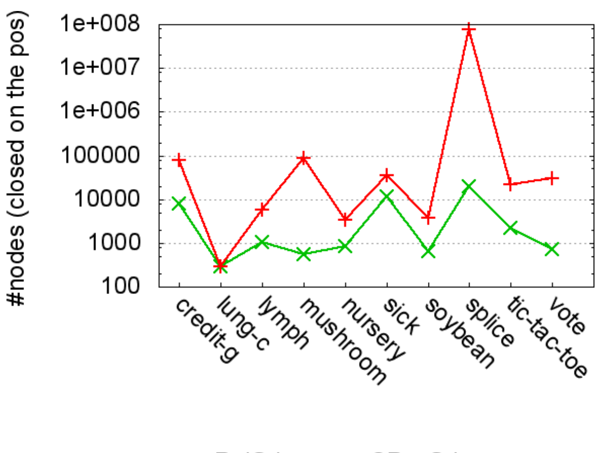
Figure 4a shows the number of nodes considered if the binomial test quality function is used, while Figure 4b considers the case when the WRACC quality is used. Again, the figures use a logarithmic scale. Please note that for our algorithm ("RelSd"), all nodes are closed-on-the-positives, while for the closed subgroup discovery approach ("CloSd") they are closed and for the classic approach ("StdSd") they are arbitrary subgroup descriptions.

For several datasets, the number of nodes considered by our algorithm ("RelSd") is considerably smaller; however, there are some exceptions. This might seem surprising at first, given that our algorithm considers a smaller candidate space than the other approaches. The reasons for this are the following:

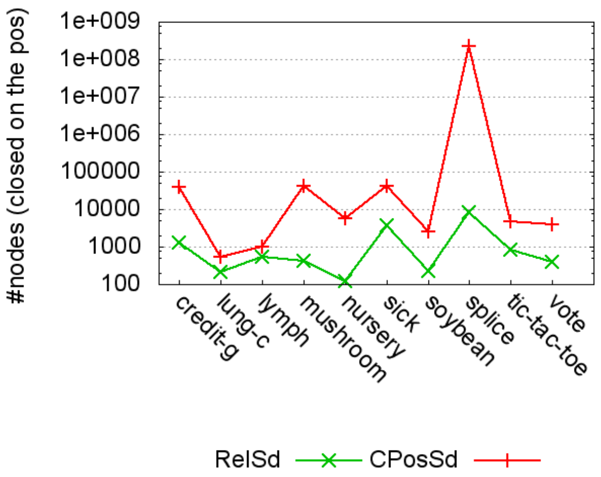
- First and foremost, the quality of the k-th subgroup differs for the different algorithms. For the relevant subgroup algorithm, the k-best quality tends to be lower, simply because this approach suppresses high-quality but irrelevant subgroups. This effect can be seen by reconsidering Table 2. As a result, the threshold used by our algorithm tends to be lower than for the other algorithms, which allows less pruning. This effect is the price if we want to suppress irrelevant subgroups but require the *same* total number of subgroups.

- Second, the algorithms apply different traversal strategies, and sometimes the other algorithms visit the high-quality subgroups earlier.

[1] Of course, BSD makes use of more sophisticated data structures. These, however, do not affect the number of nodes considered, and could also be used in our approach.



(a) Binomial test quality

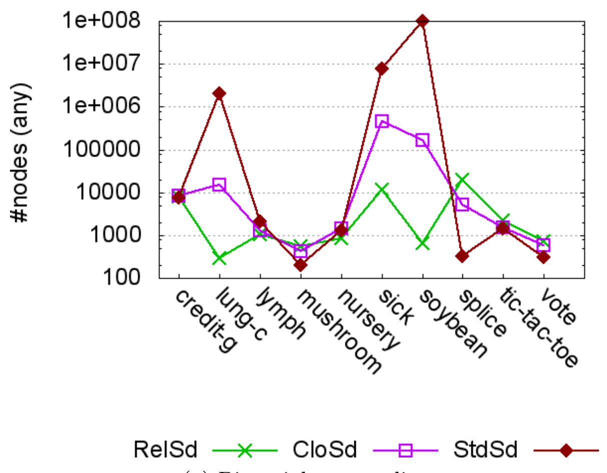


(b) WRACC quality

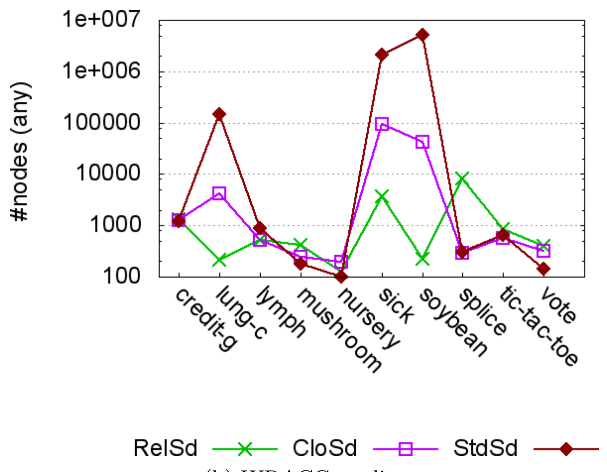
Figure 3: Number of nodes considered during the task of relevant subgroup discovery ($k = 10$)

5.4 Speedup

Finally, in Table 4 we compare the runtime of our algorithm with the other approaches. Table 4a shows the *speedup* provided by our algorithm when the binomial test quality is used (that is, the



(a) Binomial test quality



(b) WRACC quality

Figure 4: Number of nodes considered by different subgroup discovery algorithms ($k = 10$)

	Speedup: RelSd vs ...		
Dataset	CPosSd	CloSd	StdSd
Credit	253	3.18	*0.93*
Lung	8.2	12.6	342
Lymph	37	7.38	1.1
Mushroom	58	1.36	*0.5*
Nursery	41	3.13	1.4
Sick	13	30.5	35.9
Soybean	35	127	15188
Splice	157226	*0.62*	*0.22*
TicTacToe	134	5.71	1.3
Vote	149	9.27	*0.62*
Total	39207	18.8	219

(a) Binomial test quality

	Speedup: RelSd vs ...		
Dataset	CPosSd	CloSd	StdSd
Credit	910	4.98	1.5
Lung	11	8.52	28.89
Lymph	89	14.91	2.82
Mushroom	58	1.35	*0.25*
Nursery	65	4.19	1.64
Sick	44	22.73	32.24
Soybean	52	50.3	1004
Splice	259089	*0.56*	*0.34*
TicTacToe	313	12.93	2.07
Vote	229	13	*0.88*
Total	81360	9.6	29.7

(b) WRACC quality

Table 4: Speedup ($k = 10$)

runtime of the other approaches divided by the runtime of our algorithm). The table also shows the total speedup, i.e. the speedup concerning the analysis of *all* datasets. The corresponding figures for the WRACC quality are shown in Table 4b. The tables show that our approach outperforms the other approaches in most settings, often by several orders of magnitude.

More specifically, the algorithm CPosSd is outperformed in every setting, by up to 5 orders of magnitude. Closed subgroup discovery—which solves a less valuable task—is outperformed in 18 out of 20 case. For better readability, the settings where our algorithm is not faster are printed using italics.[2] In total, our algorithm outperforms the closed subgroup algorithm by (more than) one order of magnitude. Finally, for the classical subgroup miner there is quite some variance in the speedup. On one hand, for 7 out of 20 datasets our approach is slightly slower. On the other hand, in 13 out of 10 datasets our approach is faster, sometimes by 3 or 4 orders of magnitude. In total, our algorithm is again clearly faster, by more than one order of magnitude.

Summarizing, the new algorithm clearly outperforms CPosSd, the only other exact relevant subgroup discovery algorithm. Moreover, it is faster than the other (non-relevant) subgroup discovery algorithms, while it solves a more valuable task. This is a very positive answer to our two questions.

[2]The careful reader might have noticed that the closed subgroup discovery algorithm is also outperformed on some datasets where it considers slightly less nodes than our algorithm. The reason is that for both approaches, the closure computation is the most expensive step. However, our algorithm does only need to consider the positive records in the closure computation, unlike the closed subgroup miner which has to consider all records.

6 Conclusions

In this paper, we have presented a new algorithm for the task of relevant subgroup discovery. The algorithm is the first that combines optimistic estimate pruning with a traversal of the subgroups closed-on-the-positives. We have described the pitfalls that arise when these techniques are combined, and how we overcome them. In the empirical evaluation section, we have shown that our algorithm is on average *faster* than all existing subgroup discovery algorithms, while solving a *more valuable* task than the classical algorithms.

It is important to note that using the closed-on-the-positives as candidate space not only reduces the size of the search space, but actually is a prerequisite for the correctness of our algorithm. In fact, it ensures the correctness of the relevance check, which is at the heart of our algorithm. Among other prerequisites (like the traversal strategy), this test makes use of the fact that closed-on-the-positive subgroups can only be dominated by their *generalizations* – a property that does *not* hold if we consider arbitrary subgroup descriptions.

An issue that deserves further investigations is the space complexity of our algorithm. Our traversal strategy has a rather high memory footprint, compared with depth-first approaches. One option to reduce the memory requirements (without sacrificing the efficient relevance check) would be to apply an iterative deepening scheme. It would also be interesting to investigate the use of the more advanced 4-support bounds [20] within our approach, instead of the optimistic estimates used in this study. Another interesting question is whether it is possible to efficiently calculate equivalent minimum-length descriptions for the relevant subgroups, as in the case of closed subgroups [5]. We plan to investigate these issues in future work.

References

[1] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[2] M. Atzmueller, F. Lemmerich, B. Krause, and A. Hotho. Towards Understanding Spammers - Discovering Local Patterns for Concept Characterization and Description. In *From Local Patterns to Global Models, Proc. of the Workshop at ECML-PKDD*, 2009.

[3] Martin Atzmüller and Florian Lemmerich. Fast subgroup discovery for continuous target concepts. In *ISMIS*, pages 35–44, 2009.

[4] Martin Atzmüller and Frank Puppe. SD-map - a fast algorithm for exhaustive subgroup discovery. In *PKDD*, pages 6–17, 2006.

[5] Mario Boley and Henrik Grosskreutz. Non-redundant subgroup discovery using a closure system. In *ECML/PKDD (1)*, 2009.

[6] B. Bringmann, S. Nijssen, and A. Zimmermann. Pattern based classification : a unifying perspective. In *LeGo worskhop colocated with ECML/PKDD*, 2009.

[7] Björn Bringmann and Albrecht Zimmermann. The chosen few: On identifying valuable patterns. In *ICDM*, 2007.

[8] Gemma C. Garriga, Petra Kralj, and Nada Lavrač. Closed sets for labeled data. *J. Mach. Learn. Res.*, 9:559–580, 2008.

[9] Henrik Grosskreutz, Stefan Rüping, and Stefan Wrobel. Tight optimistic estimates for fast subgroup discovery. In *ECML/PKDD (1)*, pages 440–456, 2008.

[10] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD Conference*, pages 1–12, 2000.

[11] A. Hapfelmeier, J. Schmidt, M. Mueller, R. Perneczky, A. Drzezga, A. Kurz, and S. Kramer. Interpreting pet scans by structured patient data: A data mining case study in dementia research. In *ICDM*, 2008.

[12] W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249–271. 1996.

[13] A. Knobbe, B. Cremilleux, J. Fürnkranz, and M. Scholz. From local patterns to global models: The lego approach to data mining. In *From Local Patterns to Global Models: Proceedings of the ECML/PKDD-08 Workshop*, 2008.

[14] Petra Kralj, Nada Lavrač, Blaž Zupan, and Dragan Gamberger. Experimental comparison of three subgroup discovery algorithms: Analysing brain ischemia data. In *Information Society*, pages 220 – 223, 2005.

[15] N. Lavrac, B. Kavsek, P. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5(Feb):153–188, 2004.

[16] Nada Lavrac and Dragan Gamberger. Relevancy in constraint-based subgroup discovery. In *Constraint-Based Mining and Inductive Databases*, 2005.

[17] Nada Lavrac, Dragan Gamberger, and Viktor Jovanoski. A study of relevance for learning in deductive databases. *J. Log. Program.*, 40(2-3):215–249, 1999.

[18] Florian Lemmerich and Martin Atzmueller. Fast discovery of relevant subgroup patterns. In *FLAIRS*, 2010.

[19] S. Morishita and J. Sese. Traversing itemset lattice with statistical metric pruning. In *PODS*, 2000.

[20] Siegfried Nijssen, Tias Guns, and Luc De Raedt. Correlated itemset mining in roc space: a constraint programming approach. In *KDD*, pages 647–656, 2009.

[21] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Inf. Syst.*, 24(1):25 – 46, 1999.

[22] T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *Discovery Science*, 2004.

[23] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *PKDD*. Springer, 1997.