Inductive Querying for Discovering Subgroups and Clusters*

Albrecht Zimmermann and Luc De Raedt

Chair of Machine Learning, Institute of Computer Science,
Albert-Ludwigs-University, Freiburg, Georges-Köhler-Allee 079,
79110 Freiburg, Germany
{azimmerm, deraedt}@informatik.uni-freiburg.de

Abstract. We introduce the problem of **cluster-grouping** and show that it integrates several important data mining tasks, i.e. subgroup discovery, mining correlated patterns and aspects from clustering. The problem of cluster-grouping can be regarded as a new type of inductive optimization query that asks for the k best patterns according to a convex criterion. The algorithm *CG* for solving cluster-grouping problems is presented and the underlying mechanisms are discussed. The approach is experimentally evaluated on a number of real-life data sets. The results indicate that the algorithm improves upon the subgroup discovery algorithm *CN2-WRAcc* and is competitive with the clustering algorithm *CobWeb*.

Keywords: clustering, subgroup discovery, correlated pattern mining, inductive querying, constraint-based mining.

1 Introduction

Many problems and settings are described in the machine learning and data mining literature. The techniques that will be addressed in this paper include: subgroup discovery [1,2], clustering [3,4], and correlated pattern mining [5].

In subgroup discovery, the goal is to find groups (often in the form of conjunctive rules $c_1 \wedge ... \wedge c_n \rightsquigarrow a$) that are statistically over- or under-represented w.r.t. a particular target attribute a and thus, since they show unexpected behaviour, are considered interesting. For instance, the group *smoker* is an interesting subgroup w.r.t. *cancer*, as smokers have a higher probability of having cancer. Correlated pattern mining [5,6] can be viewed as an extension of subgroup discovery where one is looking for *interesting* rules (w.r.t. statistical criteria such as χ^2) but with no fixed target attribute. Subgroup discovery is thus closely related to rule learning, while correlated pattern mining is more similar to association rule discovery. In clustering [3,4], the goal is to compute interesting groups, and in *conceptual* clustering, it is furthermore desired to obtain symbolic descriptions of the clusters.

* A 3-page abstract of this paper appeared as Albrecht Zimmermann, Luc De Raedt: *Cluster-Grouping: From Subgroup Discovery to Clustering.* ECML 2004: 575–577.

Although these three techniques are in the literature perceived as being quite different, it turns out that they share a number of characteristics. The key contribution of this paper is the introduction of the *cluster-grouping* problem that subsumes subgroup discovery [1,2], clustering [3], and correlated pattern mining [5]. Cluster-grouping is concerned with finding rules $b_1 \wedge ... \wedge b_k \rightsquigarrow h_1 \vee ... \vee h_n$ that score best w.r.t. an interestingness function σ and a data set $\mathcal{E}$. Cluster grouping rules state that for examples covered by the condition part $b_1 \wedge ... \wedge b_n$ it is possible to reliably predict the attributes h_i in the conclusion part. To address the cluster-grouping problem, we develop the branch-and-bound algorithm *CG* that utilizes the convexity of different evaluation functions. It significantly extends the correlated pattern mining framework by Morishita *et al.* [5], in that it generalizes Morishita *et al.*'s approach to an arbitrary dimension. *CG* is experimentally validated against the heuristic *CN2-WRAcc* algorithm [1] for subgroup discovery and the *CobWeb* algorithm [3] for conceptual clustering. The experiments show that *CG* typically finds better (even optimal) subgroups while exploring less candidates than *CN2-WRAcc*, and that *CG* finds cluster definitions in the form of conjunctions, which are competitive in terms of *Category Utility* [7] with the clusters found by *CobWeb*.

The cluster grouping problem can be viewed as a novel type of inductive query [8,9], in which one is interested in the k best rules with regard to the interestingness function σ. This type of query thus looks for the k optimal rules, which explains why we employ a branch-and-bound algorithm. This is an inductive optimization query which differs from the typical type of constraints used, in which one can decide whether a pattern satisfies the constraint independently of the other patterns in the space. Still, cluster-grouping queries are both declarative and powerful as they can be used to find optimal solutions for a wide range of mining tasks, i.e. subgroup discovery, correlated pattern mining as well as clustering.

We proceed as follows. In the next section we introduce the necessary notations used throughout the paper, and define the problem. In section 3, we explain how cluster-grouping unifies the data mining tasks mentioned. In the fourth section, we discuss the upper bound computation which is necessary for efficiently solving the problem and in section 5 present the algorithm developed. We evaluate our approach in section 6 and finally, we touch upon related work, formulate our conclusions and discuss future research directions in sections 7 and 8.

2 Preliminaries

We consider the problem of cluster-grouping as one of learning interesting, i.e. strongly correlating, rules on a given data set.

2.1 Rules

Let $\mathcal{A} = \{A_1, ..., A_d\}$ be a set of ordered attributes and $\mathcal{V}[A] = \{V_1, ..., V_p\}$ the domain of A. An *instance* e is then a tuple $\langle v_1, ..., v_d \rangle$ with $v_i \in \mathcal{V}[A_i]$. A multiset $\mathcal{E} = \{e_1, ..., e_n\}$ is called a *data set*.

Definition 1 ***(Literal).*** *A* ***literal*** *l is an attribute-value-pair $A = v$ with $v \in \mathcal{V}[A]$. An instance $\langle v_1, ..., v_d \rangle$ is* ***covered*** *by a literal l of the form $A_i = v$ iff $v_i = v$.*

Definition 2 ***(Rule).*** *A* ***rule*** *r is of the form $b \Rightarrow h$ with $b = l_1 \wedge ... \wedge l_i$ the* ***rule body****, and $h = l'_1 \vee ... \vee l'_d$ the* ***rule head****. An instance e is covered by b iff it is covered by all its literals and it is covered by the entire rule r iff it is covered by at least one literal in h as well.*

2.2 Interestingness Measures

As mentioned above we consider interesting rules to be rules that have a strong correlation between their body and their head. It is necessary to define the notion of *support*, when working with correlation measures:

Table 1. Contingency table for $b \Rightarrow h_1$

	h_1	$\neg h_1$	
b	$sup(b \Rightarrow h_1) = y_1$	$sup(b \Rightarrow \neg h_1) = x - y_1$	$sup(b) = x$
$\neg b$	$sup(\neg b \Rightarrow h) = m_1 - y_1$	$sup(\neg b \Rightarrow \neg h_1) = n - m_1 - (x - y_1)$	$sup(\neg b) = n - x$
	$sup(h_1) = m_1$	$sup(\neg h_1) = n - m_1$	n

Definition 3 ***(Support).*** *For a literal l, we define*

$$sup(l) = |\{e \mid e \text{ is covered by } l\}|$$

the ***support*** *of l. Similarly, the support of a rule body b is defined as*

$$sup(b) = |\{e \mid e \text{ is covered by } b\}|$$

and of a rule with a single consequent

$$sup(b \Rightarrow h) = |\{e \mid e \text{ is covered by } b \wedge e \text{ is covered by } h\}|$$

For the remainder of this paper, we will use the following notation to refer to occurrence counts of rules:

Definition 4 ***(Occurrence Counts).*** *For a given rule $b \Rightarrow h_1 \vee ... \vee h_d$ and a given data set $\mathcal{E}$ we define:*

$$\boldsymbol{n} = |\mathcal{E}|,\ \boldsymbol{m}_i = sup(h_i),\ \boldsymbol{x} = sup(b),\ \boldsymbol{y}_i = sup(b \Rightarrow h_i)$$

To facilitate the use of correlation measures, occurrence counts are often organized in contingency tables. A contingency table for a rule body having two values (i.e. *true* and *false*) and a single binary-valued target literal is shown in Table 1. Note that the sum of the cells in a row (column) is equal to the margins of the table, i.e. the rightmost (down-most) entry in a row (column). Correlationmeasures compare for a given cell the product of the corresponding margins to the cell count, thus comparing *expected* to *observed* frequency, and score the difference. Comparing the expected and observed frequency for the upper left cell would e.g. in the χ^2-measure take the form $\frac{(y_1 - m_1 x/n)^2}{m_1 x/n}$.

Example 1. *Consider as an example a database consisting of 50 instances* ($\boldsymbol{n}$)*, for half of which* "heavy" *is* true ($\boldsymbol{m}_1$). *Assume furthermore that* "strong eater" *occurs with support 10* ($\boldsymbol{x}$) *in the database. If eight of the ten instances for which* "strong eater" *is* true *also have* "heavy"=true ($\mathbf{y}_1$)*, then the* χ^2 *measure would give the deviation of expected from observed frequency for the upper left cell a score of* 1.8.

Table 2. Pseudo-Contingency table for $b \Rightarrow h_1 \vee h_2$

	h_1	$\neg h_1$	h_2	$\neg h_2$	
b	$sup(b \Rightarrow h_1)$	$sup(b \Rightarrow \neg h_1)$	$sup(b \Rightarrow h_2)$	$sup(b \Rightarrow \neg h_2)$	$sup(b) = x$
$\neg b$	$sup(\neg b \Rightarrow h_1)$	$sup(\neg b \Rightarrow \neg h_1)$	$sup(\neg b \Rightarrow h_2)$	$sup(\neg b \Rightarrow \neg h_2)$	$sup(\neg b) = n - x$
	$sup(h_1) = m_1$	$sup(\neg h_1) = n - m_1$	$sup(h_2) = m_2$	$sup(\neg h_2) = n - m_2$	n

Normally, increasing the number of involved literals leads to an increase of dimension of the contingency table to capture all dependencies among the literals. Since we are not interested in the dependencies between the h_i, we use tables such as the one in Table 2. We call this kind of table a *pseudo*-contingency table. The main difference w.r.t. a regular high-dimensional contingency table, a so-called multi-way table, is that the margin of a row is not equal to the sum of row-cells anymore. Calculation of a correlation measure still consists of comparing the product of the margins to the cell count.

Definition 5 ***(Stamp Point).*** *For a given data set* $\mathcal{E}$ *every rule* r *of the form* $b \Rightarrow h_1 \vee ... \vee h_d$ *induces a tuple* $\langle x, y_1, ..., y_d \rangle$ *of variables introduced in definition 4. This tuple is called the* ***stamp point*** *of* r*, denoted* sp(r)*, cf. [5].*

Consider now an interestingness measure such as χ^2, *Category Utility*, *Information Gain*, or *Weighted Relative Accuracy* defined on a pseudo-contingency table. Since $\boldsymbol{n}$ and the $\boldsymbol{m}_i$ are constant for a given data set, a given interestingness measure $\sigma(r)$ can be defined as a function of $d + 1$ variables

$$\sigma : \mathbb{N}^{d+1} \mapsto \mathbb{R},$$

mapping the stamp point *sp(r)* to a real number.

Example 2. *This means that* sp("strong eater"⇒"heavy")$=\langle 10, 8 \rangle$*, a deviation that is quantified by* $\chi^2(\langle 10, 8 \rangle) = 4.5$. *Lets also consider* "sportive" *as interesting. Under the assumption that among all instances that are matched by* "strong eater"=true*, three also have* "sportive"=true*, we arrive at the stamp point* sp("strong eater"⇒"heavy"∨"sportive") $= \langle 10, 8, 3 \rangle$*, and if furthermore* "sportive"*'s frequency on the entire dataset is 30,* $\chi^2(\langle 10, 8, 3 \rangle) = 10.397$.

Now we are able to define the cluster-grouping problem:

Definition 6 ***(Cluster-Grouping Problem).***
Given:

– *A set of literals $\mathcal{L}$*
– *A data set $\mathcal{E}$*
– *An interestingness measure σ*
– *A set of target literals $\mathcal{T}$*

Find:
The set of k rules expressible in $\mathcal{L}$ having the highest value of σ on $\mathcal{E}$ w.r.t. the given target literals $\mathcal{T}$.

3 Unifying Several Data Mining Tasks

Let us now formulate the three data mining tasks mentioned before, i.e. correlated pattern mining, subgroup discovery and conceptual clustering, in the framework introduced above.

3.1 Correlated Pattern Mining

The correlated pattern mining task was introduced by Morishita and Sese [5] in the context of itemset mining. Their main motivation lies in the fact that association rules with very high confidence might still carry no information. Therefore, using a correlation measure to evaluate the quality of rules found will result in more interesting relationships. The task of correlated pattern mining can be formulated as a cluster-grouping problem with the following characteristics:

Let $\mathcal{I} = \{I_1, ..., I_z\}, \forall I \in \mathcal{I} : \mathcal{V}[I] = \{false, true\}$ the set of items,

– $\mathcal{L} = \{I = true \mid I \in \mathcal{I}\}$
– $\mathcal{E}$ a transaction data base
– σ is a correlation measure such as χ^2
– $\mathcal{T}$ a single literal $l \in \mathcal{L}$

While Morishita and Sese restrict their approach to the classical itemset setting, the use of a correlation measure would also allow the mining of negative relationships, i.e. the expansion of $\mathcal{L}$ to include literals of the form $i = false, i \in \mathcal{I}$, would allow one to find relationships in which the absence of an item correlates with the presence of another.

3.2 Subgroup Discovery

Lavrač *et al.* [1] argue convincingly that a rule learning algorithm such as *CN2* [10], used together with a metric measuring positive correlation such as *Weighted Relative Accuracy* (*WRAcc*) [11], can be employed to find subgroups, i.e., subsets of a data set that show unexpected behaviour with regard to a specific target attribute. The subgroup discovery problem can be formulated as a cluster-grouping problem in the following way:

Let $A_t \in \mathcal{A}$ be the target attribute we are interested in.

- $\mathcal{L} = \{A = v \mid A \in \mathcal{A} \setminus \{A_t\}, v \in \mathcal{V}[A]\}$
- $\mathcal{E}$ a data set
- σ is *WRAcc*
- $\mathcal{T} = \{A_t = v \mid v \in \mathcal{V}[A_t]\}$

Lavrač *et al.* employ beam search to find subgroups. Depending on the beam size, this approach may lead to suboptimal rules being found. Additionally, if there are several *best* subgroups the wrong beam size might cause the exclusion of some of them.

3.3 Conceptual Clustering

The goal in conceptual clustering is to group instances in a data set into groups that exhibit high intra-cluster similarity and high inter-cluster dissimilarity. Instances are considered similar if they agree on the values of many attributes. One measure for judging the quality of a set of clusters is *Category Utility.* The task of conceptual clustering - with binary attributes only - can be formulated as a cluster-grouping task with the following characteristics:

Let $\mathcal{A} = \{A_1, ..., A_d\}, \forall A \in \mathcal{A} : \mathcal{V}[A] = \{false, true\}$ be a set of attributes:

- $\mathcal{L} = \{A = v \mid A \in \mathcal{A}, v \in \mathcal{V}[A]\}$
- $\mathcal{E}$ a data set
- σ is *Category Utility*
- $\mathcal{T}$ all literals of the form $A = true, A \in \mathcal{A}$

The well-known clustering algorithm *CobWeb* [3] aims at finding clusters with high *Category Utility.* The drawbacks of *CobWeb* lie in the fact that it processes instances iteratively, possibly leading to a sub-optimal solution, and that the direct assignment of instances to clusters might lead to clusters that cannot be described using a conjunction of literals. Instead, *CobWeb* characterizes clusters by conditional probabilities on attribute-value pairs. For high-dimensional data this representation is probably not very human-readable; additionally, it makes assignments of future instances to clusters somewhat more difficult.

3.4 Contribution

The key contribution of this paper is 1) the introduction of the cluster-grouping problem that makes abstraction of these three problem settings and 2) the formulation of an algorithm that allows one to tackle the cluster-grouping problem.

The algorithm will always output the solution or solutions achieving the highest value of σ. Usually, this can only be guaranteed by considering all possible rules which is often computationally not feasible. This problem is solved using pruning techniques that are derived from the observation that many correlation measures (such as the ones mentioned above) are in fact convex functions.

4 Upper Bound on Convex Correlation Measures

While the cluster-grouping task can be tackled using a heuristic algorithm, such an approach would not be guaranteed to find the optimal rules while an exhaustive technique is not feasible for higher-dimensional data (and therefore a large set of literals). Based on the convexity of correlation measures it is however possible to calculate an upper bound on the future value of σ for specializations of a given rule. This upper bound is used to prune away parts of the search space known not to produce interesting solutions and focus the search on promising parts of the search space.

4.1 Convexity

Convexity is formally defined as follows.

Definition 7 ***(Convexity).*** *A function* $f : D \mapsto \mathbb{R}$ *is convex iff* D *is a convex set and* $\forall x_1, x_2 \in D, \lambda \in [0,1] : f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda) f(x_2)$.

It can be proven that χ^2, *WRAcc*, and *Category Utility* as well as other correlation measures fulfill the second criterion.

For the definition to hold it is also necessary for the domain of the function to form a convex set. A set $S \subset \mathbb{N}^n$ is convex if there are no points a and b in S such that there is a point on the line between a and b that does not belong to S. We will argue below following paragraph why this holds for the domain of the correlation functions mentioned above.

Let r denote a rule of the form $b \Rightarrow h_1 \vee ... \vee h_d$, b' a specialization, that is an extension with at least one literal, of b, r' of the form $b' \Rightarrow h_1 \vee ... \vee h_d$, and $sp(r), sp(r')$ the corresponding stamp points. Define $S_{act} = \{sp(r')\}$ as the set of *actual* stamp points of all rules r' whose rule body is a specialization of b. This set is unknown until all specializations of b have been created and evaluated on the data set. However, instead of computing S_{act}, one can try to approximate it by the set of *possible* stamp points $S_{poss} \supseteq S_{act}$. The approximation is possible because the stamp point $sp(r) = \langle x, y_1, ..., y_d \rangle$ constrains all $sp(r')_{poss} = \langle x', y'_1, ..., y'_d \rangle \in S_{poss}$ in the following way:

1. $x \geq x' \geq 0$
2. $\forall i : x \geq y_i \geq 0$
3. $\forall i : y_i \geq y'_i \geq 0$
4. $\forall i : x' - y'_i \geq x - y_i$

Each of these inequalities defines one or more convex sets in $d+1$-dimensional space. Since S_{poss} is the intersection of all these sets it is a convex set itself. S_{poss} is the domain of σ given b. Convex functions take their extreme values at the points forming the convex hull of their domain D [12]. So by evaluating σ on the points forming the convex hull of S_{poss}, it is possible to obtain an upper bound for the value that σ can take on *any* point $sp(r') \in S_{poss}$. Since S_{act} is a subset of S_{poss}, this upper bound is also an upper bound on the value of σ on any point of S_{act}.

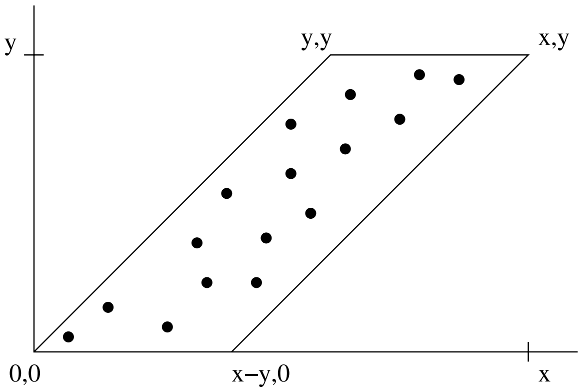

Fig. 1. Actual stamp points $sp(r')$ and convex hull of S_{poss} for $\langle x, y\rangle$(taken from [5])

For the two-dimensional case, the convex hull of S_{poss} is the parallelogram defined by the vertices $\langle x, y\rangle$, $\langle y, y\rangle$, $\langle x-y, 0\rangle$, $\langle 0, 0\rangle$ as shown in Figure 1. These vertices are derived by computing the four points $\langle x_{max}, y_{max}\rangle$, $\langle x_{min}, y_{max}\rangle$, $\langle x_{max}, y_{min}\rangle$, $\langle x_{min}, y_{min}\rangle$, while taking into account the four inequalities above.

Example 3. *Continuing our example, if the body of the rule is extended, e.g. with* "strong smoker"*, at most 10 instances are covered by the new rule body. Among those, maximally 8 will have* "heavy"=True*. Should* "strong smoker" *and* "heavy" *correlate negatively, the presence of* "strong smoker" *would reduce the number of instances in which* "heavy" *was present, which also means that the total number of instances covered is reduced. Any future stamp point will therefore be inside the parallelogram defined by* $\langle 0, 0\rangle, \langle 8, 8\rangle, \langle 2, 0\rangle, \langle 10, 8\rangle$.

For computing the upper bound on σ, the points $\langle x, y\rangle$ and $\langle 0, 0\rangle$ do not have to be considered. The first point describes a specialization with the same coverage as the current rule. Since such a specialization is expected to generalize less well to as yet unseen examples, we give preference to the rule already found. The second denotes a rule that does not cover any examples, which renders it useless. Therefore, $ub_\sigma(b) = \max\{\sigma(y, y), \sigma(x - y, 0)\}$.

Example 4. *Continuing our example from above, this means that for σ being χ^2, $ub_{\chi^2}(r) = \max\{9.52, 2.08\}$, given $x = 10$, $y_1 = 8$. Since* 9.52 *is larger than*

$\chi^2(x, y_1) = 4.5$ *there might be a specialization of T that discriminates better than r itself and therefore exploring this search path is worthwhile.*

4.2 Extension to Higher Dimensions

The approach described above was introduced by Morishita and Sese [5]. Their work is however restricted to a *single binary* target literal. To calculate an upper bound for the type of rules given in Definition 2, an extension to higher dimensions is necessary. Even though this extension is not too difficult, two problems arise:

Firstly, the number of convex hull points grows exponentially with the dimensionality of the data since all combinations of minimum and maximum values for x and the y_i have to be considered, meaning that $|\{min, max\}|^d = 2^d$ points would have to be evaluated. Secondly, it is insufficient to simply consider y_i as maximum value of y'_i and 0 as minimum. Instead, the dependencies between different y_i have to be considered.

Both of these problems can be somewhat lessened by enumerating the convex hull points in a different manner. This approach was first applied by Morishita and Sese [13] to the problem of clustering numerical values into clusters describable by binary attributes. To make the enumeration technique more understandable, consider *Category Utility* as an example. *Category Utility* for two clusters is defined as:

$$CU(C_1, C_2) = \frac{1}{2} \sum_{C \in \{C_1, C_2\}} P(C) \sum_{A \in \mathcal{A}} \sum_{v \in \mathcal{V}[A]} P(A = v \mid C)^2 - P(A = v)^2$$

Now consider the case that membership in the first of the two clusters is based on whether an instance is covered by a rule body b_r (for a given set of binary target attributes $\mathcal{A}_t$). If the instance is not covered by b_r, it is assigned to the second cluster. In that case, *Category Utility* can be re-written as:

$$CU(b_r, \mathcal{A}_t) = \frac{1}{2} \left(\sum_{b \in \{b_r, \neg b_r\}} P(b) \sum_{A \in \mathcal{A}_t} \sum_{v \in \{\text{true, false}\}} \frac{P(b \Rightarrow A = v)^2}{P(b)^2} - P(A = v)^2 \right)$$

This formula can be simplified by pushing the $\frac{1}{2}P(b)$ into the sum:

$$CU(b_r, \mathcal{A}_t) = \sum_{A \in \mathcal{A}_t} \sum_{v \in \{\text{true, false}\}} \sum_{b \in \{b_r, \neg b_r\}} \left(\frac{1}{2} P(b) \left(\frac{P(b \Rightarrow A = v)^2}{P(b)^2} - P(A = v)^2 \right) \right),$$

so that the total *Category Utility* can be expressed as the sum of several partial *Category Utilities*:

$$CU(b_r, \mathcal{A}_t) = \sum_{A \in \mathcal{A}_t} CU(b_r, \{A\})$$

Note that while *Category Utility* is used in this example the same property holds for correlation measures such as χ^2, *WRAcc*, *Information Gain* and others if the relations between the target attributes are ignored as in the cluster-grouping setting.

The set $\mathcal{A}_t$ corresponds to the tuple $\langle y_1, \ldots, y_d \rangle$, with every single $A_i \in \mathcal{A}_t$ corresponding to a y_i. Formulated in the *stamp point* notation, the above equation thus becomes:

$$CU(x, y_1, ..., y_d) = \sum_{i=1}^{d} CU(x, y_i),$$

and the maximization of CU the maximization of the sum of the partial *Category Utilities*:

$$\max_{\langle x', y'_1, \ldots, y'_d \rangle} CU(x', y'_1, \ldots, y'_d) = \max_{\langle x', y'_1, \ldots, y'_d \rangle} \sum_{i=1}^{d} CU(x', y'_i).$$

This alone does not make the maximization process easier since all 2^d points on the convex hull would still have to be evaluated. The number of computations would decrease if the maximization could happen for each term of the sum independently of the others. If this would be done for arbitrary x', the maximal values for different terms could induce conflicting values for x', resulting in an inconsistent stamp point. But for fixed x' each of the d terms can be maximized separately and if x' takes all values in $[0, x]$, we are not losing any solutions:

$$\max_{\langle x', y'_1, \ldots, y'_d \rangle} \sum_{i=1}^{d} CU(x', y'_i) = \max_{0 \leq x' \leq x} \left\{ \sum_{i=1}^{d} \max_{y'_i} CU(x', y'_i) \right\}$$

For each of the y' this maximum is independent of the other y'_i. The minimum and maximum values y^i_{min}, y^i_{max} are determined by the values x' and y_i. Thus, maximizing the partial *Category Utility* turns into calculating it for the minimum and maximum value and keeping the larger one:

$$\max_{0 \leq x' \leq x} \left\{ \sum_{i=1}^{d} \max_{y'_i} CU(x', y'_i) \right\} = \max_{0 \leq x' \leq x} \left\{ \sum_{i=1}^{d} \max_{y'_i \in \{y^i_{min}, y^i_{max}\}} CU(x', y'_i) \right\}$$

Therefore for a given x', $2d$ calculations have to be performed. Since $x' = 0$ and $x' = x$ can be ignored, following the argument in the last paragraph of section 4.1, a total of $2d(x - 2)$ calculations have to be performed to compute the upper bound on the values of σ for a given b_r.

5 The *CG*-Algorithm

In this section we present the algorithm *CG* for cluster-grouping. We also explain the upper bound calculation in more depth.

```
CG
$\mathcal{E}$ - data set
$\sigma$ - correlation measure

$P := \{\top\}, \tau := -\infty, S := \emptyset$
while $P \neq \emptyset$
  $b_{mp} := \arg\max_{ub(b)}\{b \in P\}$
  $C := \rho(b_{mp})$
  $\forall c_i \in C$
    compute $sp(c_i)$, calculate $\sigma(c_i)$
    $ub_\sigma(c_i) := UpperBound(sp(c_i))$
    $\tau := \max\{\tau, \sigma(c_i)\}$
  $S := \{s \in S \mid \sigma(s) = \tau\} \cup \{c \in C \mid \sigma(c) = \tau\}$
  $P := \{p \in P \mid ub_\sigma(p) \geq \tau\} \cup \{c \in C \mid ub_\sigma(c) \geq \tau\}$
return $S$
```

Fig. 2. The *CG* algorithm

The cluster-grouping algorithm *CG* (listed in Figure 2) is essentially a branch-and-bound algorithm. Starting from the most general rule body (denoted by $\top$), in each iteration the rule body $b_{mp} \in P$ with the highest upper bound is specialized. We use an optimal refinement operator ρ:

Definition 8 ***(Optimal Refinement Operator).*** *Let $\mathcal{L}$ be a set of literals, $\prec$ a total order on $\mathcal{L}$, $\tau \in \mathbb{R}$.*

$$\rho(b) = \{b \wedge l_i \mid l_i \in \mathcal{L}, ub_\sigma(l_i) \geq \tau, \forall l \in r : l \prec l_i\}$$

is an ***optimal refinement operator****.*

The operator ensures that each rule body will be created and evaluated at most once during a run of the algorithm. Since only literals are added whose upper bound exceeds the threshold, the resulting specializations have a chance of exceeding or matching the current threshold. The created specializations are then evaluated on the data set and the σ-scores and upper bounds are calculated. All specializations whose upper bound is not above the threshold τ are pruned. If possible, the threshold is raised. For the case shown in the Figure 2, the best score seen so far is used as threshold, but the algorithm can be trivially modified so that k best rules are found. Specializations whose score matches the current threshold are added to the set of solutions S. The set of promising rule bodies P is pruned using the threshold and all specializations whose upper bound exceeds τ are added.

5.1 Upper Bound Computation

As the upper bound is so essential for the mining process, we explain the algorithm for computing it in greater detail.

```
UpperBound
$\sigma$ - correlation measure
$\langle x, y_1, ..., y_d \rangle$ - stamp point

$x' = 1$
$ub_\sigma = 0$
while $x' \leq x - 1$
  $\forall i \in \{1, ..., d\}$
    $y^i_{max} = \min\{x', y_i\}, y^i_{min} = \max\{0, y_i - (x - x')\}$
    $add_i = \max\{\sigma(x', y^i_{max}), \sigma(x', y^i_{min})\}$
  $ub_\sigma = \max\{ub_\sigma, \sum_{i=1}^{d} add_i\}$
  $x' + +$
return $ub_\sigma$
```

Fig. 3. Algorithm for calculating the upper bound on σ

It works as follows. The loop runs from 1 through $x - 1$ (since $x' = 0$ and $x' = x$ do not have to be considered). For each possible x' value the corresponding maximum and minimum values y^i_{max}, y^i_{min} of a variable y'_i, given y_i and x', are calculated. Maximizing the correlation measure for fixed x' amounts to maximizing the terms $\sigma(x', y'_i)$. This is achieved by evaluating the measure for each maximum and minimum value y^i_{max}, y^i_{min} and choosing the larger value of those two to be added to the σ value for the current x'. The largest of these $x - 2$ correlation values is taken as the upper bound on σ.

To calculate the maximum value of σ for a given x', it is necessary to derive the minimum and maximum value for each y'_i while considering the inequalities in section 4.1. The details of deriving those extreme values are described in the following paragraphs.

Minimum-Maximum-Derivation. From inequality 2 it follows that y'_i cannot be greater than x' and inequality 3 states that $y'_i \leq y_i$. Therefore y^i_{max} can be calculated as $y^i_{max} = \min\ \{x', y_i\}$.

Since the y_i are occurrence counts in a data set, they cannot become negative. Inequality 4 finally has to be understood in the following way: since every instance that is counted for a y_i is also counted for x, a decrease in y_i has to be accompanied by a decrease in x. This means that y^i_{min} cannot always be set to zero. If the difference between x' and x is less than y_i, y_i must not be decreased by more than that difference. Therefore $y^i_{min} = \max\ \{0, y_i - (x - x')\}$.

Example 5. *Lets consider two values for x' during the loop. For $x' = 9$, $y'_1 \leq \min\{8, 9\}$, and since only one instance less would be covered, even if* "heavy" =True *held for this instance, the occurrence of* "heavy" *cannot sink below 7. Thus $7 \leq y'_1 \leq 8$. By similar reasoning one arrives at $2 \leq y'_2 \leq 3$. For $x' = 4$ the situation is somewhat different. Since x was reduced by 6 and each of the instances not covered anymore might have included* "heavy" =True *and* "sportive" =True, *the minimum values for y'_1 and y'_2 are* $\max\{0, 8 - 6 = 2\}$

and $\max\{0, 3-6=-3\}$ *respectively. Similarly, their value is bounded by the respective values of the* y_i *and* $x'=4$*. This leads to* $2 \leq y'_1 \leq 4$ *and* $0 \leq y'_2 \leq 3$*.*

6 Experimental Evaluation

Let us now investigate the applicability and performance of the *CG* algorithm on the three tasks addressed in this paper: correlated pattern mining, subgroup discovery and conceptual clustering.

The data sets employed in the experiments were selected from the *UCI Machine Learning Repository* [14]. Since the approach is limited to data sets described by binary attributes, continuous attributes were discretized and nominal attributes binarized. Two options for discretization were employed. In the naïve version, the mean value of an attribute is computed and taken as a threshold. For some data sets this leads to unbalanced attributes. For these sets, we also split the attribute values into two equal frequency bins. They are denoted with a trailing "-equal" in the name

6.1 Correlated Pattern Mining

Since the rules mined by Morishita and Sese are special cases of clustering grouping rules and the pruning technique is based on the same principles, it follows that the *CG* algorithm is applicable to correlated pattern mining and also that it will produce the same solution as Morishita and Sese's approach. We therefore repeat no experiments on this task.

6.2 Subgroup Discovery

To evaluate *CG* on subgroup discovery, we compare it with *CN2-WRAcc* [1] in two settings. The first setting corresponds to the inner loop of the *CN2-WRAcc* algorithm that employs beam search to find the best rule; the second one to the full *CN2-WRAcc* implementation that incorporates the covering algorithm.

***CN2-WRAcc* Beam Search.** As argued before, this setting corresponds to cluster-grouping. Whereas *CG* guarantees to find the optimal rules, the *CN2-WRAcc* beam search is heuristic and offers no guarantees of optimality. Therefore, in a first set of experiments concerning subgroup discovery, we investigate (1) to what extent *CN2-WRAcc* beam search finds optimal rules and also (2) how the *CG* algorithm compares with beam search in terms of efficiency. As the criterion for the optimality (1), we computed all optimal rules using *CG*, and verified whether *CN2-WRAcc* finds *any* optimal rule (*OF* in the tables) and *all* rules with optimal values (*AF*). With regard to (2), we compared the number of candidate rules that were evaluated by the two algorithms. Because the performance of beam search depends heavily on the beam-size, we experimented with different beam sizes.

Table 3. Number of Candidate Rules evaluated and Optimality of Subgroups found by Beam Search

Data Set	$BS1_{Avg}$	$BS5_{Avg}$	$BS10_{Avg}$	CG_{Min}	CG_{Max}	CG_{Avg}	AF	OF
Car	85	278	483	21	102	45	Yes	Yes
Zoo	2133	9173	17630	147	5775	1525	No*	Yes
Nursery	141	498	970	27	274	85	Yes	Yes
Breast-W	529	1882	3539	91	99	95	Yes	Yes
Voting Record	301	1139	2162	33	36	35	Yes	Yes
Mushroom	1806	7674	15006	172	219	196	No**	No**
Soybean	2036	9076	17712	1943557 (13116^+)	2097405 (468869^+)	2007660 (284680^+)	No*	No*

*Not for all classes, for some not for all beam sizes
**Not for all beam sizes
$^+$Break value $f = 5$

The first column of Table 3, $BS1_{Avg}$, denotes the number of candidate patterns evaluated during a search with beam size 1 by *CN2*, $BS5_{Avg}$ and $BS10_{Avg}$ for beam size 5 and 10 accordingly. The columns CG_{Min} and CG_{Max} denote the minimum and maximum number of candidates generated by *CG*. For *CG* those numbers vary with the class considered, whereas for *CN2* these numbers are quite close to one another for the different classes. This also explains why for *CN2* we report on the average values. In addition, the average number of candidates CG_{Avg} is reported to show the general behavior of the algorithm.

In most cases there is a substantial reduction in the number of candidate rules considered during the search process even when the *CG*-algorithm is compared to the greedy (i.e. beam size 1) approach. The difference is even more pronounced for wider beams. For those cases where more candidate rules are considered by *CG*, the beam search algorithm often either fails to uncover all interesting rules or finds suboptimal ones. This also happens in cases in which the beam size approach evaluates more rules than *CG*.

The experiments also show that increasing the beam size does not necessarily lead to better results. For the first class considered in the *soybean* data set, beam sizes 10 and 15 induce incomplete rule sets while beam sizes 4 and 5 generate the correct results, showing the difficulty of guessing the right beam size. The most likely explanation for this effect is probably that locally good solutions are kept, that are excluded by narrower beams. If those solutions have a large number of refinements they replace promising rules that would have been kept for specialization in runs with smaller beam sizes. It might be interesting to further investigate this phenomenon. The circumstances under which not (all) optimal rules are found are denoted by the asterisks.

In some cases the *CG* algorithm investigates an excessive amount of candidate rules. The effect occurs especially for rather small values of σ and is due to the upper bound being overly optimistic. We therefore built a variant that is no longer guaranteed to find the optimal rules: whenever the pruning threshold is raised the number of candidate rules considered so far is memorized. Once

f times that number of candidates has been visited without improvement of the pruning threshold the search is terminated. The cutoff value f has to be determined by the user. This version of the algorithm behaves like a local search approach, assuming that failure to find a higher value in the (rather extensive) neighborhood of the currently best solution means that a global maximum has been found. This variant is computationally less expensive than *CG*, while it still finds all optimal rules on the studied data sets even for small value of f ($f = 5$).

***CN2-WRAcc* Sequential Covering.** In the second experiment, we employ the sequential covering algorithm as a wrapper around *CG* and the *CN2-WRAcc* beam search procedure. This corresponds to the original setting addressed by *CN2-WRAcc*. The results are shown below (Table 4).

To consider the most favourable efficiency results for *CN2-WRAcc*, we report only the minimum and maximum number of candidate rules considered using beam-size 1. Wider beams will generate more candidate patterns. Averaging these numbers is not really sensible since the impact of the number of iterations per class is obviously strong. These results are contrasted with those for *CG*. The last two columns convey the same information as in Table 3. To allow for a fair comparison we considered larger beam sizes for *CN2-WRAcc* when deciding whether the algorithm was capable of finding (all) optimal rules. The results for sequential covering are in line with and confirm those for the underlying components.

Table 4. Number of Candidate Rule evaluated and Optimality of Subgroups found by the Sequential Covering Approach

Data Set	$BS1_{Min}$	$BS1_{Max}$	CG_{Min}	CG_{Max}	CG_{Avg}	AF	OF
Car	76	651	21	203	101	Yes	Yes
Zoo	2109	2153	147	5775	1525	*No**	Yes
Nursery	129	292	27	274	95	Yes	Yes
Breast-W	3703	6349	739	1382	1061	Yes	Yes
Voting Record	903	903	2235	34659 (4480^+)	18447 (3358^+)	Yes	Yes
Mushroom	3558	8376	805	5638	3222	*No***	*No***

*Not for all classes, for some not for all beam sizes
**Not for all beam sizes
$^+$Break value $f = 5$

6.3 Clustering

In the third experiment, we evaluate the performance of a hierarchical clustering algorithm based on *CG* with *Category Utility*. *CG* is used to find the best rule w.r.t. to *Category Utility* on the original data set. The rule found is used to split the data set into two subsets. *CG* is then called recursively on these subsets.

The algorithm continues until the best *Category Utility* on a subset falls below a user-specified criterion. In this way, a kind of decision tree is induced that can be used to assign future instances to the clusters. We compare our clustering approach to the well-known clustering algorithm *CobWeb*. We employed the Weka [15] implementation of *CobWeb* in our experiments. For each data set with c classes, the two clustering algorithms were run until they constructed c clusters. This was enforced by setting the minimal threshold for further refining clusters to an adequate value.[1] This was done to provide a fair basis for comparison. Furthermore, because *CobWeb* is incremental and sensitive to the order in which the examples are presented, the *CobWeb* results were averaged over 10 randomized orders.

The results are shown in Table 5. The evaluation criteria employed to compare the performance of both algorithms are:

- *Rand Index* [16] for comparing the agreement of the clusterings found,
- *Category Utility* of the discovered solutions.

The *Rand* index is the fraction of pairwise grouping decisions on which the two clusterings agree. Let $\mathcal{E} = \{e_1, ..., e_n\}$ be a data set and $\mathcal{C}_1, \mathcal{C}_2$ two clusterings of $\mathcal{E}$. For each pair of instances e_i, e_j, $\mathcal{C}_l$ either assigns them to the same cluster or to different clusters. Let *pos* be the number of decision where e_i, e_j are in the same cluster in both clusterings and *neg* the number of decisions where they belong to different clusters in both $\mathcal{C}_l$. The *Rand Index* is defined as:

$$Rand(\mathcal{C}_1, \mathcal{C}_2) = \frac{pos + neg}{n * (n-1)/2}$$

Table 5. *Category Utility* and *Rand Index* for clusterings found

Data set	***CU CG***	***CU CobWeb***	***Rand Index***
Breast-W	0.62	0.6496 ± 0.0001	0.91675 ± 0.006
Breast-W-equal	1.088	$1.147 \pm 1.95 * 10^{-5}$	0.93093 ± 0.0025
Credit-A	0.379	0.374 ± 0.0178	0.95283 ± 0.148
Credit-A-equal	0.6241	0.6243 ± 0.00067	0.9959 ± 0.0034
Glass	0.301	0.291 ± 0.0125	0.90391 ± 0.081
Hepatitis	0.446	0.459 ± 0.0142	0.83667 ± 0.0534
Iris	0.5369	0.5321 ± 0.0083	0.91952 ± 0.0799
Sick	0.2132	0.2077 ± 0.0171	0.98196 ± 0.0567
Voting Record	1.362	1.468 ± 0.0001	0.85753 ± 0.0043
Zoo (6 clusters)	0.6398	0.6349 ± 0.005	0.994093 ± 0.0043
Zoo (5 clusters)	0.7187	0.7196 ± 0.004	0.964357 ± 0.0056

As can be seen in Table 5, the results of *CG* are comparable with those of *CobWeb*. Indeed, the *Category Utilities* of the obtained clusterings are very

[1] Due to the different orderings of the examples, *CobWeb* sometimes finds 5 clusters and sometimes 6 in the zoo data set. Therefore, both 5 cluster and 6 cluster experiments were performed using *CG* as well.

similar. Sometimes, *CG* is slightly better, sometimes *CobWeb* is. Furthermore, the *Rand Index* indicates that the agreement between the clusters found is high, more than 90% in all but two cases.

On the other hand, *CG* computes definitions of the clusters, whereas *CobWeb* does not. Therefore, as a final means of comparison, we employed the rule learner PART [17], available within Weka, on the clusters produced by *CobWeb*. This procedure allows one to obtain conjunctive rules describing the clusters computed by *CobWeb*, which can then be compared to the conjunctive descriptions generated by the *CG*.

Table 6. Rules learned on *CobWeb* clusters using PART

Data set	*CobWeb*/*CG*-Rules	Assignment Accuracy
Breast-W	Superset	97.8%
Breast-W-equal	Superset/Specializations	97.8%
Credit-A	Same	100%
Credit-A-equal	Same/Superset	99.7%
Glass	Superset/Specializations	98.6%
Hepatitis	Superset/Specializations	94.2%
Iris	Same/Different	99.3%/100%
Sick	Same/Superset	100%
Voting Record	Superset	96.3
Zoo (6 clusters)	Same	97%
Zoo (5 clusters)	Same/Generalizations	98%

Table 6 shows the results of this evaluation. For each data set the relation between rules learned on the *CobWeb* clusters and the rules computed by *CG* is listed. Additionally, the right column, labeled **Assignment Accuracy**, shows the fraction of instances that those rules would be assign to their correct clusters. This is used as a measure for how well *CobWeb*'s clusters can be described by conjunctive rules.

There are varying relations between *CG*'s and *CobWeb*'s rules. *Superset* means that the *CobWeb* rules form a superset of the *CG* rules, *Specializations* that they are specializations (*Generalizations* analogous), *Same* that the same set of rules was found. Several entries for one data set denote that several *CobWeb* solutions (due to different ordering of instances) induced different rule sets and therefore different relations.

A special case is the iris data set. On some *CobWeb* solutions PART induced rules that were completely different from *CG* rules. Those had no misassignments. The rules learned using PART provide further evidence that the results obtained by *CobWeb* and *CG* are closely related. As *CobWeb* does not enforce conjunctive descriptions of the clusters *during* its search process, it is more flexible in assembling the clusters. However, because of the involvement of all attributes in the representation the results are also less interpretable. Because the resulting clusters are so close, *CG* seems preferable as it also produces symbolic descriptions.

7 Related Work

The *CG* algorithm is a substantial extension of the Morishita and Sese algorithm for correlated pattern mining. They have also developed a clustering algorithm based on their technique [13], the main difference being that their goal is to cluster numerical values (gene expression levels) using interclass variance. They also aim at describing the clusters found by conjunctions of binary attributes to make the resulting clusters more human-understandable.

The cluster-grouping problem is also related to feature selection in conceptual clustering and semi-flexible prediction [18,19]. Talavera's [18] motivation for feature selection in conceptual clustering is somewhat related to our motivation insofar as he is aiming for better comprehensibility, exclusion of irrelevant features and more efficient clustering processes (both when creating and using the clusters). There is also some similarity in where in the algorithm the feature selection happens, since it is redone for each node in the hierarchical clustering tree. This is called *local* or *dynamic* selection. The main differences are two-fold: firstly, Talavera's work still retains *CobWeb*'s representation and only achieves better comprehensibility by reducing the number of considered attributes. Secondly, in his approach each attribute is scored *before* the actual clustering step, whereas *CG* performs feature selection as part of the clustering process itself.

Cardie [19] defines semi-flexible prediction as learning to predict a set of features known *a priori* as opposed to inflexible prediction (classification) and flexible prediction (clustering). Her approach involves automated feature selection for each attribute to be predicted separately. These features are then used in subsequent independent prediction of the attributes. In contrast, we attempt to predict a disjunction of attributes from a shared set of antecedents instead.

Finally, cluster-grouping is in many aspects related to the confirmatory induction setting in the *Tertius* system by Flach *et al.* [20]. As in *CG*, rules with disjunctive rule heads are found. It is interesting to note in this context that the rule head is treated as a single target while *CG* treats each literal separately. Flach's work abstracts from the general correlation setting in which correlation is symmetric and instead focuses on the number of counter-instances to a given rule, thus considering only directed associations. Using an *optimistic* estimate (an upper bound) they prune non-promising candidates and find and rank optimal rules. Focusing on counter instances only allows more flexibility regarding the rule head, i.e. the set of literals need not be fixed.

8 Conclusion and Future Work

We have introduced the problem of cluster-grouping and argued that it unifies several popular data mining tasks. We developed the algorithm *CG*, a branch-and-bound algorithm that relies on convex correlation functions to find optimal solutions. The cluster-grouping framework lends itself in a natural way for inductive querying. However, rather than imposing normal constraints, cluster-grouping queries are a form of optimization query, in that they look for the k best patterns.

We have shown that our approach is an extension of Morishita's and Sese's work in that it allows one to deal with several target attributes. We have provided experimental evidence that the *CG* algorithm is well-suited for subgroup discovery in that it offers significant advantages when compared to the *CN2-WRAcc* approach, albeit sometimes at the cost of efficiency. *CG* is also competitive with one of the best-known clustering algorithms, *CobWeb*, while creating better interpretable solutions.

Further research will proceed in several directions. First, as can be seen in the experiments, the effectiveness of the pruning step depends strongly on the tightness of the upper bound calculated. Therefore, it is desirable to tighten future support estimates and therefore attainable values of σ. Second, since *Information Gain* is also convex, the technique should in principle be usable in the formation of multi-variate decision trees [21]. Third, probably necessary for usage with decision trees and an interesting extension in itself is the question of how to extend the *CG* to handle target attributes having more than two values. Fourth, since *Foil-Gain* is similar to *Information Gain* and also (under certain assumptions) a convex function, extension of the *CG* algorithm to first-order logic should be possible. Finally, in the context of inductive querying, some new challenges are raised by optimization queries. Most notably, the question arises as to whether we can integrate inductive optimization queries with the more traditional monotonic and anti-monotonic constraints that have been employed within the inductive querying literature.

Acknowledgments

We sincerely thank Shinichi Morishita and Jun Sese for helpful discussion and comments on our work. We also thank our fellow researchers for constructive comments and helpful suggestions.

This work was partly supported by the EU IST project cInQ (consortium on discovering knowledge with Inductive Queries), contract no. IST-2000-26469.

References

1. Todorovski, L., Flach, P.A., Lavrač, N.: Predictive performance of weighted relative accuracy. In Zighed, D.A., Komorowski, H., Zytkow, J.M., eds.: PKDD 2000, Lyon, France, Springer (2000) 255–264
2. Klösgen, W.: Efficient discovery of interesting statements in databases. Journal of Intelligent Information Systems **4** (1995) 53–69
3. Fisher, D.H.: Knowledge acquisition via incremental conceptual clustering. Machine Learning **2** (1987) 139–172
4. Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., Freeman, D.: Autoclass: A bayesian classification system. In Laird, J.E., ed.: ICML 1988, Ann Arbor, Michigan, USA, Morgan Kaufmann (1988) 54–64
5. Morishita, S., Sese, J.: Traversing itemset lattices with statistical metric pruning. In: Proceedings of the Nineteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Dallas, Texas, USA, ACM (2000) 226–236

6. Bay, S.D., Pazzani, M.J.: Detecting group differences: Mining constrast sets. Data Mining and Knowledge Discovery **5** (2001) 213–246
7. Gluck, M.A., Corter, J.E.: Information, uncertainty, and the utility of categories. In: Proceedings of the 7th Annual Conference of the Cognitive Science Society, Irvine, California, USA, Lawrence Erlbaum Associate (1985) 283–287
8. Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. Commun. ACM **39** (1996) 58–64
9. Raedt, L.D.: A perspective on inductive databases. SIGKDD Explorations **4** (2002) 69–77
10. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning **3** (1989) 261–283
11. Lavrač, N., Flach, P.A., Zupan, B.: Rule evaluation measures: A unifying view. In Džeroski, S., Flach, P.A., eds.: ILP 1999, Bled, Slovenia, Springer (1999) 174–185
12. Horst, R., Tuy, H.: Global Optimization - Deterministic Approaches. Springer (1996)
13. Sese, J., Morishita, S.: Itemset classified clustering. In Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D., eds.: PKDD 2004, Pisa, Italy, Springer (2004) 398–409
14. Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
15. Frank, E., Witten, I.H.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)
16. Rand, W.M.: Objective criteria for evaluation of clustering methods. Journal of the American Statistical Association **66** (1971) 846–850
17. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In Shavlik, J.W., ed.: ICML 1998, Madison, Wisconsin, USA, Morgan Kaufmann (1998) 144–151
18. Talavera, L.: Dynamic feature selection in incremental hierarchical clustering. In de Mántaras, R.L., Plaza, E., eds.: ECML 2000, Barcelone, Catalonia, Spain, Springer (2000) 392–403
19. Cardie, C.: Using decision trees to improve case-based learning. In: ICML 1993, Amherst, Massachusetts, USA, Morgan Kaufmann (1993) 25–32
20. Flach, P.A., Lachiche, N.: Confirmation-guided discovery of first-order rules with Tertius. Machine Learning **42** (2001) 61–95
21. Murthy, S.K.: On Growing Better Decision Trees from Data. PhD thesis, John Hopkins University, Baltimore, Maryland, USA (1997)