SD-Map – A Fast Algorithm for Exhaustive Subgroup Discovery

Martin Atzmueller and Frank Puppe

University of Würzburg, 97074 Würzburg, Germany
Department of Computer Science
Tel.: +49 931 888-6739; Fax: +49 931 888-6732
{atzmueller, puppe}@informatik.uni-wuerzburg.de

Abstract. In this paper we present the novel SD-Map algorithm for exhaustive but efficient subgroup discovery. SD-Map guarantees to identify all interesting subgroup patterns contained in a data set, in contrast to heuristic or sampling-based methods. The SD-Map algorithm utilizes the well-known FP-growth method for mining association rules with adaptations for the subgroup discovery task. We show how SD-Map can handle missing values, and provide an experimental evaluation of the performance of the algorithm using synthetic data.

1 Introduction

Subgroup discovery [1,2,3,4] is a powerful and broadly applicable method that focuses on discovering interesting subgroups of individuals. It is usually applied for data exploration and descriptive induction, in order to identify relations between a dependent (target) variable and usually many independent variables, e.g., ”the subgroup of 16-25 year old men that own a sports car are more likely to pay high insurance rates than the people in the general population”, but can also be used for classification (e.g., [5]).

Due to the exponential search space commonly heuristic (e.g., [2,6]) or sampling approaches (e.g., [7]) are applied in order to discover the k best subgroups. However, these do not guarantee the discovery of all the optimal (most interesting) patterns. For example, if a heuristic method implements a greedy approach or uses certain pruning heuristics, then whole subspaces of the search space are not considered. Furthermore, heuristic approaches usually have ”blind spots” that prevent certain patterns from being discovered: Considering beam search for example, if only a combination of factors is interesting, where each single factor is not, then the combination might not be identified [2]. In consequence, often not all interesting patterns can be discovered by the heuristic methods, and only estimates about the best k subgroups are obtained. This can be a problem especially for explorative and descriptive methods since some of the truly interesting patterns cannot be identified. In contrast to heuristic methods exhaustive approaches guarantee to discover the best solutions. However, then the runtime costs of an exhaustive algorithm usually prohibit its application for larger search spaces.

In this paper we propose the novel *SD-Map* algorithm for fast and exhaustive subgroup discovery, based upon the FP-growth algorithm [8] for mining association rules with adaptations for the subgroup discovery task. We show how SD-Map can efficiently

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 6–17, 2006.

handle missing values that are common for some domains, and also how the algorithm can be applied for more complex description languages using internal disjunctions. Additionally, we provide an experimental evaluation of the algorithm using synthetic data.

The rest of the paper is organized as follows: We introduce the subgroup discovery setting in Section 2, and present the SD-Map algorithm in Section 3. After that, an experimental evaluation of the algorithm using synthetic data is provided in Section 4. Finally, we conclude the paper with a discussion of the presented work, and we show promising directions for future work in Section 5.

2 Subgroup Discovery

The main application areas of subgroup discovery are exploration and descriptive induction, to obtain an overview of the relations between a target variable and a set of explaining variables. Then, not necessarily complete relations but also partial relations, i.e., (small) subgroups with "interesting" characteristics can be sufficient.

Before describing the subgroup discovery setting, we first introduce the necessary notions concerning the used knowledge representation: Let Ω_A be the set of all attributes. For each attribute $a \in \Omega_A$ a range $dom(a)$ of values is defined. Furthermore, we assume $\mathcal{V}_A$ to be the (universal) set of attribute values of the form $(a = v)$, where $a \in \Omega_A$ is an attribute and $v \in dom(a)$ is an assignable value. We consider nominal attributes only so that numeric attributes need to be discretized accordingly. Let *CB* be the case base (data set) containing all available cases, also often called instances. A case $c \in CB$ is given by the n-tuple $c = ((a_1 = v_1), (a_2 = v_2), \dots, (a_n = v_n))$ of $n = |\Omega_A|$ attribute values, where $v_i \in dom(a_i)$ for each a_i.

A subgroup discovery setting mainly relies on the following four main properties: the target variable, the subgroup description language, the quality function, and the search strategy. In this paper we focus on binary target variables. Similar to the *MIDOS* approach [3] we try to identify subgroups that are, e.g., as large as possible, and have the most unusual (distributional) characteristics with respect to the concept of interest given by the target variable.

The description language specifies the individuals belonging to the subgroup. In the case of single-relational propositional languages a subgroup description can be defined as follows:

Definition 1 (Subgroup Description). *A subgroup description* $sd = \{e_1, e_2, \dots, e_n\}$ *is defined by the conjunction of a set of selection expressions. These 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_{sd} *as the set of all possible subgroup descriptions.*

A quality function measures the interestingness of the subgroup mainly based on a statistical evaluation function, e.g., the chi-squared statistical test. It is used by the search method to rank the discovered subgroups during search. Typical quality criteria include the difference in the distribution of the target variable concerning the subgroup and the general population, and the subgroup size. We assume that the user specifies a minimum support threshold $\mathcal{T}_{Supp}$ corresponding to the number of target class cases contained in the subgroup in order to prune very small and therefore uninteresting subgroup patterns.

Definition 2 (Quality Function). *Given a particular target variable $t \in \mathcal{V}_A$, a quality function $q : \Omega_{sd} \times \mathcal{V}_A \rightarrow R$ is used in order to evaluate a subgroup description $sd \in \Omega_{sd}$, and to rank the discovered subgroups during search.*

Several quality functions were proposed, e.g., [1,2,7]. For binary target variables, examples for quality functions are given by

$$q_{BT} = \frac{(p - p_0) \cdot \sqrt{n}}{\sqrt{p_0 \cdot (1 - p_0)}} \cdot \sqrt{\frac{N}{N - n}}\,, \quad q_{RG} = \frac{p - p_0}{p_0 \cdot (1 - p_0)}\,,$$

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, $N = |CB|$ is the size of the total population, and n denotes the size of the subgroup.

Considering the subgroup search strategy an efficient search is necessary, since the search space is exponential concerning all the possible selectors of a subgroup description. In the next section, we propose the novel SD-Map algorithm for exhaustive subgroup discovery based on the *FP-growth* [8] method.

3 SD-Map

SD-Map is an exhaustive search method, dependent on a minimum support threshold: If we set the minimum support to zero, then the algorithm performs an exhaustive search covering the whole unpruned search space. We first introduce the basics of the FP-growth method, referring to Han et al. [8] for a detailed description. We then discuss the SD-Map algorithm and describe the extensions and adaptations of the FP-growth method for the subgroup discovery task. After that, we describe how SD-Map can be applied efficiently for the special case of (strictly) conjunctive languages using internal disjunctions, and discuss related work.

3.1 The Basic FP-Growth Algorithm

The FP-growth [8] algorithm is an efficient approach for mining frequent patterns. Similar to the Apriori algorithm [9], FP-growth operates on a set of items which are given by a set of selectors in our context. The main improvement of the FP-growth method compared to Apriori is the feature of avoiding multiple scans of the database for testing each frequent pattern. Instead, a recursive divide-and-conquer technique is applied.

As a special data structure, the frequent pattern tree or FP-tree is implemented as an extended prefix-tree-structure that stores count information about the frequent patterns. Each node in the tree is a tuple (selector, count, node-link): The count measures the number of times the respective selector is contained in cases reached by this path, and the node-link links to the nodes in the FP-tree with the same assigned selector. The construction of an FP-tree only needs two passes through the set of cases: In the first pass the case base is scanned collecting the set of frequent selectors L that is sorted in support descending order. In the second pass, for each case of the case base the contained frequent selectors are inserted into the tree according to the order of L, such that the chances of sharing common prefixes (subsets of selectors) are increased. After construction, the paths of the tree contain the count information of the frequent selectors (and patterns), usually in a much more compact form than the case base.

For determining the set of frequent patterns, the FP-growth algorithm applies a divide and conquer method, first mining frequent patterns containing one selector and then recursively mining patterns conditioned on the occurrence of a (prefix) 1-selector. For the recursive step, a conditional FP-tree is constructed, given the conditional pattern base of a frequent selector of the FP-Tree and its corresponding nodes in the tree. The conditional pattern base consists of all the prefix paths of such a node v, i.e., considering all the paths P_v that node v participates in. Given the conditional pattern base, a (smaller) FP-tree is generated, the conditional FP-tree of v utilizing the adapted frequency counts of the nodes. If the conditional FP-Tree just consists of one path, then the frequent patterns can be generated by considering all the combinations of the nodes contained in the path. Otherwise, the process is performed recursively.

3.2 The SD-Map Algorithm

Compared to association rules that measure the confidence ($precision$) and the $support$ of rules [9], subgroup discovery uses a special quality function to measure the interestingness of a subgroup. A naive adaptation of the FP-growth algorithm for subgroup discovery just uses the FP-growth method to collect the frequent patterns; then we also need to test the patterns using the quality function.

For the subgroup quality computation mainly four parameters are used: The true positives tp (cases containing the target variable t in the given subgroup s), the false positives fp (cases not containing the target t in the subgroup s), and the positives TP and negatives FP regarding the target variable t in the general population of size N. Thus, we could just apply the FP-growth method as is, and compute the subgroup parameters as

- $n = count(s)$,
- $tp = support(s) = count(s \wedge t)$,
- $fp = n - tp$,
- $p = tp/(tp + fp)$, and
- $TP = count(t)$,
- $FP = N - TP$,
- $p_0 = TP/(TP + FP)$.

However, one problem encountered in data mining remains: The missing value problem. If missing values are present in the case base, then not all cases may have a defined value for each attribute, e.g., if the attribute value has not been recorded. For association rule mining this usually does not occur, since we only consider items in a transaction: An item is present or not, and never undefined or missing. In contrast missing values are a significant problem for subgroup mining in some domains, e.g., in the medical domain [4,10]. If the missing values cannot be eliminated they need to be considered in the subgroup discovery method when computing the quality of a subgroup: We basically need to adjust the counts for the population by identifying the cases where the subgroup variables or the target are not defined, i.e., where these have missing values.

So, the simple approach described above is redundant and also not sufficient, since (a) we would get a larger tree if we used a normal node for the target; (b) if there are missing values, then we need to restrict the parameters TP and FP to the cases for which all the attributes of the selectors contained in the subgroup description have defined values; (c) furthermore, if we derived $fp = n - tp$, then we could not distinguish the cases where the target is not defined.

In order to improve this situation, and to reduce the size of the constructed FP-tree, we consider the following important observations:

1. For estimating the subgroup quality we only need to determine the four basic subgroup parameters tp, fp, TP, and FP with a potential adaptation for missing values. Then, the other parameters can be derived: $n = tp + fp$, $N = TP + FP$.
2. For subgroup discovery, the concept of interest, i.e., the target variable is fixed, in contrast to the arbitrary ”rule head” of association rules. Thus, the necessary parameters described above can be directly computed, e.g., considering the true positives tp, if a case contains the target and a particular selector, then the tp count of the respective node of the FP-tree is incremented.

If the tp, fp counts are stored in the nodes of the FP-tree, then we can compute the quality of the subgroups directly while generating the frequent patterns. Furthermore, we only need to create nodes for the independent variables, not for the dependent (target) variable. In the SD-Map algorithm, we just count for each node if the target variable occurs (incrementing tp) or not (incrementing fp), restricted to cases for which the target variable has a defined value. The counts in the general population can then be acquired as a by-product.

For handling missing values we propose to construct a second FP-tree-structure, the *Missing-FP-tree*. The FP-tree for counting the missing values can be restricted to the set of frequent attributes of the main FP-Tree, since only these can form subgroup descriptions that later need to be checked with respect to missing values. Then, the Missing-FP-tree needs to be evaluated in a special way to obtain the respective missing counts. To adjust for missing values we only need to adjust the number of TP, FP corresponding to the population, since the counts for the subgroup FP-tree were obtained for cases where the target was defined, and the subgroup description only takes cases into account where the selectors are defined. Using the Missing-FP-tree, we can identify the situations where any of the attributes contained in the subgroup description is undefined: For a subgroup description $sd = (e_1 \wedge e_2 \wedge \cdots \wedge e_n), e_i = (a_i, V_i), V_i \subseteq dom(a_i)$ we need to compute the missing counts $missing(a_1 \vee a_2 \vee \cdots \vee a_n)$ for the set of attributes $M = \{a_1, \ldots, a_n\}$. This can be obtained applying the following transformation:

$$missing(a_1 \vee \cdots \vee a_n) = \sum_{i=1}^{n} missing(a_i) - \sum_{m \in 2^M} missing(m)\,, \tag{1}$$

where $|m| \geq 2$. Thus, in order to obtain the *missing adjustment* with respect to the set M containing the attributes of the subgroup, we need to add the entries of the header nodes of the Missing-FP-tree corresponding to the individual attributes, and subtract the entries of every suffix path ending in an element of M that contains at least another element of M.

Algorithm 1. SD-Map algorithm

```
Require: target variable $t$, quality function $q$, set of selectors $E$ (search space)
1: Scan 1 – Collect the frequent set of selectors, and construct the frequent node list $L$ for the main FP-tree:
    1. For each case for which the target variable has a defined value, count the $(tp_e, fp_e)$ for each selector $e \in E$.
    2. Prune all selectors which are below a minimum support $\mathcal{T}_{Supp}$, i.e., $tp(e) = frequency(e \wedge t) < \mathcal{T}_{Supp}$.
    3. Using the unpruned selectors $e \in E$, construct and sort the frequent node list $L$ in support/frequency descending order.
2: Scan 2 – Build the main FP-tree:
   For each node contained in the frequent node list, insert the node into the FP-tree (according to the order of $L$), if observed in a case, and count the number of $(tp, fp)$ for each node
3: Scan 3 – Build the Missing-FP-tree (This step can also be integrated as a logical step into scan 2):
    1. For all frequent attributes, i.e., the attributes contained in the frequent nodes $L$ of the FP-tree, generate a node denoting the missing value for this attribute.
    2. Then, construct the Missing-FP-tree, counting the $(tp, fp)$ for each (attribute) node.
4: Perform the adapted FP-growth method to generate subgroup patterns:
5: repeat
6:    for each subgroup $s_i$ that denotes a frequent subgroup pattern do
7:       Compute the adjusted population counts $TP', FP'$ as shown in Equation 2
8:       Given the parameters $tp, fp, TP', FP'$ compute the subgroup quality $q(s_i)$ using a quality function $q$
9: until FP-growth is finished
10: Post-process the set of the obtained subgroup patterns, e.g., return the $k$ best subgroups, or return the subgroup set $S = \{s \mid q(s) \geq q_{min}\}$, for a minimal quality threshold $q_{min} \in \mathbb{R}$ (This step can also be integrated as a logical filtering step in the discovery loop (lines 5-9)).
```

By considering the (tp, fp) counts contained in the Missing-FP-tree we obtain the number of $(TP_{missing}, FP_{missing})$ where the cases cannot be evaluated statistically since any of the subgroup variables are not defined, i.e., at least one attribute contained in M is missing. To adjust the counts of the population, we compute the correct counts as follows:

$$TP' = TP - TP_{missing}, FP' = FP - FP_{missing} \,. \tag{2}$$

Then, we can compute the subgroup quality based on the parameters tp, fp, TP', FP'. Thus, in contrast to the standard FP-growth method, we do not only compute the frequent patterns in the FP-growth algorithm, but we can also directly compute the quality of the frequent subgroup patterns, since all the parameters can be obtained in the FP-growth step. So, we perform an integrated grow-and-test step, accumulating the subgroups directly. The SD-Map algorithm is shown in Algorithm 1.

SD-Map includes a post-processing step for the selection and the potential redundancy management of the obtained set of subgroups. Usually, the user is interested only in the best k subgroups and does not want to inspect all the discovered subgroups. Thus, we could just select the best k subgroups according to the quality function and return these. Alternatively, we could also choose all the subgroups above a minimum quality threshold. Furthermore, in order to reduce overlapping and thus potentially redundant subgroups, we can apply post-processing, e.g., clustering methods or a weighted covering approach (e.g., [11]), as in the Apriori-SD [12] algorithm. The post-processing step could potentially also be integrated into the ”discovery loop” (while applying the adapted FP-growth method), see Algorithm 1.

A subgroup description as defined in Section 2 can either contain selectors with internal disjunctions, or not. Using a subgroup description language without internal disjunctions is often sufficient for many domains, e.g., for the medical domain [4,10]. In this case the description language matches the setting of the common association rule mining methods. If internal disjunctions are not required in the subgroup descriptions, then the SD-Map method can be applied in a straight-forward manner: If we construct a selector $e = (a, \{v_i\})$ for each value v_i of an attribute a, then the SD-Map method just derives the desired subgroup descriptions: Since the selectors do not overlap, a conjunction of selectors for the same attribute results in an empty set of covered cases. Thus, only conjunctions of selectors will be regarded as interesting if these correspond to a disjoint set of attributes. Furthermore, each path of a constructed frequent pattern tree will also only contain a set of selectors belonging to a disjoint set of attributes.

If internal disjunctions are required, then the search space is significantly enlarged in general, since there are $2^m - 1$ (non-empty) value combinations for an attribute with m values. However, the algorithm can also be applied efficiently for the special case of a description language using selectors with internal disjunctions. In the following section we describe two approaches for handling that situation.

3.3 Applying SD-Map for Subgroup Descriptions with Internal Disjunctions Efficiently

Naive Approach. First, we can just consider all possible selectors with internal disjunctions for a particular attribute. This technique can also be applied if not all internal disjunctions are required, and if only a selection of aggregated values should be used. For example, a subset of the value combinations can be defined by a taxonomy, by the ordinality of an attribute, or by background knowledge, e.g., [4].

If all possible disjunctions need to be considered, then it is easy to see that adding the selectors corresponding to all internal disjunctions significantly extends the set of selectors that are represented in the paths of the tree compared to a subgroup description language without internal disjunctions. Additionally, the selectors overlap: Then, the sizes of many conditional trees being constructed during the mining phase is increased. However, in order to improve the efficiency of this approach we can apply the following pruning techniques:

1. During construction of a conditional tree, we can prune all parent nodes that contain the same attribute as the conditioning node, since these are subsumed by the conditioning selector.
2. When constructing the combinations of the selectors on a single path of the FP-tree we only need to consider the combinations for disjoint sets of the corresponding attributes.

Using Negated Selectors. An alternative approach is especially suitable if all internal disjunctions of an attribute are required. The key idea is to express an internal disjunction by a conjunction of negated selectors: For example, consider the attribute a with the values v_1, v_2, v_3, v_4; instead of the disjunctive selector $(a, \{v_1, v_2\})$ corresponding to the value $v_1 \vee v_2$ we can utilize the conjunctive expression $\neg(a, \{v_3\}) \wedge \neg(a, \{v_4\})$ corresponding to $\neg v_3 \wedge \neg v_4$.

This works for data sets that do not contain missing values. If missing values are present, then we just need to exclude the missing values for such a negated selector: For the negated selectors of the example above, we create $\neg(a, \{v_3, v_{missing}\}) \wedge \neg(a, \{v_4, v_{missing}\})$. Then, the missing values are not counted for the negated selectors. If we need to consider the disjunction of the all attribute values, then we add a selector $\neg(a, \{v_{missing}\})$ for each attribute a. Applying this approach, we only need to create negated selectors for each attribute value of an attribute, instead of adding all internal disjunctions: The set of selectors that are contained in the frequent pattern tree is then significantly reduced compared to the naive approach described above. The evaluation of the negated selectors can be performed without any modifications of the algorithm. Before the subgroup patterns are returned, they are transformed to subgroup descriptions without negation by merging and replacing the negated selectors by semantically equivalent selectors containing internal disjunctions.

3.4 Discussion

Using association rules for classification has already been proposed, e.g., in the *CBA* [13], the *CorClass* [14], and the *Apriori-C* [15] algorithms. Based upon the latter, the Apriori-SD [12] algorithm is an adaptation for the subgroup discovery task. Applying the notion of *class association rules*, all of the mentioned algorithms focus on association rules containing exactly one selector in the rule head.

Compared to the existing approaches, we use the exhaustive FP-growth method that is usually faster than the Apriori approach. To the best of the authors' knowledge, it is the first time that an (adapted) FP-growth method has been applied for subgroup discovery. The adaptations of the Apriori-style methods are also valid for the FP-growth method. Then, due the subgroup discovery setting the memory and runtime complexity of FP-growth can usually be reduced (c.f., [15,12]). In comparison to the Apriori-based methods and a naive application of the FP-growth algorithm, the SD-Map method utilizes a modified FP-growth step that can compute the subgroup quality directly without referring to other intermediate results. If the subgroup selection step is included as a logical filtering step, then the memory complexity can also be decreased even further.

Moreover, none of the existing algorithms tackles the problem of missing values explicitly. Therefore, we propose an efficient integrated method for handling missing values. Such an approach is essential for data sets containing very many missing values. Without applying such a technique, an efficient evaluation of the quality of a subgroup is not possible, since either too many subgroups would need to be considered in a post-processing step, or subgroups with a high quality might be erroneously excluded.

Furthermore, in contrast to algorithms that apply branch-and-bound techniques requiring special (convex) quality functions, e.g., the Cluster-Grouping [16] and the CorClass [14] algorithms, SD-Map can utilize arbitrary quality functions.

4 Evaluation

In our evaluation we apply a data generator for synthetic data that uses a Bayesian network as its knowledge representation. The prior-sample algorithm is then applied in order to generate data sets with the same characteristics, but different sizes. The Bayesian network used for data generation is shown in Figure 1. It models an artificial vehicle insurance domain containing 15 attributes with a mean of 3 attribute values.

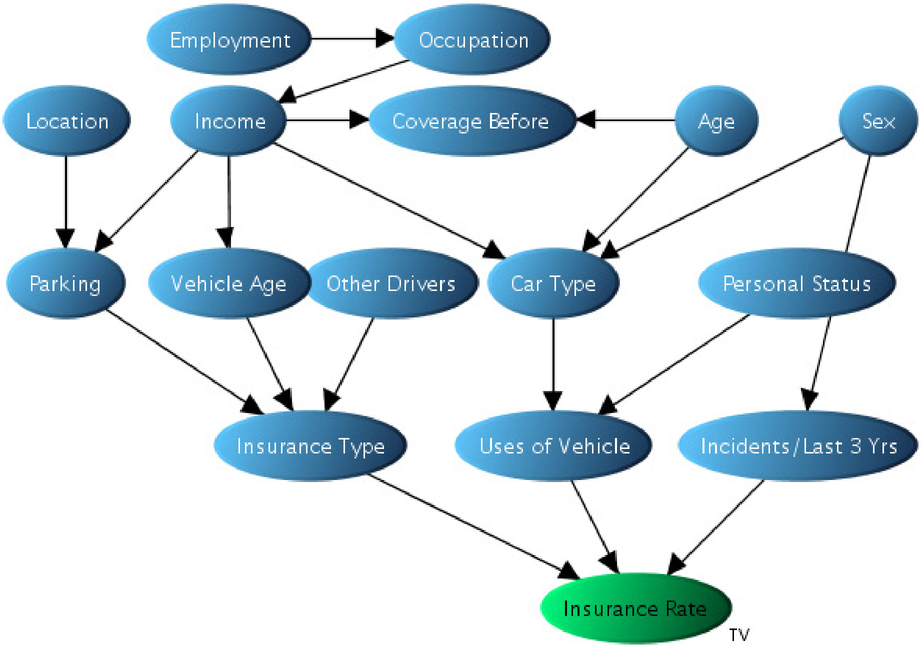

Fig. 1. Artificial vehicle insurance domain: Basic Bayesian network used for generating the synthetic evaluation data

The conditional probability tables of the network were initialized with uniform value distributions, i.e., each value in the table of an unconditioned node is equiprobable, and the conditional entries of a conditioned node are also defined by a uniform distribution.

In the experiments described below we compare three subgroup discovery algorithms: a standard beam search algorithm [2], the Apriori-SD method [12] and the proposed novel SD-Map algorithm. The beam search method and the Apriori-SD approach were implemented to the best of our knowledge based on the descriptions in the literature. All the experiments are performed on a 3.0 GHZ Pentium 4 PC machine running

Windows XP, providing a maximum of 1GB main memory for each experiment (the size of the main memory was never a limiting factor in the experiments).

4.1 Results

We generated data sets containing 1000, 10000 and 100000 instances. In each of these, there are exponentially numerous frequent attribute value combinations as the support threshold is reduced, considering the total search space of about 10^{10} patterns with respect to all combinations of the attribute values without internal disjunctions.

We measured the run-times of the algorithms by averaging five runs for each algorithm for each of the test data sets. For the beam search method we applied a quite low beam width ($w = 10$) in order to compare a fast beam search approach to the other methods. However, since beam search is a heuristic discovery method it could never discover all of the interesting subgroups compared to the exhaustive SD-Map and Apriori-SD algorithms in our experiments.

Since the minimal support threshold is the parameter that is common to all search methods we can utilize it in order to compare their scalability: In the experiments, we used two different low support thresholds, 0.01 and 0.05, to estimate the effect of increasing the pruning threshold, and to test the performance of the algorithms concerning quite small subgroups that are nevertheless considered as interesting in some domains, e.g., in the medical domain. We furthermore vary the number of instances, and the number of attributes provided to the subgroup discovery method in order to study the effect of a restricted search space compared to a larger one. Thus, by initially utilizing 5, then 10 and finally 15 attributes, an exponential increase in the search space for a fixed data set could be simulated. We thus performed 18 experiments for each algorithm in total, for each description language variant. The results are shown in the tables in Figure 2. Due to the limited space we only include the results for the 0.05 minimum support level with respect to the description language using selectors with internal disjunctions.

4.2 Discussion

The results of the experiments show that the SD-Map algorithm clearly outperforms the other methods. All methods perform linearly when the number of cases/instances is increased. However, the SD-Map method is at least faster by one magnitude compared to beam search, and about two magnitudes faster than Apriori-SD.

When the search space is increased exponentially all search method scale proportionally. In this case, the SD-Map method is still significantly faster than the beam search method, even if SD-Map needs to handle an exponential increase in the size of the search space. The SD-Map method also shows significantly better scalability than the Apriori-SD method for which the runtime grows exponentially for an exponential increase in the search space (number of attributes). In contrast, the run time of the SD-Map method grows in a much more conservative way. This is due to the fact that Apriori-SD applies the candidate-generation and test strategy depending on multiple scans of the data set and thus on the size of the data set. SD-Map benefits from its divide-and-conquer approach adapted from the FP-growth method by avoiding a large number of generate-and-test steps.

Runtime (sec.) - Conjunctive Description Language (No Internal Disjunctions)

MinSupport=0.01	1000 instances			10000 instances			100000 instances		
#Attributes	5	10	15	5	10	15	5	10	15
SD-Map	0.06	0.19	1.0	0.2	0.6	4.4	2.0	3.9	24.2
Beam search (w=10)	0.7	2.2	4.8	6.8	26.4	45.8	63.0	275.4	421.3
Apriori-SD	1.3	39.2	360.1	11.7	366.0	2336.1	108.7	3762.1	22128.5

Runtime (sec.) - Conjunctive Description Language (No Internal Disjunctions)

MinSupport=0.05	1000 instances			10000 instances			100000 instances		
#Attributes	5	10	15	5	10	15	5	10	15
SD-Map	0.02	0.05	0.14	0.2	0.4	1.1	2.0	3.7	9.2
Beam search (w=10)	0.5	1.8	3.1	4.6	12.2	25.7	49.4	173.0	295.8
Apriori-SD	0.3	2.3	6.0	2.4	23.2	63.7	23.7	229.5	632.6

Runtime (sec.) - Conjunctive Description Language (Internal Disjunctions)

MinSupport=0.05	1000 instances			10000 instances			100000 instances		
#Attributes	5	10	15	5	10	15	5	10	15
SD-Map (Negated S.)	0.05	0.4	8.5	0.3	1.4	38.7	2.8	6.1	144.5
SD-Map (Naive)	0.09	0.8	13.9	0.6	3.5	91.0	5.5	12.9	425.6
Beam search (w=10)	3.6	6.9	15.9	34.1	63.7	142.9	450.2	473.5	1093.6
Apriori-SD	6.9	124.3	7307.4	59.7	1000.2	23846.3	609.3	10148.3	208849

Fig. 2. Efficiency: Runtime of the algorithms for a conjunctive description language either using internal disjunctions or not

As shown in the tables in Figure 2 minimal support pruning has a significant effect on the exhaustive search methods. The run-time for SD-Map is reduced by more than half for larger search spaces, and for Apriori-SD the decrease is even more significant.

The results for the description language using internal disjunctions (shown in the third table of Figure 2) are similar to these for the non-disjunctive case. The SD-Map approach for handling internal disjunctions using negated selectors is shown in the first row, while the naive approach is shown in the second row of the table: It is easy to see that the first scales usually better than the latter. Furthermore, both outperform the beam search and Apriori-SD methods clearly.

5 Summary and Future Work

In this paper we have presented the novel SD-Map algorithm for fast but exhaustive subgroup discovery. SD-Map is based on the efficient FP-growth algorithm for mining association rules, adapted to the subgroup discovery setting. We described, how SD-Map can handle missing values, and how internal disjunctions in the subgroup description can be implemented efficiently. An experimental evaluation showed, that SD-Map provides huge performance gains compared to the Apriori-SD algorithm and even to the heuristic beam search approach.

In the future, we are planning to combine the SD-Map algorithm with sampling approaches and pruning techniques. Furthermore, integrated clustering methods for subgroup set selection are an interesting issue to consider for future work.

References

1. Klösgen, W.: Explora: A Multipattern and Multistrategy Discovery Assistant. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds.: Advances in Knowledge Discovery and Data Mining. AAAI Press (1996) 249–271
2. Klösgen, W.: 16.3: Subgroup Discovery. In: Handbook of Data Mining and Knowledge Discovery. Oxford University Press (2002)
3. Wrobel, S.: An Algorithm for Multi-Relational Discovery of Subgroups. In: Proc. 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-97), Berlin, Springer Verlag (1997) 78–87
4. Atzmueller, M., Puppe, F., Buscher, H.P.: Exploiting Background Knowledge for Knowledge-Intensive Subgroup Discovery. In: Proc. 19th Intl. Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland (2005) 647–652
5. Lavrac, N., Kavsek, B., Flach, P., Todorovski, L.: Subgroup Discovery with CN2-SD. Journal of Machine Learning Research **5** (2004) 153–188
6. Friedman, J., Fisher, N.: Bump Hunting in High-Dimensional Data. Statistics and Computing **9**(2) (1999)
7. Scheffer, T., Wrobel, S.: Finding the Most Interesting Patterns in a Database Quickly by Using Sequential Sampling. Journal of Machine Learning Research **3** (2002) 833–862
8. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns Without Candidate Generation. In Chen, W., Naughton, J., Bernstein, P.A., eds.: 2000 ACM SIGMOD Intl. Conference on Management of Data, ACM Press (2000) 1–12
9. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: Proc. 20th Int. Conf. Very Large Data Bases, (VLDB), Morgan Kaufmann (1994) 487–499
10. Atzmueller, M., Puppe, F., Buscher, H.P.: Profiling Examiners using Intelligent Subgroup Mining. In: Proc. 10th Intl. Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2005), Aberdeen, Scotland (2005) 46–51
11. Kavsek, B., Lavrac, N., Todorovski, L.: ROC Analysis of Example Weighting in Subgroup Discovery. In: Proc. 1st Intl. Workshop on ROC Analysis in AI, Valencia, Spain (2004) 55–60
12. Kavsek, B., Lavrac, N., Jovanoski, V.: APRIORI-SD: Adapting Association Rule Learning to Subgroup Discovery. In: Proc. 5th Intl. Symposium on Intelligent Data Analysis, Springer Verlag (2003) 230–241
13. Liu, B., Hsu, W., Ma, Y.: Integrating Classification and Association Rule Mining. In: Proc. 4th Intl. Conference on Knowledge Discovery and Data Mining, New York, USA (1998) 80–86
14. Zimmermann, A., Raedt, L.D.: CorClass: Correlated Association Rule Mining for Classification. In Suzuki, E., Arikawa, S., eds.: Proc. 7th Intl. Conference on Discovery Science. Volume 3245 of Lecture Notes in Computer Science. (2004) 60–72
15. Jovanoski, V., Lavrac, N.: Classification Rule Learning with APRIORI-C. In: EPIA '01: Proc. 10th Portuguese Conference on Artificial Intelligence, London, UK, Springer-Verlag (2001) 44–51
16. Zimmermann, A., Raedt, L.D.: Cluster-Grouping: From Subgroup Discovery to Clustering. In: Proc. 15th European Conference on Machine Learning (ECML04). (2004) 575–577