Efficient Algorithms for Finding Richer Subgroup Descriptions in Numeric and Nominal Data

Michael Mampaey*†, Siegfried Nijssen*‡, Ad Feelders†, and Arno Knobbe*
*LIACS, Leiden University, The Netherlands
†Utrecht University, The Netherlands
‡KULeuven, Belgium

***Abstract*—Subgroup discovery systems are concerned with finding interesting patterns in labeled data. How these systems deal with numeric and nominal data has a large impact on the quality of their results. In this paper, we consider two ways to extend the standard pattern language of subgroup discovery: using conditions that test for interval membership for numeric attributes, and value set membership for nominal attributes. We assume a greedy search setting, that is, iteratively refining a given subgroup, with respect to a (convex) quality measure. For numeric attributes, we propose an algorithm that finds the optimal interval in linear (rather than quadratic) time, with respect to the number of examples and split points. Similarly, for nominal attributes, we show that finding the optimal set of values can be achieved in linear (rather than exponential) time, with respect to the number of examples and the size of the domain of the attribute. These algorithms operate by only considering subgroup refinements that lie on a convex hull in ROC space, thus significantly narrowing down the search space. We further provide efficient algorithms specifically for the popular Weighted Relative Accuracy quality measure, taking advantage of some of its properties. Our algorithms are shown to perform well in practice, and furthermore provide additional expressive power leading to higher-quality results.**

***Keywords*-Subgroup discovery, numeric data, nominal data, convex functions, ROC analysis**

I. Introduction

Subgroup discovery is a descriptive local pattern mining task that aims to find subsets of a given dataset where the distribution of a binary target variable is substantially different from its distribution in the whole data, measured by a quality measure [1], [2]. Generally, subgroups are described by conjunctions of conditions on the attributes of the dataset.

As an example, consider a dataset containing the 26 countries participating in the 2010 Winter Olympics [3]. Several attributes such as *population* or *continent* describe the countries, and a binary target indicates whether a country obtained more than ten medals, which holds in 38% of the cases. In this dataset, the subgroup of countries where a Germanic language is spoken, simply described by the condition *language_family* = *Germanic*, consists of ten countries, 60% of which have a positive target value, making this a rather interesting subgroup. The subgroup

$$\textit{language\_family} = \textit{Germanic} \wedge \textit{athletes} \geq 60$$

with a target share of 86% can be considered to be even more interesting. Mature subgroup discovery systems find this description by *specializing* or *refining* the more simple subgroup with an additional condition. As in the above example, usually conditions on nominal an numeric attributes are expressed as equalities and inequalities, respectively.

An important aspect of a subgroup discovery system is how it deals with numeric data. Most pattern mining systems are based on exhaustive search and cannot directly find patterns on numeric data; they require that a coarse-grained discretization is determined in a pre-processing step. Subgroup discovery systems, on the other hand, often operate on numeric data directly. By performing discretization during the search process, they may find patterns that pattern mining systems are incapable of. However, as repeated discretization is expensive, typically a heuristic approach is employed within which the search only continues for the most promising refinements. Efficiently considering a sufficiently wide range of conditions is therefore important in these systems, to ensure that interesting subgroups can still be found.

In this paper we propose an improvement of the state of the art of greedy subgroup discovery, based on the observation that a larger space of conditions allows for finding subgroups of higher quality. It is crucial that these refinements can be found efficiently, so as to ensure the overall feasibility of the search even for large datasets. This improvement consists of enriching the standard subgroup description pattern language with two fundamental types of conditions: intervals for continuous attributes, and value sets for nominal attributes. Our main contribution is that we prove that under reasonable assumptions, the best scoring such refinements can be found without increasing the computational complexity of subgroup discovery systems.

Interval conditions are of the form $\textit{athletes} \in [60, 100]$. They allow us to focus directly on ranges of values of interest. Such conditions are hard to find by traditional algorithms, as they would need to find the conditions $\textit{athletes} \geq 60$ and $\textit{athletes} \leq 100$ in two stages. Instead, we propose a new efficient algorithm that can directly mine the best scoring specialization of a subgroup over all possible intervals. The straightforward approach to find an optimal interval involves $O(N + t^2)$ calculations, where N is the number of examples

in the dataset, and t is the number of thresholds considered; however, we show that an optimum can be found in $O(N+t)$ time, for any convex quality measure; hence, finding an interval is not more expensive than finding an inequality such as *athletes* ≥ 60 and is feasible even on large datasets.

Set-valued conditions are conditions of the form $language\_family \in \{Turkic, Italic, Finno\text{-}Ugric\}$. It is quite likely that using an individual language family does not result in a subgroup of high quality, while a combination does. As such, if we only consider single values of nominal attributes, we miss out on some potentially useful subgroups. To some extent this can be alleviated by imposing a hierarchy on the attribute values, but such a hierarchy would have to be specified by the user and still considerably restricts the number of possible groupings. We can avoid such problems by directly searching for an optimal subset of attribute values. We present an algorithm to address this task, similar to an algorithm by Breiman et al. [4], and prove that with a straightforward optimization, it has linear time complexity. To the best of our knowledge, this is a novel result.

From a theoretical perspective, the above algorithms can be added to a subgroup discovery system without changing its computational complexity. However, as the algorithms are conceptually more involved, it may be harder to implement them efficiently. We address this issue by showing that simpler algorithms can be used for the Weighted Relative Accuracy (WRAcc) measure, the most commonly used quality measure in subgroup discovery.

While our focus in this paper is on extending the pattern language in subgroup discovery, it should be noted that the algorithms presented in this paper can also be used in other contexts, such as rule learning or decision tree induction.

II. Related Work

The subgroup discovery field, and more generally the pattern mining field, is dominated by algorithms that are restricted to discrete data. Typically, subgroup discovery algorithms focus on nominal data, and thus require the pre-processing of numeric data [5], [6], with the associated risk of losing information. The subgroup discovery tool Cortana [7] directly supports numeric attributes, although this is limited to inequalities (rather than interval conditions). The search in Cortana is heuristic, making the discovery of optimal intervals by combining inequalities sub-optimal.

Grosskreutz and Rüping [8] propose an exhaustive algorithm for mining top-*k* subgroups in numeric data. Their approach returns the *k* globally best subgroups using interval descriptions, for a family of quality measures that includes WRAcc. The method is defined purely for numeric attributes, rather than heterogeneous data. Search space pruning is achieved by using a data structure that maintains for each attribute an optimistic estimate for every interval; the size of this data structure is thus quadratic in the number of split points in the worst case. In contrast, the methods presented in this paper focus on finding the optimal local refinement of a given subgroup, and as such are meant to be embedded in a greedy search strategy, which scales better for large numbers of split points, albeit with the downside that the discovered subgroups are not guaranteed to be globally optimal.

Fukuda et al. [9] present two algorithms for mining association rules $A \in [l, r] \Rightarrow C$, for a fixed numeric attribute A and consequent C, in the well-known support-confidence framework, optimizing for one measure whilst using a threshold for the other (in order to avoid trivial rules). These algorithms have linear runtime as well, but are specific to the support and confidence quality measures, and work in a support-positive support space rather than in ROC space.

The topic of this paper is algorithmically closely related to node splitting in decision tree induction [4]. However, the goals of subgroup discovery and classification are distinct: a classifier is a global predictive model, whereas a subgroup is a local descriptive pattern. A good subgroup is therefore not necessarily also a good classification rule or vice versa. Many subgroup discovery quality measures are asymmetric, considering only the subgroup itself, whereas a node split in a decision tree takes into account the impurity of all its child nodes. Moreover, the impurity of a single specialization in a decision tree is typically measured against the class distribution in the parent node, whereas a subgroup's quality is compared against the global target distribution. That being said, from an algorithmic perspective many techniques are readily transferrable between both areas.

A numeric attribute at a given node in a decision tree is usually split into k child nodes, using $k-1$ threshold values (typically $k = 2$). This is slightly different from our context; we use two thresholds (the interval endpoints), but the split is binary, i.e., we separate data points within the interval from those outside of it. It was shown by Elomaa and Rousu [10] that for a convex impurity measure, a threshold in an optimal k-way split never separates two adjacent base intervals with an equal class distribution. The number of potential split points can thus be reduced considerably, in linear time with respect to the number of data points. This result is applicable in our context as well. For $k = 2$, finding the optimal split clearly takes linear time; for $k \geq 3$, the optimal k-way split can be found in quadratic time with respect to the number of split points, using dynamic programming.

For nominal attributes, one can either perform a d-way split on all possible values (where d is the domain size of the attribute), or, as in our approach, one can search for the optimal subset of values (and its complement), to split the node into two child nodes. The algorithm we present here is similar to the node splitting algorithm used by CART [4]. In this paper we provide an intuitive proof of its correctness in ROC space. Our contribution lies in showing that a straightforward optimization lowers the complexity of the algorithm from $O(N + d \log d)$ to $O(N + d)$.

III. Preliminaries and Notation

Throughout this paper, we assume a dataset D containing elements $\vec{x} \in D$ of the form $\vec{x} = (a_1, \ldots, a_k, c)$, where k is a positive integer. We call $\vec{a} = (a_1, \ldots, a_k)$ the *attributes* of $\vec{x}$, and c is called the *target*. The target is assumed to be binary, and each attribute value a_i is taken from a domain $dom(A_i)$. We consider binary, nominal, and numeric attributes, that is, attributes for which, $dom(A) = \{0, 1\}$, $|dom(A)| \in \mathbb{N}_0$, and $dom(A) = \mathbb{R}$, respectively. The size of the dataset is denoted by $N = |D|$. We write the positive part of the dataset as $D^+ = \{\vec{x} \in D \mid c = 1\}$, and its size as $N^+ = |D^+|$. Their counterparts D^- and N^- are defined analogously.

For the definition of a subgroup we need to define *patterns*. A pattern is a function $p : \mathcal{A} \to \{0, 1\}$, where $\mathcal{A} = \prod_{i=1}^{k} dom(A_i)$. A pattern p is said to *cover* a data point $\vec{x}$ if and only if $p(\vec{a}) = 1$. The patterns considered in this paper are described as conjunctions of three types of basic conditions on attributes:

1) equalities: $A = a$, where $a \in dom(A)$,
2) intervals: $A \in]l, r]$, where $l < r \in dom(A)$, and
3) value sets: $A \in V$, where $V \subseteq dom(A)$.

Conditions 1 and 3 are applicable to binary and nominal attributes, whereas condition 2 is applicable to numeric and ordinal attributes.

Definition 1: A *subgroup* corresponding to a pattern p is the bag of data points $G_p \subseteq D$ that are covered by p:

$$G_p = \{\vec{x} \in D \mid p(\vec{a}) = 1\} \ .$$

If no confusion can arise, we omit the subscript p. We write $n = |G|$ for the size of G, and n^+ and n^- for the number of positives and negatives in G, respectively. The complement of a subgroup is written as $\overline{G} = D \setminus G$, and its size as $\overline{n} = |\overline{G}|$.

In order to objectively evaluate a candidate pattern in a given dataset, we use a *quality measure*. For each pattern p in the pattern language $\mathcal{P}$, this is a function that measures how interesting the induced subgroup G_p is.

Definition 2: Given a dataset D, a *quality measure* is a function $\varphi_D : \mathcal{P} \to \mathbb{R}$ that assigns a numeric value to a pattern p.

For convenience, we also write φ_D as a function of n^+ and n^-, the number of positives and negatives in G_p, that is, $\varphi_D(p) = \varphi_D(n^+, n^-)$. Whenever the dataset D is clear from the context, we omit the subscript.

Subgroup discovery is aimed at discovering patterns of high quality. Typically this is done either with a top-k approach or an interestingness constraint $\varphi(p) \geq \varphi_{\min}$, and a minimum support threshold $n \geq minsup$ that guarantees the relative frequency of the subgroups in the dataset. Further constraints may involve properties such as the complexity of the pattern p. In most cases, a subgroup discovery algorithm traverses a search lattice of candidate patterns in a top-down, general-to-specific fashion. The structure of the lattice is determined by a *refinement operator* $\rho : \mathcal{P} \to 2^{\mathcal{P}}$, a syntactic operation which determines how simple patterns can be extended into more complex ones by atomic additions. In our application (and most others), the refinement operator is assumed to be a *specialization operator*: $\forall q \in \rho(p) : p \succeq q$ (p is more general than q).

Below we give the definitions of some commonly used quality measures in subgroup discovery. For more measures see, e.g., Fürnkranz and Flach [11].

Definition 3: Given a dataset D of size N and a pattern p, let $n = |G_p|$, $n^+ = |G_p \cap D^+|$, and $n^- = n - n^+$. Further, let $\overline{n} = N - n$, $\overline{n}^+ = N^+ - n^+$, and $\overline{n}^- = \overline{n} - \overline{n}^+$.
The *Weighted Relative Accuracy* of p is defined as

$$WRAcc_D(n^+, n^-) = \frac{n}{N}\left(\frac{n^+}{n} - \frac{N^+}{N}\right) \ .$$

The *Binomial test* [1] of p is defined as

$$Bin_D(n^+, n^-) = \sqrt{\frac{n}{N}}\left(\frac{n^+}{n} - \frac{N^+}{N}\right) \ .$$

Weighted Relative Accuracy is one of the most used quality measures in subgroup discovery [1], [2]. It strikes a balance between the target divergence of the subgroup and its size. WRAcc is also known under different names in other contexts in data mining, for instance, leverage in association rule mining, see Novak et al. [12]. The Binomial test is similar in form to WRAcc, but is weighted by the square root of the relative support of the subgroup. It can be shown to be order equivalent to the standardized z-score.

We also consider the well-known *Chi squared* (χ^2) and *Information Gain* quality measures.

Definition 4: Let $\theta = N^+/N$. The χ^2 measure is defined

$$\chi^2_D(n^+, n^-) = \frac{(n^+ - \theta n)^2}{\theta n} + \frac{(n^- - (1-\theta)n)^2}{(1-\theta)n} + \frac{\left(\overline{n}^+ - \theta\overline{n}\right)^2}{\theta\overline{n}} + \frac{\left(\overline{n}^- - (1-\theta)\overline{n}\right)^2}{(1-\theta)\overline{n}} \ .$$

The *Information Gain* measure is defined as

$$IG_D(n^+, n^-) = h\left(\frac{N^+}{N}\right) - \frac{n}{N}h\left(\frac{n^+}{n}\right) - \frac{\overline{n}}{N}h\left(\frac{\overline{n}^+}{\overline{n}}\right)$$

where h denotes entropy, which is defined as $h(x) = -x \log x - (1-x)\log(1-x)$.

A useful property for quality measures is *convexity*.

Definition 5: A function $f : \mathbb{R}^k \to \mathbb{R}$ is called convex, if for any $a, b \in \mathbb{R}^k$ and $\lambda \in [0, 1]$, it holds that

$$f(\lambda a + (1-\lambda)b) \leq \lambda f(a) + (1-\lambda) f(b) \ .$$

If $-f$ is convex, f is called concave.

We assume that the quality measures we use are convex, which is true for most common quality measures, including WRAcc, χ^2, and Information Gain. This can be seen by verifying that their second derivatives are positive everywhere.

On the other hand, the Binomial test itself is neither convex nor concave. However, if $Bin(n^+, n^-) \geq 0$, that is, if $n^+/n \geq N^+/N$, it is order equivalent to the convex measure $Bin^2(n^+, n^-)$. Typically, subgroups for which $Bin(n^+, n^-) < 0$ are not of interest (otherwise we can, for instance, invert the target). For our intents and purposes, we can assume the Binomial test to be convex.

IV. Convex Hulls in ROC Space

To get an intuitive insight into subgroups and their quality, we will reason with them in *coverage space*, which is directly related to ROC space [11]. For a dataset D with N^+ positive and N^- negative examples, the corresponding coverage space is the subset $[0, N^-] \times [0, N^+]$ of the plane, as shown in Figure 1. The x-axis represents negative counts, and the y-axis represents positive counts. A subgroup p with counts n^+ and n^- is represented by the *stamp point* (n^-, n^+). An important consideration is that all points in coverage space have integer coordinates. In the following we will identify subgroups with their corresponding points in coverage space. Coverage space is trivially related to ROC space, since the latter is simply a normalized version of the former, rescaled to the unit square. In other words, the coordinates of the aforementioned subgroup p in ROC space would be $(n^-/N^-, n^+/N^+)$. For ease of notation, in the following we will mostly be working in coverage space.

For almost any sensible quality measure, it holds that subgroups that are close to the diagonal are considered uninteresting, since they have approximately the same target distribution as the whole dataset; on the other hand, subgroups closer to $(0, N^+)$ – or sometimes $(N^-, 0)$ as well – are of high quality since their target distribution is very different. For a given quality measure, we can plot its isometrics in coverage space, as shown in Figure 1 for Information Gain.

Given a subset S of $\mathbb{R}^2$, its *convex hull* is defined as the minimal superset of S such that if $x_1, x_2 \in CH(S)$ and

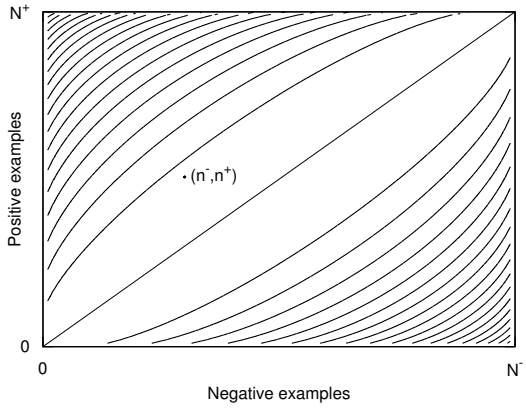


Figure 1: Example of a coverage space, with the isometrics of Information Gain.

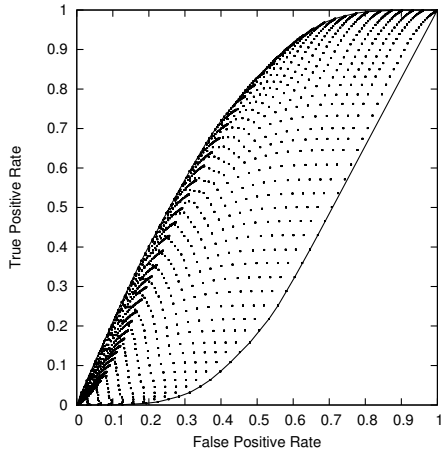


Figure 2: The convex hull of a set of points in ROC space.

$\lambda \in [0, 1]$, then it holds that

$$\lambda x_1 + (1 - \lambda) x_2 \in CH(S) \ .$$

If S is finite (say, a set of stamp points corresponding to a collection of subgroups), it can be seen that $CH(S)$ forms a convex polygon, and thus can be uniquely identified by its non-degenerate vertices, which we shall denote $H(S)$. In other words, $H(S)$ consists of all points on the 'edge' of S. In the remainder, we will call both $CH(S)$ and $H(S)$ the convex hull of S; from the context it should be clear which is meant. For a convex quality measure φ one can verify that

$$\max \{\varphi(s) \mid s \in S\} = \max \{\varphi(s) \mid s \in H(S)\} \ .$$

Hence, in order to find an optimal point, it suffices to evaluate the members of the convex hull $H(S)$, rather than all of S.

An example is given in Figure 2. We took the *Age* attribute from the *Adult* dataset [13], and computed the stamp point of every subgroup $Age \in]a, b]$. Since the attribute under consideration has a cardinality of 74, there are 2 775 possible intervals to consider. However, since only points that lie on the convex hull can maximize a convex quality measure, we only need to consider a small fraction of all intervals. In this example, the convex hull consists of a mere 56 points. If we could directly find those intervals whose stamp points lie on the convex hull, we could save ourselves a lot of effort.

The Graham scan algorithm [14] is one of several convex hull algorithms. Given a set of points S in the plane, it computes the convex hull $H(S)$ in $O(|S| \log |S|)$ time, or in $O(|S|)$ time if S is given in a specific form. In our setting, however, it is desirable to directly construct $H(S)$, without having to consider all points in S, and to do so in time closer to $O(|H(S)|)$ than to $O(|S|)$.

To this end, it is useful to investigate just how large $|H(S)|$ can get. The following property is integral to the complexity results presented below. It provides an upper bound on the size of the convex hull of a set of points in coverage space.

Property 1: Given a finite set of points S in a coverage space of size $N^- \times N^+$, the number of vertices on the convex hull $H(S)$ is bounded by $O(N^{2/3})$.

Proof: The proof focusses on the upper left section H' of the convex hull; the proof for the three remaining parts of $H(S)$ is completely analogous. Let us denote the integer coordinates of the non-degenerate vertices of H' as (x_i, y_i), ordered by x_i, and define $(u_i, v_i) = (x_i - x_{i-1}, y_i - y_{i-1})$, where $(x_0, y_0) = (0, 0)$. It holds that the slopes v_i/u_i of the consecutive edges of H' are positive and strictly decreasing. Consider the closely related set of fractions $\{v_i/(u_i + v_i)\}_i \subset [0, 1]$. It was shown by Calders et al. [15] that the size of a set of fractions in the unit interval with a fixed denominator sum (i.e., the sum of all denominators in the set) is maximized if the set forms a *Farey sequence* [16]. A Farey sequence of order k is defined as the ordered set

$$F_k = \left\{ \frac{a}{b} \;\middle|\; 0 < a \leq b < k,\ gcd(a, b) = 1 \right\} .$$

Calders et al. further showed that asymptotically the size of a Farey sequence $|F_k|$ is $O(q^{2/3})$, where $q = \sum_{a/b \in F_k} b$. Writing $h = |H'|$, in our setting we have

$$\begin{aligned} q = \sum_{i=1}^{h} u_i + v_i &= \sum_{i=1}^{h} x_i - x_{i-1} + y_i - y_{i-1} \\ &= x_h + y_h \\ &\leq N^- + N^+ = N . \end{aligned}$$

Hence, we find that $|H(S)| \in O(N^{2/3})$. ■

What the above property tells us, is that no matter how many candidate stamps points (corresponding to subgroups) there are, the number of convex hull points that we ultimately need to consider depends only on the number of examples in the dataset, and moreover, that this number of points is sublinear in the number of examples.

This result, however, does not tell us *how* to consider only those points. To efficiently compute convex hulls in ROC space, we use the concept of a *Minkowski sum*.

Definition 6: The Minkowski sum of two sets $A, B \subseteq \mathbb{R}^k$ is defined as

$$A \oplus B = \{a + b \mid a \in A, b \in B\} .$$

As an example, Figure 3 illustrates the Minkowski sum of two convex polygons. The sum of the bottom left vertices of A and B, for instance, is the bottom left vertex of $A \oplus B$.

The following useful properties can be observed.

Property 2: For two finite sets A and B, it holds that

$$\begin{aligned} H(A \cup B) &\subseteq H(A) \cup H(B) , \\ CH(A \oplus B) &= CH(A) \oplus CH(B) , \\ |H(A \oplus B)| &\leq |H(A)| + |H(B)| . \end{aligned}$$

In the last case, equality holds if and only if all vertices are non-degenerate.

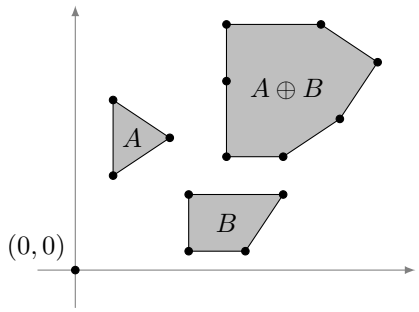


Figure 3: The Minkowski sum of two convex polygons.

Given two convex polygons in the plane, the convex hull of their Minkowski sum is constructed as follows. Assuming the vertices (and edges) of A and B are sorted, say, clockwise, the corresponding edge lists are merged, sorted by their slopes. In Figure 3, each edge of $A \oplus B$ corresponds to an edge of either A or B. The sum $H(A) \oplus H(B)$ has one degenerate hull point and hence $H(A \oplus B)$ contains six out of a maximum of seven points.

V. Algorithms

In this section we present two algorithms to find the optimal refinement of a given subgroup, using intervals for numeric attributes, and value sets for categorical attributes. The basic idea is to directly construct the convex hull of a set of points in ROC space, rather than checking all of them.

A. Intervals for Numeric Attributes

Our aim is to find a subgroup specialization of a given pattern p of the form $p \wedge A \in]l, r]$, where $l < r \in dom\,(A)$, maximizing a convex quality measure φ.

We assume that the set of candidate interval endpoints $T = \{t_i\}_i$ is given as a parameter to the algorithm. These endpoints can just consist of all distinct values occurring in the data. Alternatively, they can be some smaller set obtained through binning in a pre-processing step, or can be manually specified by the user. We further assume that these endpoints are sorted. For equal-width and equal-weight binning, they can be obtained in linear time; if we use all available data points, we can, e.g., sort the attribute once when the data is read, and then reuse the order for each subgroup specialization. As such, we will omit $O(|T| \log |T|)$ from the complexity analysis below.

T induces a partitioning of $dom\,(A)$ consisting of *base intervals* $]t_i, t_{i+1}]$. First of all, some of the endpoints can be pruned beforehand. Event if T contains all possible endpoints, for numeric attributes with low cardinality in the data (as well as for ordinal attributes), it is quite likely that a single value occurs multiple times with different target values. Following Elomaa and Rousu [10], as a pre-processing step we can remove any endpoint t_i for which the adjacent base intervals $]t_{i-1}, t_i]$ and $]t_i, t_{i+1}]$ exhibit the same target distribution,

since these points will never participate in an optimal solution. This pre-processing step takes linear time. In the worst case, however, it may still hold that $|T| \in O(N)$.

The BESTINTERVAL algorithm, given here as Algorithm 1, constructs the convex hull of all interval stamp points in ROC space. Let us write this set as

$$I = \{\,]t, t'] \mid t, t' \in T, t < t' \} \ .$$

First, note that an interval $]t, t']$ can be written as the difference of two half-intervals $]-\infty, t']$ and $]-\infty, t]$. To obtain the convex hull of I, we will decompose I into disjoint subsets, and compute the convex hulls of these subsets, using Minkowski differences of sets of half-intervals. Note that I itself cannot directly be written as a single Minkowski difference of half-intervals due to the constraint $t < t'$.

For the purpose of exposition, assume $|T|$ is a power of two. Let T^k denote the partition of the set $\{\,]-\infty, t] \mid t \in T\}$ into 2^k equal-size bins, for $k = 1, \ldots, \log|T|$ (where the base of the logarithm is 2). Further, let us write T^k_ℓ for the ℓ-th bin, for $\ell = 1, \ldots, 2^k$. Then we define $I^k_\ell = T^k_{2\ell} \ominus T^k_{2\ell-1}$, that is, (slightly abusing notation),

$$I^k_\ell = \left\{\,]t, t'] \mid t \in T^k_{2\ell-1}, t' \in T^k_{2\ell} \right\} \ ,$$

where $\ell = 1, \ldots, 2^{k-1}$. By construction it holds that $t < t'$. We can now decompose I into disjoint subsets as

$$I = \bigcup_{k=1}^{\log|T|} \bigcup_{\ell=1}^{2^{k-1}} I^k_\ell \ .$$

Next we can compute the convex hull of all stamp points in ROC space corresponding to the intervals in I. For ease of notation, we identify these intervals with their stamp points. Let us write $H(T^k_\ell)$ as H^k_ℓ. Using Property 2 we obtain

$$\begin{aligned} H(I) &\subseteq \bigcup_{k=1}^{\log|T|} \bigcup_{\ell=1}^{2^{k-1}} H(I^k_\ell) \\ &= \bigcup_{k=1}^{\log|T|} \bigcup_{\ell=1}^{2^{k-1}} H(T^k_{2\ell} \ominus T^k_{2\ell-1}) \\ &= \bigcup_{k=1}^{\log|T|} \bigcup_{\ell=1}^{2^{k-1}} H(H^k_{2\ell} \ominus H^k_{2\ell-1}) \ . \end{aligned}$$

The BESTINTERVAL algorithm computes the convex hulls of sets of half-intervals bottom-up. That is, we start with the partition $T^{\log|T|}$, where each bin contains a single half-interval (lines 6–7). Thus, it trivially holds that $H^k_\ell = T^k_\ell$ for $k = \log|T|$. Then, for each subsequent k down to 1, using Property 2 and the fact that $T^k_\ell = T^{k+1}_{2\ell-1} \cup T^{k+1}_{2\ell}$, we can compute H^k_ℓ by combining $H^{k+1}_{2\ell-1}$ and $H^{k+1}_{2\ell}$ (lines 16–17). For each adjacent pair of convex hulls of half-intervals, the convex hull of their Minkowski difference is computed to obtain intervals (line 10), and for each of the intervals it is checked whether it maximizes φ (lines 11–15).

Algorithm 1: BESTINTERVAL(p, A, T, φ)

input : subgroup description p, numeric attribute A, sorted endpoints T, convex quality measure φ
output: interval $]l, r]$ with $l, r \in T$ maximizing $\varphi(p \wedge A \in]l, r])$

1 $]l, r] \leftarrow]-\infty, \infty]$
2 $\varphi_{\max} \leftarrow \varphi(p)$
3 **foreach** t_i **in** T **do**
4 compute n^+_i, n^-_i for $p \wedge A \in]-\infty, t_i]$
5 $k \leftarrow \log|T|$
6 **for** $\ell = 1$ **to** 2^k **do**
7 $H^k_\ell \leftarrow \{]-\infty, t_\ell]\}$
8 **while** $k \geq 1$ **do**
9 **for** $\ell = 1$ **to** 2^{k-1} **do**
10 $I^k_\ell \leftarrow H(H^k_{2\ell} \ominus H^k_{2\ell-1})$
11 **foreach** $]t_i, t_j]$ **in** I^k_ℓ **do**
12 $\varphi_{i,j} \leftarrow \varphi(n^+_j - n^+_i, n^-_j - n^-_i)$
13 **if** $\varphi_{i,j} > \varphi_{\max}$ **then**
14 $\varphi_{\max} \leftarrow \varphi_{i,j}$
15 $]l, r] \leftarrow]t_i, t_j]$
16 **for** $\ell = 1$ **to** 2^{k-1} **do**
17 $H^{k-1}_\ell \leftarrow H(H^k_{2\ell} \cup H^k_{2\ell-1})$
18 $k \leftarrow k - 1$
19 **return** $]l, r]$

The computational complexity of the algorithm is as follows. The positive and negative counts for all half-intervals $]-\infty, t]$ are obtained in $O(N + |T|)$ time with a single scan over that data (lines 3–4). Every H^k_ℓ corresponds to a convex polygon in ROC space. By construction, these polygons can be stored in sorted order. As a result, the computations of I^k_ℓ (line 10) and H^{k-1}_ℓ (line 17) can be performed in linear time. The computation of I^k_ℓ is linear in $\left|I^k_\ell\right|$, since it is computed as the Minkowski sum of two convex polygons. The computation of H^{k-1}_ℓ is linear in $\left|H^{k-1}_\ell\right|$, by applying the Graham scan algorithm on the two sorted polygons [14]. Next, for a fixed k, note that since $\left|T^k_\ell\right| = N/2^k$, due to Property 1 we have $\left|H^k_\ell\right|, \left|I^k_\ell\right| \in O((N/2^k)^{2/3})$. Summing over $\ell = 1, \ldots, 2^k$ for a fixed k, we thus find we need

$$O(2^k (N/2^k)^{2/3}) = O(2^{k/3} N^{2/3})$$

computations. Summing over all $k = 1, \ldots, \log|T|$ we obtain

$$O(\sum_{k=1}^{\log|T|} 2^{k/3} N^{2/3}) = O(N^{2/3} |T|^{1/3}) \ .$$

Hence, the total time complexity of the algorithm is

$$O(N + |T| + N^{2/3} |T|^{1/3}) = O(N + |T|) \ .$$

Algorithm 2: BESTVALUESET(p, A, φ)

input : subgroup description p, nominal attribute A, convex quality measure φ
output: value set $V \subseteq dom\,(A)$ maximizing $\varphi(p \wedge A \in V)$

1 $V_{\max} \leftarrow dom\,(A)$
2 $\varphi_{\max} \leftarrow \varphi(p)$
3 **foreach** value v **in** $dom\,(A)$ **do**
4 compute n_v^+, n_v^- for $p \wedge A = v$
5 $\mathbb{V} \leftarrow \bigcup_{n_v^+/n_v^-} \left\{\{v' \mid n_{v'}^+/n_{v'}^- = n_v^+/n_v^-\}\right\}$
6 $V' \leftarrow \emptyset$
7 $(n^+, n^-) \leftarrow (0, 0)$
8 **foreach** V **in** $\mathbb{V}$ decreasing w.r.t. n_V^+/n_V^- **do**
9 $V' \leftarrow V' \cup V$
10 $(n^+, n^-) \leftarrow (n^+ + n_V^+, n^- + n_V^-)$
11 **if** $\varphi(n^+, n^-) > \varphi_{\max}$ **then**
12 $V_{\max} \leftarrow V'$
13 $\varphi_{\max} \leftarrow \varphi(n^+, n^-)$
14 **return** $V_{\max}$

B. Value Sets for Nominal Attributes

Given a subgroup description p and a nominal attribute A, we now consider the problem of finding the best refinement $p \wedge A \in V$, where $V \subseteq dom\,(A)$, with respect to a convex quality measure φ. Let $d = |dom\,(A)|$ be the size of the domain of A. The naive approach is to consider all $O(2^d)$ subsets of $dom\,(A)$, which is clearly infeasible even for relatively small domains.

Let us consider the stamp points of all $p \wedge A \in V$ in coverage space. Since φ is convex, we know that the subset V maximizing φ lies on the convex hull of these points. The BESTVALUESET algorithm, given as Algorithm 2, makes use of the fact that all values $v \in dom\,(A)$ are mutually exclusive. Hence, the stamp point of any value set $V \subseteq dom\,(A)$ can be written as the sum of the stamp points of the individual values $v \in V$. As a result, constructing the convex hull is easy: we compute the positive/negative ratio for each individual value, and then incrementally construct the hull by adding the values one by one in decreasing order with respect to this ratio. For each intermediate value set we compute the quality of the corresponding subgroup, and in the end the best one is reported.

Note that this only constructs the upper part of the convex hull; the lower part is analogously formed by considering values in increasing order. The upper hull thus forms a chain, and the lower hull consists of the upper hull's complements.

The BESTVALUESET algorithm is similar to the value set splitting algorithm used in CART by Breiman et al. [4], who provided a correctness proof based on the convexity of the quality measure. In the ROC space setting, since the algorithm constructs the convex hull greedily, it is conceivable that its solution is suboptimal, i.e., that the constructed hull in fact does not contain all points. Here we present an intuitive geometric proof of correctness in coverage space.

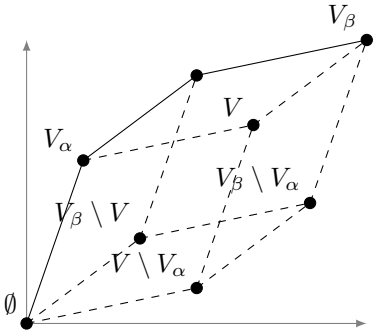


Figure 4: Illustration of the proof of correctness for the BESTVALUESET algorithm. In this example $|dom\,(A)| = 3$.

Property 3: The BESTVALUESET algorithm computes $\arg\max_{V \subseteq dom(A)} \varphi(p \wedge A \in V)$.

Proof: For any $i = 0, \ldots, d$, let V_i be the i-th set constructed by the algorithm. It is easy to see that by construction the sequence $V_0, \ldots, V_d$ is convex. Take any $V \subseteq dom\,(A)$. We will prove that V is either an element of the constructed (upper) hull, or that it lies below it. Consider the sets $V_\alpha = \max_i \{V_i \mid V_i \subseteq V\}$ and $V_\beta = \min_i \{V_i \mid V \subseteq V_i\}$. It holds that $V_\alpha \subseteq V \subseteq V_\beta$. Now, we subtract V_α from each set. If $V \setminus V_\alpha = \emptyset$, then V trivially lies on the constructed hull. Otherwise, it holds that for any v in $V \setminus V_\alpha$, there is a v' in $V_\beta \setminus V$ such that $n_v^+/n_v^- \leq n_{v'}^+/n_{v'}^-$, and vice versa. Hence

$$\frac{\sum_{v \in V \setminus V_\alpha} n_v^+}{\sum_{v \in V \setminus V_\alpha} n_v^-} \leq \frac{\sum_{v' \in V_\beta \setminus V} n_{v'}^+}{\sum_{v' \in V_\beta \setminus V} n_{v'}^-} \,.$$

Since $(V \setminus V_\alpha) \cap (V_\beta \setminus V) = \emptyset$, the points in coverage space corresponding to the sets $\emptyset$, $V \setminus V_\alpha$, $V_\beta \setminus V$, and $(V \setminus V_\alpha) \cup (V_\beta \setminus V) = V_\beta \setminus V_\alpha$ define a parallelogram (see Figure 4). It follows from the inequality above that $V \setminus V_\alpha$ lies under the line segment between $\emptyset$ and $V_\beta \setminus V_\alpha$. If we now translate these points by V_α again, we find that $V = (V \setminus V_\alpha) \cup V_\alpha$ must lie under the line segment between $V_\alpha = \emptyset \cup V_\alpha$ and $V_\beta = (V_\beta \setminus V_\alpha) \cup V_\alpha$. Since both of these points lie on the hull constructed by the algorithm, and this hull is convex, V must lie under this hull as well. ■

We further point out that a straightforward optimization can be applied, by noting that we do not need to check degenerate points on the convex hull. Thus, rather than checking all values v individually, we can group all v having the same ratio n_v^+/n_v^- (line 5). For large d, it is quite likely that several values have the same ratio, and hence this can reduce the number of evaluations (lines 8–13).

It is not difficult to see that without grouping, due to the sorting step the computational complexity of the algorithm is $O(N + d \log d)$. We now show that using this optimization,

the time complexity is reduced. The basic idea is that even for large d (i.e., close to N), sorting is bounded by $O(N)$. A sorting algorithm that properly deals with duplicates takes $O(\eta)$ time per entry on average, where η is the entropy of its input data [17]. Consider the set $\rho = \{n_v^+/n_v^- \mid v \in dom\,(A)\}$ of all distinct ratios, and the distribution μ over ρ, defined as $\mu(r) = |\{v \mid n_v^+/n_v^- = r\}|\,/d$ for $r \in \rho$. We will show that $d \cdot \eta \in O(N)$, where $\eta = -\sum_r \mu(r) \log \mu(r)$. Let $\delta = |\rho| \leq d$, and write $a = d/\delta$. Now, η is maximized if μ is uniform, i.e., if $\mu(r) = 1/\delta$ for all ratios r, and there are a values for each distinct ratio. In the worst case, $\delta \in O\left((N/a)^{2/3}\right)$. As such, we find $d \cdot \eta \leq d \cdot \log \delta \in O(a(N/a)^{2/3} \log(N/a)^{2/3}) = O(N)$. Hence, the total time complexity of the algorithm is $O(N + d)$.

VI. Algorithms for Weighted Relative Accuracy

In this section we provide two algorithms specifically tailored to Weighted Relative Accuracy, one of the most popular quality measures in subgroup discovery. Although these algorithms have the same asymptotic complexity as those in the previous section, we include them here since they are elegant and simple, faster in practice, and because they provide some insight into WRAcc.

The algorithms make use of the additivity of WRAcc.

Property 4: WRAcc is an additive function, i.e., given two patterns p_1 and p_2 with $G_{p_1} \cap G_{p_2} = \emptyset$, it holds that

$$WRAcc(p_1 \vee p_2) = WRAcc(p_1) + WRAcc(p_2)\ .$$

Proof: Denote $n_i = |G_{p_i}|$, $n_i^+ = \left|G_{p_i}^+\right|$ for $i = 1, 2$. Since p_1 and p_2 are disjoint, we know that $|G_{p_1 \vee p_2}| = n_1 + n_2$, and $\left|G_{p_1 \vee p_2}^+\right| = n_1^+ + n_2^+$. Hence

$$\begin{aligned} WRAcc(p_1 \vee p_2) &= \frac{n_1 + n_2}{N}\left(\frac{n_1^+ + n_2^+}{n_1 + n_2} - \frac{N^+}{N}\right) \\ &= \frac{n_1^+ + n_2^+}{N} - (n_1 + n_2)\frac{N^+}{N^2} \\ &= \frac{n_1}{N}\left(\frac{n_1^+}{n_1} - \frac{N^+}{N}\right) + \frac{n_2}{N}\left(\frac{n_2^+}{n_2} - \frac{N^+}{N}\right) \\ &= WRAcc(p_1) + WRAcc(p_2) \end{aligned}$$ ■

A. Intervals for Numeric Attributes

Algorithm 3 finds the optimal interval specialization of a subgroup p, with respect to WRAcc. It has the same linear time complexity as Algorithm 1, but is conceptually simpler and can be performed in a single pass over the data.

The algorithm iterates over the endpoints (line 5), maintaining the best interval encountered so far. Assume that at endpoint t we have two intervals $]t_1, t]$ and $]t_2, t]$, such that the former has a higher quality. Then the following property states that for any future extensions $]t_1, t']$ and $]t_2, t']$, where $t' > t$, their relative difference in quality remains the same, even though their qualities might change.

Algorithm 3: BestIntervalWRAcc(p, A, T)

input : subgroup description p, numeric attribute A, sorted set of endpoints T
output: interval $]l, r]$ with $l, r \in T$ maximizing $WRAcc(p \wedge A \in]l, r])$

```
1  ]l, r] ← ]−∞, +∞]
2  WRAcc_max ← WRAcc(p)
3  h_max ← −∞
4  t_max ← −∞
5  foreach t_i in T in increasing order do
6      compute n⁺_{i−1}, n⁻_{i−1} for p ∧ A ∈ ]t_{i−1}, ∞[
7      h ← WRAcc(n⁺_{i−1}, n⁻_{i−1})
8      if h > h_max then
9          h_max ← h
10         t_max ← t_{i−1}
11     if WRAcc(p ∧ A ∈ ]t_max, t_i]) > WRAcc_max then
12         ]l, r] ← ]t_max, t_i]
13         WRAcc_max ← WRAcc(p ∧ A ∈ ]t_max, t_i])
14 return ]l, r]
```

Property 5: For any interval endpoints t_1, t_2, t, t' such that $t_1, t_2 < t < t'$, it holds that

$$\begin{aligned} &WRAcc(p \wedge A \in]t_1, t]) > WRAcc(p \wedge A \in]t_2, t]) \\ &\Updownarrow \\ &WRAcc(p \wedge A \in]t_1, t']) > WRAcc(p \wedge A \in]t_2, t'])\ . \end{aligned}$$

Proof: Follows directly from Property 4 and the fact that $]t_i, t'] =]t_i, t] \cup]t, t']$. ■

Therefore, at each endpoint t_i only one candidate interval needs to be maintained. Every time a new right endpoint t_i is considered, a new left endpoint t_{i-1} is available, which is checked on lines 6–10. To be able to compare candidate intervals for different right hand sides, we use the quality of their maximal extension, i.e., $]t_{i-1}, \infty[$.

B. Value Sets for Nominal Attributes

Algorithm 4 is a simplification of Algorithm 2 for Weighted Relative Accuracy. Using the additivity of WRAcc, the following property shows that adding a value v with a positive WRAcc to a value set V, increases the total quality.

Algorithm 4: BestValueSetWRAcc(p, A)

input : subgroup description p, nominal attribute A
output: value set $V \subseteq dom\,(A)$ maximizing $WRAcc(p \wedge A \in V)$

```
1 foreach value v in dom(A) do
2     compute n⁺_v, n⁻_v for p ∧ A = v
3 V_max ← {v ∈ dom(A) | n⁺_v/n⁻_v ≥ N⁺/N}
4 return V_max
```

Property 6: Let $V \subseteq dom(A)$ and $v \in dom(A)$ with $v \notin V$, let p be a pattern. If $WRAcc(p \wedge A = v) \geq 0$ then

$$WRAcc(p \wedge A \in V \cup \{v\}) \geq WRAcc(p \wedge A \in V) .$$

Equality holds if and only if $WRAcc(p \wedge A = v) = 0$.

Proof: Follows directly from Property 4. ■

Therefore, it is not necessary to construct and check the entire convex hull. Instead, we can directly construct the optimum: the value set that maximizes WRAcc is the union of all attribute values with non-negative WRAcc. The time complexity of Algorithm 4 is therefore $O(N + d)$.

VII. Experimental Evaluation

In this section we demonstrate the efficiency of the algorithms, and furthermore show that using richer descriptions results in subgroups of higher quality.

Table I presents the characteristics of the datasets we used. We created three synthetic datasets and used three benchmark datasets from the UCI Machine Learning repository [13]. The *Random* dataset consists of one numeric attribute that is independent of the target concept c, with $Pr(c = 1) = 0.5$. The numeric attribute in the *Noisy interval* dataset contains an interval where the target is 1 inside and 0 outside, with 10% random noise added everywhere. The *Farey* dataset corresponds to the worst case convex hull. For each type of synthetic dataset we generated instances with sizes ranging from 10^4 to 10^7. From the UCI ML Repository we used the *Adult*, *CMC*, and *Mushroom* datasets. The last column of Table I shows which class was taken as the positive class, the other class(es) were taken as the negative class.

Table I: Main characteristics of the datasets used in the experiments. Shown are the number of records N, the number of nominal and numeric attributes, and the positive class.

		attributes		
Dataset	# records	nominal	numeric	positive class
Random	10^4–10^7	0	1	$c = 1$
Noisy interval	10^4–10^7	0	1	$c = 1$
Farey	10^4–10^7	0	1	$c = 1$
Adult	48 842	8	6	$income > 50k$
CMC	1 473	7	2	*method 1*
Mushroom	8 124	22	0	*poisonous*

We implemented the algorithms in the open-source Subgroup Discovery tool Cortana [7].[1] All experiments were executed on a system with a 2.4GHz dual quad core processor and 24GB of memory, running Linux. For the synthetic datasets, the minimum support threshold was set to 1. For each benchmark dataset, a minimum support of 10% and maximum refinement depth of 3 were used, the search strategy was beam search with a beam width of 100.

[1] http://datamining.liacs.nl/cortana.html

Table II: Average score of the top 10 discovered subgroups using either standard descriptions (equalities and inequalities), value sets, intervals, or both, for various quality measures, with *minsup*=10%, search depth=3, and beam width=100.

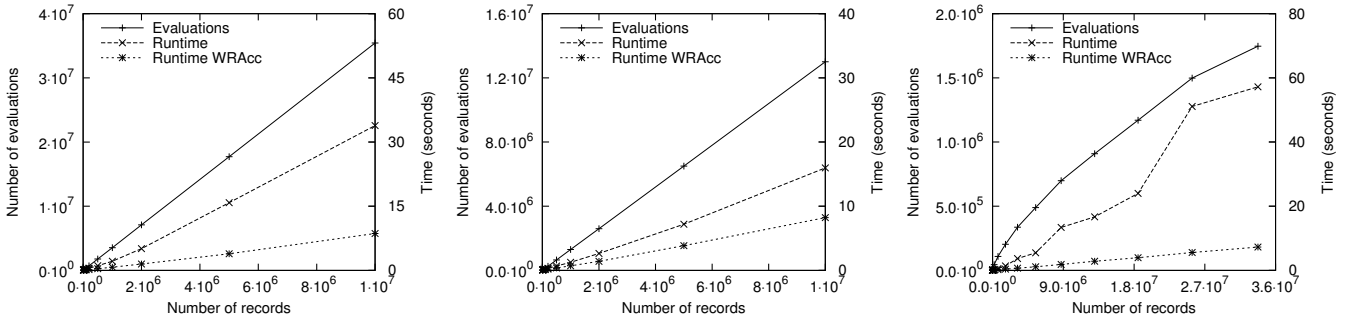
Dataset	Description	WRAcc	Bin	χ^2	IG
Adult	Standard	0.0948	0.1419	9 734.14	0.1426
	Value sets	0.0967	0.1523	10 443.94	0.1619
	Intervals	0.1010	0.1799	11 640.52	0.1727
	Both	0.1020	0.1727	10 639.03	0.1665
CMC	Standard	0.0421	0.0466	65.82	0.0329
	Value sets	0.0486	0.0929	77.60	0.0381
	Intervals	0.0504	0.0519	176.74	0.0782
	Both	0.0506	0.1090	179.45	0.0810
Mushroom	Standard	0.1910	0.2889	4 842.42	0.5067
	Value sets	0.2282	0.3130	6 161.40	0.7101

A. Performance

Figure 5 shows the results of performance experiments with the BestInterval and BestIntervalWRAcc algorithms on the synthetic datasets. For the former, we plot the number of subgroup evaluations and the runtime expressed in seconds (excluding reading the file) as a function of the number of records in the datasets, for the latter the runtime is plotted. Runtimes were averaged over ten runs per dataset and size. All values occurring in the data were used as split points. In the BestInterval algorithm we used Information Gain, but the choice of quality measure does not affect the number of evaluations, and should not have a significant impact on runtime. First, note that the algorithm can handle a million records in just a few seconds, and for the largest datasets takes less than a minute. Figures 5a and 5b show that for *Random* and *Noisy interval* both the number of evaluations and runtime scale linearly with respect to dataset size. For *Farey* (Figure 5c), the number of evaluations scales sublinearly with respect to dataset size, since the number of possible split points grows sublinearly for Farey sequences (Property 1). Runtime also seems to be roughly linear, though it behaves a bit more erratically. The BestIntervalWRAcc algorithm performs very well on all three datasets, scaling linearly in the dataset size, and requiring less than ten seconds to find the optimal interval even for the largest datasets.

B. Quality

Table II contains the results of qualitative experiments on the benchmark datasets. We compared the average quality of the top 10 subgroups, using different types of descriptions: standard (equalities and inequalities), value sets instead of single values, intervals instead of inequalities, and both. As the table shows, using intervals or value sets as descriptions increases the average quality of the discovered subgroups. Using both types of descriptions increases quality as well. Only for the *Adult* dataset does the combination not always yield the best *overall* results, however, this can be attributed to the heuristic search process.



(a) *Random* (b) *Noisy interval* (c) *Farey*

Figure 5: Execution times and number of evaluations of the BESTINTERVAL algorithm, and the execution times of the BESTINTERVALWRACC algorithm, as a function of the number of records.

VIII. CONCLUSIONS AND FUTURE WORK

We presented efficient linear-time algorithms for mining optimal subgroup refinements with respect to a convex quality measure, having descriptions in the form of intervals for numeric attributes, and value sets for nominal attributes. By directly constructing the convex hull of the set of all subgroup stamp points in coverage space, we can disregard vast portions of the search space; the observation that the size of this convex hull is sublinear in the number of examples, entails linear time complexity. Experiments demonstrated that as such, even for large datasets we can efficiently discover high quality subgroups with rich descriptions.

Directions for future work include methods for different types of descriptions, and the extension to more complex targets. Depending on the quality measure, the latter may increase the dimensionality of coverage space, resulting in higher computational complexity.

ACKNOWLEDGEMENTS

This research is partially supported by the Netherlands Organization for Scientific Research (NWO) under project nr. 612.065.822 (Exceptional Model Mining), and by a Postdoc grant from the Research Foundation—Flanders (FWO).

REFERENCES

[1] W. Klösgen, *Handbook of Data Mining and Knowledge Discovery*. New York: Oxford University Press, 2002.

[2] S. Wrobel, "An algorithm for multi-relational discovery of subgroups," in *Proceedings of PKDD*, 1997, pp. 78–87.

[3] B. Pieters, A. Knobbe, and S. Džeroski, "Subgroup discovery in ranked data, with an application to gene set enrichment," in *Proceedings of PL 2010 at ECML PKDD*, 2010.

[4] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[5] B. Kavšek, N. Lavrač, and V. Jovanoski, "Apriori-sd: Adapting association rule learning to subgroup discovery," in *Proceedings of IDA*, 2003, pp. 230–241.

[6] M. Atzmüller and F. Puppe, "SD-Map–A fast algorithm for exhaustive subgroup discovery," in *Proceedings of PKDD*, 2006, pp. 6–17.

[7] M. Meeng and A. Knobbe, "Flexible enrichment with Cortana – software demo." in *Proceedings of BeneLearn*, 2011, pp. 117–119.

[8] H. Grosskreutz and S. Rüping, "On subgroup discovery in numerical domains," *Data mining and knowledge discovery*, vol. 19, no. 2, pp. 210–226, 2009.

[9] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Mining optimized association rules for numeric attributes," *Journal of Computer and System Sciences*, vol. 58, no. 1, pp. 1–12, 1999.

[10] T. Elomaa and J. Rousu, "Efficient multisplitting revisited: Optima-preserving elimination of partition candidates," *Data Mining and Knowledge Discovery*, vol. 8, no. 2, pp. 97–126, 2004.

[11] J. Fürnkranz and P. Flach, "Roc 'n' rule learning—towards a better understanding of covering algorithms," *Machine Learning*, vol. 58, no. 1, pp. 39–77, 2005.

[12] P. Novak, N. Lavrač, and G. Webb, "Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining," *The Journal of Machine Learning Research*, vol. 10, pp. 377–403, 2009.

[13] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[14] R. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information processing letters*, vol. 1, no. 4, pp. 132–133, 1972.

[15] T. Calders, N. Dexters, J. Gillis, and B. Goethals, "Mining frequent itemsets in a stream," *Information Systems*, in press.

[16] J. Conway and R. Guy, "Farey fractions and Ford circles," *The Book of Numbers*, pp. 152–154, 1996.

[17] R. Sedgewick and J. Bentley, "Quicksort is optimal," *Knuthfest, Stanford University, Stanford*, 2002.