Fast Discovery of Relevant Subgroup Patterns

Florian Lemmerich and **Mathias Rohlfs** and **Martin Atzmueller**
University of Würzburg,
Department of Computer Science VI
Am Hubland, 97074 Würzburg, Germany
{lemmerich, rohlfs, atzmueller}@informatik.uni-wuerzburg.de

Abstract

Subgroup discovery is a prominent data mining method for discovering local patterns. Since often a set of very similar, overlapping subgroup patterns is retrieved, efficient methods for extracting a set of relevant subgroups are required. This paper presents a novel algorithm based on a vertical data structure, that not only discovers interesting subgroups quickly, but also integrates efficient filtering of patterns, that are considered irrelevant due to their overlap. Additionally, we show how the algorithm can be easily applied in a distributed setting. Finally, we provide an evaluation of the presented approach using representative data sets.

Introduction

Subgroup discovery is a data mining technique that focuses on finding parts of a dataset, that show an interesting behavior with respect to a specified concept of interest. In the medical domain for example, we could be interested in discovering subgroups of patients that show a high deviation of the diagnosis *gallstones* compared to the total population, i.e., a higher or lower risk of developing gallstones. Similar to the rule body of association rules, subgroups are usually described by a conjunction of attribute-value pairs; the concept of interest can then be regarded as the rule head. Results of subgroups can be used for both, immediate presentation of the obtained knowledge to domain experts or to build any predictive global model, e.g., for classification.

Since subgroup discovery is a non-covering approach, in real-world applications often a set of very similar (and overlapping) subgroups is retrieved by a typical automatic k-best discovery task. This can cause a decreased interestingness of the k-best set of patterns, but also a loss of information since further potentially interesting and more diverse subgroups patterns are suppressed and hidden from the user. For an effective approach, a compact set of relevant subgroup patterns needs to be identified efficiently. However, as most state-of-the-art discovery algorithms do not integrate a filter for overlapping subgroups, either an inefficient check needs to be incorporated, or the filtering has to be applied in separate post-processing step. This can result in a massive reduction of the discovered patterns.

In this paper, we therefore introduce the novel BSD algorithm, that is tailored to the task of discovering *relevant* subgroup patterns. An efficient data structure and advanced pruning strategies enable a fast examination of the search space. Additionally, the utilized vertical data structure allows for a very efficient detection of subgroup descriptions, that due to their overlap are irrelevant to each other. Thus, by suppressing such patterns a final set of *relevant* patterns can be identified. Furthermore, we present an extension of this approach, that enables the parallelization of the search in multiple processes in order to distribute the discovery effort and to accomplish further gains in performance. A practical evaluation using data sets with varying data characteristics and sizes from representative real-world applications and from the well-known UCI-repository (Newman et al. 1998), shows the benefit of the presented approaches.

The rest of the paper is structured as follows: We first summarize the basics of subgroup discovery and the handling of (ir-)relevant patterns. Then, different approaches for the efficient mining of those patterns, including a new specialized algorithm, that is, the bitset-based BSD algorithm, are discussed. Furthermore, we present a parallelized adaptation of this algorithm. Next, we discuss related work. After that, a comprehensive practical evaluation is given. The paper concludes with a summary and interesting directions for future research.

Preliminaries

In the following, we introduce the necessary notions and give a brief overview on the task of subgroup discovery. Then, we formally define irrelevant subgroup patterns.

Subgroup Discovery

Subgroup discovery is applied for finding relations between a (dependent) target variable and a set of explaining (independent) variables. Then, the goal is to describe subsets of the data, that have the most unusual characteristics with respect to the concept of interest given by the target variable (Wrobel 1997).

For some basic notation, let Ω_A denote the set of all attributes. For each attribute $a \in \Omega_A$ a range $dom(a)$ of values is defined. Let CB be the case base (data set) containing all available cases (instances). A case $c \in CB$ is given by the m-tuple $c = ((a_1 = v_1), \ldots, (a_m = v_m))$ of $m = |\Omega_A|$ attribute values, $v_i \in dom(a_i)$ for each a_i.

The subgroup description language specifies the individuals belonging to the subgroup. For a commonly applied single-relational propositional language a subgroup description can be defined as follows:

Definition 1 (Subgroup Description) *A subgroup description $sd(s) = \{e_1, \ldots, e_k\}$ of the subgroup s is defined by the conjunction of a set of selection expressions (selectors). The individual selectors $e_i = (a_i, V_i)$ are selections on domains of attributes, $a_i \in \Omega_A, V_i \subseteq dom(a_i)$. We define Ω_E as the set of all selection expressions and Ω_{sd} as the set of all possible subgroup descriptions.*

A subgroup s described by the subgroup description $sd(s)$ is given by all cases $c \in CB$ covered by the subgroup description $sd(s)$. A subgroup s' is called a *refinement* of s, if $sd(s) \subset sd(s')$.

A subgroup discovery task usually searches for the k best subgroups according to a quality function, which measures the interestingness of the subgroup. Typical quality criteria include the difference in the distribution of the target variable concerning the subgroup and the general population, and the subgroup size. While in principle any function can be used as quality function, usually they satisfy the (monotony) axioms postulated in (Klösgen 1996). Essentially, these imply, that larger subgroups and subgroups with a higher frequency of the target concept are considered more interesting with an increase in these parameters.

Definition 2 (Quality Function) *A quality function $q : \Omega_{sd} \times \Omega_E \rightarrow \mathbb{R}$ is used in order to evaluate a subgroup description $sd \in \Omega_{sd}$ given a target concept $t \in \Omega_E$, and to rank the discovered subgroups during search.*

For binary target variables, many important quality functions can be specified in the form

$$q_a = n^a(p - p_0), a \in [0; 1]$$

where p is the relative frequency of the target variable in the subgroup, p_0 is the relative frequency of the target variable in the total population, and n denotes the size of the subgroup. a is a parameter, that trades off the increase in the target share $p - p_0$ vs. the generality n of the subgroup. For $a = 1$, for example, this results in the quality function of Piatetsky-Shapiro q_{PS}, while for $a = 0$ the resulting functions is order equivalent to the relative gain function q_{RG}.

For this family of functions an *optimistic estimate* of a subgroup s can be specified, cf. (Grosskreutz, Rüping, and Wrobel 2008; Atzmueller and Lemmerich 2009). This approximation describes an upper boundary for the quality, that any refinement of s can have. These boundaries can then be utilized in exhaustive search algorithms to prune (large) parts of the search space while maintaining the optimality of the search.

Irrelevant Patterns

The result of subgroup discovery is a set of subgroups. Since subgroups can overlap, relevancy analysis is essential in order to identify a compact but comprehensive set of subgroups. We consider a subgroup s as more relevant with respect to another subgroup s', if it covers all positive examples of s', but no negative examples, that are not covered by s' as well. Formally s' is irrelevant with respect to s, if and only if

$$TP(s') \subseteq TP(s) \text{ and } FP(s) \subseteq FP(s'),$$

where $TP(s) = \{c \in s | t(c) = true\}$ is the set of positive examples in the subgroup s and $FP(s) = s \setminus TP(s)$ the set of negative examples in s. If s' is irrelevant with respect to s, then the quality of a subgroup s' is lower or equal to the quality of subgroup s for any quality function satisfying the axioms postulated in (Klösgen 1996).

Filtering out irrelevant subgroups has been successfully applied in practical applications, cf. (Garriga, Kralj, and Lavrač 2008), and can be considered as a standard option for subgroup discovery.

Efficient Vertical Subgroup Discovery

In the following section we present strategies for the efficient mining of the best k relevant subgroups.

Naive Subgroup Filtering 1: Postprocessing

The simplest filtering approach requires the application of a standard (exhaustive) discovery algorithm, e.g., the SD-Map* (Atzmueller and Lemmerich 2009) or the DpSubgroup algorithm (Grosskreutz, Rüping, and Wrobel 2008), without incorporating adaptations. Then, any irrelevant subgroups can be removed in a separate postprocessing step. While this procedure allows for a relatively fast mining process, the size of the result set is rather unpredictable, as the number of irrelevant, filtered patterns is dataset dependent. As shown in the evaluation section, it can vary from few single subgroups to the vast majority of the result set. Therefore, to find the set of the best k relevant subgroups, a very large set of subgroups must be retrieved by the search-algorithm, thus limiting the pruning possibilities of the applied methods.

Naive Subgroup Filtering 2: During Search

To avoid a potentially massive reduction of the size of the result set during postprocessing, a relevancy-check can be included into the search algorithm. Thus, whenever a subgroup s is added to the result set S, it is checked, if s is irrelevant to any subgroup $s' \in S$. If this is not the case, the subgroup s can be added to the set S. Afterwards for any subgroup $s'' \in S$ it is checked, if s'' is covered by s and thus can be removed. Please notice, that by doing so more than one subgroup can be removed from the set, while only one is added. However, in contrast to the postprocessing approach the resulting reduction of the set size is in practice very limited, so emerging problems can be avoided in most cases by only a slight increase of the result set size.

The naive approach requires one pass over the database for each such check. Thus, the needed time to check for irrelevant patterns can easily exceed the time needed for the regular subgroup search by far.

A Bitset-based Subgroup Discovery Algorithm

Checking each possible coverage of subgroups in a separate database pass is obviously a very time consuming task. To speed up this process we propose a new vertical subgroup mining algorithm, which is tailored to the task of finding relevant subgroups. It combines an efficient vertical *bitset based* (vectors of zeros and ones) representation of the needed information with advanced pruning strategies and an efficient relevancy check. Bitsets are implemented very time and memory efficient in most programming languages, including logical operations like OR/AND.

Algorithm 1 function bsd

```
Require:
    $sel_{cond}$: List of conditioned selectors,
    $sel_{rel}$: List of relevant selectors,
    $c_{condPos}$: Bitset of positive instances for $sel_{cond}$
    $c_{condNeg}$: Bitset of negative instances for $sel_{cond}$
    depth: Current search depth,
    res: The result set of the best $k$ found subgroups
Ensure:
    res as a set of the best $k$ relevant subgroups
 1: $newSel_{rel}$:= new List()
 2: for all Selector $s_{curr}$ in $sel_{rel}$ do
 3:     $c_{currPos}$= $c_{condPos}$ AND($s_{curr}$.bitsetPos)
 4:     tp = $c_{currPos}$.cardinality()
 5:     if optEstimate(tp) > res.getMinQuality() then
 6:         $c_{currNeg}$= $c_{condNeg}$ AND($s_{curr}$.bitsetNeg)
 7:         $n$ = tp + $c_{currNeg}$.cardinality()
 8:         $newSel_{rel}$.add($s_{curr}$)
 9:         $s_{curr}$.attach($c_{currPos}$, $c_{currNeg}$, optEstimate(tp))
10:         if quality(tp, n) > res.getMinQuality() then
11:             r = checkRel(res, $c_{currPos}$, $c_{currNeg}$)
12:             if r then
13:                 sg= createSubgroup($sel_{cond}$, $s_{curr}$)
14:                 res.add(sg, $c_{currPos}$, $c_{currNeg}$)
15:                 res.checkRelevancies(sg)
16:                 if res.size > k then
17:                     result.removeLowestQualitySubgroup()
18: sort ($newSel_{rel}$)
19: if depth < MAXDEPTH then
20:     for all Selector s: $newSel_{rel}$ do
21:         if s.optEstimate > res.getMinQuality() then
22:             $newSel_{rel}$(s)
23:             $sel_{cond}$.add(s)
24:             $c_{new}$= getCurrentBitSetFor(s)
25:             bsd ($sel_{cond}$+ s, $newSel_{rel}$, s.getPositives(),
                s.getNegatives(), depth + 1, res)
```

The bitset based subgroup discovery algorithm (BSD) for exhaustive mining of relevant sets of subgroups uses a branch-and-bound strategy, where a conditioned search space is mined recursively, similar to the SD-Map* (Atzmueller and Lemmerich 2009) or the DpSubgroup algorithm, e.g., (Grosskreutz, Rüping, and Wrobel 2008).

In the initialization phase we construct two bitsets for each selector involved in the subgroup discovery. The first bitset represents the cases of the database with a positive target value, the latter the cases with a negative target value. Each bit represents a single case in the database, so the i-th bit in each selectors positives bitset represents the i-th case in the database labeled as positive. It is set to true, if (and only if) the respective selector is true for the respective case. The construction of these bitsets can be accomplished in one single pass through the database. The rest of the algorithm can operate on the generated data structures and does not need further database passes. The memory consumption of the bitsets is given by $n \cdot m$ bits for n instances in the database and m selectors.

	Positives Bitset										Negatives Bitset				
A	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0
B	0	0	0	1	1	1	1	1	1	1	0	0	1	1	1
C	0	0	0	1	1	1	0	0	0	0	0	0	1	1	0
D	0	0	0	1	1	1	0	0	0	1	1	0	1	0	0
E	0	0	0	1	1	1	0	0	1	0	0	0	1	0	0

Figure 1: Example bitsets: C is constructed by logical AND from A and B. C is not irrelevant to D, but irrelevant to E

After the initialization, the main recursive step is called with no conditioned selectors and all selectors possibly relevant for the search. In the current positive and negative bitsets all bits are initially set to true. The main step is shown in algorithm 1. It consists of two phases. During phase one (lines 1 to 17) all possible refinements using the possibly relevant selectors sel_{rel} are considered. For each of these selectors the subgroup description, that consists of the current conditioned selectors sel_{cond} and this new selector s_{curr}, is evaluated. Therefore, first the bitset representation $c_{currPos}$ of all positive labeled cases, which fulfill this new description, is computed by performing a logical AND operation on the current positives bitset and the positives bitset of the new selector (line 3). E.g., the left part of the row A from our example table describes the positive cases for the current conditioned selectors and the left part of row B the positive cases for the selector s_{curr}. Then the left part of row C (computed by a logical AND of these bitsets) describes the cases, that fullfil all current selectors as well as the currently evaluated selector s_{curr}. The cardinality of this bitset, i.e., the count of ones, determines the number of positive labeled cases tp for the new subgroup description, therefore in our example $tp = 3$. As the best possible refinement for a quality function q_a is always given by the subset, that contains exactly the positive cases, this count is sufficient to compute an optimistic estimate for the current combination of selectors (see above). We use this estimate in two ways:

First, only if the optimistic estimate is high enough, the negatives bitset for the *current* selector combination is computed. Then, using the counts of both, the positives and the negatives bitset, the quality for this subgroup description, i.e., the current combination of selectors can be computed. In doing so, the negative cases only have to be considered for promising selector combinations, leading to a significant speedup, especially if the target concept is the minority class. If the computed quality for this selector combination is high enough, a relevancy check (see below) for this subgroup is performed. If successful, the current description is added to the globally defined result set, potentially replacing the description with the lowest quality.

Second, similar to DpSubgroup, only if this optimistic estimate indicates, that there may be refinements of the current subgroup with the new selector, that have a sufficient quality to be included in the overall result, then this selector is added to the list of relevant selectors for the next level of search (line 8), thus substantially reducing the considered search space.

In a relevancy check, it is tested, if a subgroup s is irrelevant with respect to another subgroup s'. If the subgroup is considered relevant, it is stored in the result set, together with the bitset representations of the positive and negative cases of the subgroup (line 14). Additionally, if any subgroup already contained in the result set is irrelevant to the new subgroup, then this subgroup is removed from the result (line 15). These tests can be accomplished using the stored bitset representations: A representation of all cases with a positive target concept, that are contained in s, but not in s' can be computed by performing a logical AND-NOT between the positives bitset of s and s'. Analogously, a bitset reflecting all cases labeled as negative, that are contained in s' and not in s can be computed by another AND-NOT operation between the respective bitsets of negative cases. If both bitsets resulting from these operations do not contain any set bits, then s is irrelevant with respect to s'.

E.g, row C of our example is about to be added to the result set, that currently contains the bitsets D, E. For a relevancy check, first bitset D is considered. This bitset contains one positive case (the last bit of the positives), but also one negative case (the first bit of the negatives), that is not contained by C. Therefore, whether C is irrelevant with respect to D, nor D is irrelevant with respect to C. Next, we consider the second entry of the result set, E. As all set bits of C in the positives bitset are also set in the positives set of E, but there is no set bit in the negatives bitset of E, that is not set in the negatives of C as well, C is considered irrelevant with respect to E. Thus, C will not be added to the result.

At the start of phase two, the list of relevant selectors is sorted by their computed optimistic estimate (line 18). By doing so, more promising paths in the search space are evaluated first, so more branches of the search tree can be pruned earlier. Afterwards for each relevant selector this selector is added to the list of conditioned selectors and the main procedure of the algorithm is called recursively using the respective bitset and the conditioned selectors.

Distributed BSD

The algorithm shown above is a sequential, non-parallelized algorithm. However, most CPUs that are in use today include several cores, which can process several computation tasks simultaneously. Therefore, we present an adaptation of the BSD algorithm, that can distribute the mining tasks on all available cores. For the subgroup discovery task, partitioning of the case base is difficult, as subgroups can be distributed arbitrarily in the case base and even very small subgroups can be of high interest. Thus, our approach is based on partitioning the *search space* in the following way: In the second phase of the BSD algorithm, in which refinements of the candidate subgroup with a sufficient optimistic estimate are mined recursively, whenever the mining process is about to invoke a recursive call, this call is assigned to any idle processing unit, if available. Therefore, a branch of the search tree is assigned to this processing unit. By doing so, whenever there exist free processing resources, the current process shares parts of its own workload with others. As slower or slowed down processing units need more time to finish their current task, they will be idle less often and thus less subtasks will be given to such units. Thus, the load of the different available processing units will be automatically balanced. To avoid simultaneous modifications of the result set operations on this set must be synchronized.

Another issue is the detection of program termination. Due to the limited space, we refer to (Mattern 1987) for further information.

Related Work

For the subgroup discovery task several algorithms have already been proposed, e.g., heuristic methods like CN2-SD (Lavrac et al. 2004), and also exhaustive mining approaches, e.g., SD-Map (Atzmueller and Puppe 2006). As an improvement, exhaustive search algorithms were proposed recently, that incorporate pruning using tight optimistic estimates in order to remove parts of the search space, e.g., SD-Map* (Atzmueller and Lemmerich 2009) or DpSubgroup (Grosskreutz, Rüping, and Wrobel 2008). The general structure of the DpSubgroup algorithm shares some similarity with the BSD algorithm, however, the underlying data representation is completely different: DpSubgroup is based on FP-Trees while BSD relies on a vertical, bitset based data structure.

Vertical data representations have been proposed for other data mining tasks, e.g., by (Zaki 1998). (Burdick, Calimlim, and Gehrke 2001) used a bitset representation for maximal frequent itemset mining.

Distributed algorithms in the context of frequent set mining have been discussed, e.g., by (Aouad, Le-Khac, and Kechadi 2007) or (Zaiane, El-Hajj, and Lu 2001). A distributed approach specialized for subgroup mining has been described by (Wurst and Scholz 2006). In contrast to these (fully) distributed approaches, the proposed method uses shared memory for better scalability and reduced communication costs. The presented distributed approach provides for a natural extension of the presented method focused on the discovery of relevant subgroups to a distributed setting.

The issue of relevancy is a very important topic in machine learning and data mining. Lately, (Lavrac and Gamberger 2006) apply relevancy filtering for feature extraction/filtering. This approach was later extended by (Garriga, Kralj, and Lavrač 2008), covering closed itemsets and putting relevancy in context with supervised learning techniques. (Cheng et al. 2008) use a sequential coverage approach for mining local patterns relevant for classification, however the discovered patterns do not follow the formal relevancy criterias.

The approach proposed in this paper provides both the efficient and effective discovery of relevant subgroup patterns.

To the best of the authors' knowledge, up to now no exhaustive specialized algorithm for subgroup discovery has been proposed, that especially focuses on the mining of relevant subgroups.

Evaluation

This section presents several empirical observations on the described concepts and methods using data sets from the UCI-repository, real world data and synthetic evaluation data. The evaluation was performed with a variety of data sets and different search depths. From the UCI-repository of machine learning we used the datasets '*credit-g*', '*soybean*'and '*mushroom*'. Further, the approach was tested with a large real world dataset. This dataset describes spammers in a social network tagging system and consists of more than 17.000 cases. Additionally, we used evaluation data generated from a bayesian network about vehicle insurance, that consists of 100.000 cases. As quality functions for our experiments, we chose the Piatetsky-Shapiro function q_{PS} and relative gain q_{RG}, as they have most diverse properties with respect to pruning of the search space.

Post-processing

First, we evaluated the impact of the post-processing approach to the task of finding relevant subgroup patterns. To estimate the applicability of this method we utilized an exhaustive subgroup discovery algorithm using the quality function q_{PS} on several data sets; we investigated, how many subgroups remained in the result set after the filtering process. The results are shown in Figure 2.

Dataset / Max. Depth	2	3	4	5	6
mushroom	49	71	86	89	90
soybean	56	79	89	96	96
credit-g	5	8	10	19	19
spammer	19	62	81	88	88

Figure 2: Number of irrelevant rules after an exhaustive search for the 100 best rules, using different datasets and maximum depth boundaries.

It is easy to see, that the number of irrelevant rules in the result set is difficult to estimate a-priori, i.e., there are no guarantees with respect to the size of the result set. In order to obtain a minimal size of the result set after the filtering process has been applied, a huge number of saved subgroup descriptions would be needed. This indicates, that for many problems filtering irrelevant subgroups in a post processing step is not suitable and thus stresses the need of including a relevancy filtering step into the search algorithm.

BSD Algorithm

In the next part of our evaluation we compared the runtimes of the BSD algorithm with the state-of-the art exhaustive subgroup discovery algorithm SD-Map* with and without filtering of irrelevant subgroups. SD-Map* is based on FP-trees and also employs optimistic estimate pruning. For the tests including a relevancy check naive filtering during search was used for SD-Map*, thus both algorithms return the same set of subgroups. The evaluation was done using different datasets, and a maximum search depth of five. The results are shown in Figure 3.

Dataset	Rel. check	BSD q_{ps}	SD-Map* q_{ps}	BSD q_{RG}	SD-Map* q_{RG}
vehicle	No	13.79	33.82	13.60	47.96
	Yes	14.06	122.99	14.74	77.90
spammer	No	0.10	0.25	0.22	0.29
	Yes	0.16	314.09	0.33	379.44
credit-g	No	0.03	0.11	0.22	1.90
	Yes	0.05	366.29	0.19	329.82

Figure 3: Runtimes (in s) of *BSD* and *SD-Map** algorithms with and without relevancy-checks using different data sets (with a maximum search depth of 5).

	Without Relevancy Check		With Relevancy Check	
Depth	BSD	SD-Map*	BSD	SD-Map*
2	1.05	9.26	1.06	21.62
3	1.18	15.81	1.46	35.61
4	2.82	26.80	3.11	69.96
5	13.79	33.11	14.06	123.20
6	46.26	36.06	46.35	176.09

Figure 4: Runtimes (in s) of *BSD* and *SD-Map** algorithms for different search depths in the vehicle dataset using the q_{PS} quality function with and without relevancy checks.

For the performance tests including a relevancy check, SD-Map* with filtering of irrelevant subgroups needs orders of magnitude more time than without filtering. In contrast, as the BSD-algorithm seemlessly integrates the detection of irrelevant patterns in the mining process, it shows only a very slight runtime increase. Thus, in many tests the overall performance of BSD is more than an order of magnitude faster than the SD-Map* algorithm, if an online relevancy check is enabled. Additionally, even without relevancy checking the BSD algorithm showed runtimes comparable to SD-Map*, being slightly faster in all tested cases.

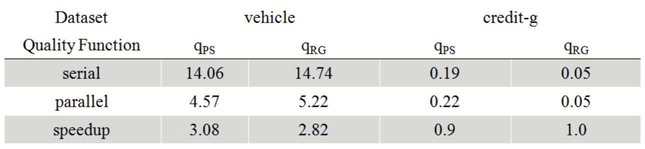
In another evaluation we examined the runtimes of the algorithms for different maximum search depths. The results are shown in Figure 4. It can be observed, that the BSD algorithm especially excels at lower maximum search depth, but shows a sharper increase of runtime at larger depths. Thus, for tasks without relevancy check, BSD is outperformed by SD-Map* at large maximum depths. We explain this by the more elaborated pruning capabilities in SD-Map* utilizing its more complex data representation, i.e., FP-Trees. However, since many of the interesting subgroup descriptions tend to be short, a maximum depth of about 5 or 6 can be considered as sufficient for most practical problems. Of course, for the tests with online filtering of irrelevant subgroups the BSD algorithm also shows an increase in runtimes, while it still outperforms SD-Map*.

Therefore, we regard the BSD algorithm as the best choice when mining for relevant subgroups and propose to consider it also for subgroup discovery without relevancy checks for low and medium search depths.

Distributed BSD Algorithm

The benefit of the distributed BSD algorithm was evaluated by a comparison with the serial variant of the algorithm with the parallelized version using 4 CPU cores. Two exemplary results are shown in Figure 5.

In general, for most tasks, e.g., the vehicle dataset, a large performance gain is shown, reaching a speedup factor of up to 3. For tasks with very small runtimes, e.g., the credit-g dataset, the administrative overhead for organizing the threads exceeds the gains of multiple processing units. However, the BSD algorithms runtime is very short (less than 1 second) in these cases anyway.



Dataset	vehicle		credit-g	
Quality Function	q_{PS}	q_{RG}	q_{PS}	q_{RG}
serial	14.06	14.74	0.19	0.05
parallel	4.57	5.22	0.22	0.05
speedup	3.08	2.82	0.9	1.0

Figure 5: Runtimes (in s) of BSD using one (serial) and four processing units (parallel). As quality functions q_{PS} and q_{RG} were used. The maximum search depth was set to 5, online relevancy-checks were applied.

Conclusions

In this paper, we have presented a novel subgroup discovery algorithm, that seemlessly integrates filtering of *irrelevant* subgroups into the mining algorithm by utilizing a specialized vertical data structure. By using efficient pruning strategies the algorithm outperforms the state-of-the-art algorithms when mining with such a relevancy check and also showed to be efficient for tasks without relevancy filtering, especially at low search depths. Additionally, we presented how the algorithm can be distributed on several processing units.

For future work, we plan to extend the distributed algorithm in a grid-like environment. Furthermore, suitable visualization techniques for inspecting the set of relevant and the (suppressed) set of irrelevant patterns are additional interesting research directions; in that respect, clustering techniques seem to be a viable option for inclusion.

Acknowledgements

This work has been partially supported by the German Research Council (DFG) under grant Pu 129/8-2.

References

Aouad, L. M.; Le-Khac, N.-A.; and Kechadi, T. M. 2007. Grid-based approach for distributed frequent itemsets mining using dynamic workload management. In *Proc. 2nd International Conference on Advances in Information Technology (IAIT2007)*.

Atzmueller, M., and Lemmerich, F. 2009. Fast Subgroup Discovery for Continuous Target Concepts. In *Proc. 18th International Symposium on Methodologies for Intelligent Systems (ISMIS 2009)*, LNCS.

Atzmueller, M., and Puppe, F. 2006. SD-Map – A Fast Algorithm for Exhaustive Subgroup Discovery. In *Proc. 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006)*, number 4213 in LNAI, 6–17. Berlin: Springer Verlag.

Burdick, D.; Calimlim, M.; and Gehrke, J. 2001. MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. In *Proc. 17th International Conference on Data Engineering (ICDE'01)*, 443–452.

Cheng, H.; Yan, X.; Han, J.; and Yu, P. S. 2008. Direct discriminative pattern mining for effective classification. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, 169–178. Washington, DC, USA: IEEE Computer Society.

Garriga, G. C.; Kralj, P.; and Lavrač, N. 2008. Closed Sets for Labeled Data. *J. Mach. Learn. Res.* 9:559–580.

Grosskreutz, H.; Rüping, S.; and Wrobel, S. 2008. Tight Optimistic Estimates for Fast Subgroup Discovery. In *Proceedings of the 2008 ECML-PKDD - Part I*, 440–456. Berlin: Springer Verlag.

Klösgen, W. 1996. Explora: A Multipattern and Multistrategy Discovery Assistant. In Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R., eds., *Advances in Knowledge Discovery and Data Mining*. AAAI Press. 249–271.

Lavrac, N., and Gamberger, D. 2006. Relevancy in Constraint-based Subgroup Discovery. In Jean-Francois Boulicaut, Luc de Raedt, H. M., ed., *Constraint-based Mining and Inductive Databases*, volume 3848 of *LNCS*. Berlin: Springer Verlag.

Lavrac, N.; Kavsek, B.; Flach, P.; and Todorovski, L. 2004. Subgroup Discovery with CN2-SD. *Journal of Machine Learning Research* 5:153–188.

Mattern, F. 1987. Algorithms for Distributed Termination Detection. *Distributed Computing* 2(3):161–175.

Newman, D.; Hettich, S.; Blake, C.; and Merz, C. 1998. UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/∼mlearn/mlrepository.html.

Wrobel, S. 1997. An Algorithm for Multi-Relational Discovery of Subgroups. In *Proc. 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, 78–87. Berlin: Springer Verlag.

Wurst, M., and Scholz, M. 2006. Distributed subgroup mining. In Fürnkranz, J.; Scheffer, T.; and Spiliopoulou, M., eds., *Proc. PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 4213 of *Lecture Notes in Computer Science*, 421–433. Springer.

Zaiane, O. R.; El-Hajj, M.; and Lu, P. 2001. Fast parallel association rule mining without candidacy generation. In *Proc. 2001 IEEE Int. Conf. on Data Mining*, 665–668.

Zaki, M. J. 1998. Efficient Enumeration of Frequent Sequences. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, 68–75. New York, NY, USA: ACM.