MLJ Special Issue on Mining and Learning with Graphs manuscript No.
(will be inserted by the editor)

Efficient Mining for Structurally Diverse Subgraph Patterns in Large Molecular Databases

Andreas Maunz · Christoph Helma · Stefan Kramer

Received: date / Accepted: date

Abstract We present a new approach to large-scale graph mining based on so-called backbone refinement classes. The method efficiently mines tree-shaped subgraph descriptors under minimum frequency and significance constraints, using classes of fragments to reduce feature set size and running times. The classes are defined in terms of fragments sharing a common backbone. The method is able to optimize structural inter-feature entropy as opposed to purely occurrence-based criteria, which is characteristic for open or closed fragment mining. We first give an intuitive explanation why backbone refinement class features lead to a set of relevant features that are suitable for classification, in particular in the area of structure-activity relationships (SARs). We then show that backbone refinement classes yield a high compression in the search space of rooted perfect binary trees. We conduct several experiments to evaluate our theoretical insights in practice: A visualization suggests low co-occurrence and high entropy of backbone refinement class features. By comparison to a class of patterns sampled from the maximal patterns previously introduced by Al Hasan *et al.*, we find a favorable tradeoff between the structural similarity and the resources needed to compute the descriptors. Cross-validation shows that classification accuracy is similar to

This research was supported by the EU seventh framework programme under contract No. Health-F5-2008-200787 (OpenTox [1]).

A. Maunz
Machine Learning Lab, Universität Freiburg
Georges-Köhler-Allee 79
D-79110 Freiburg i. Br., Germany
Tel.: +49-761-203-8442
Fax: +49-761-203-7700
E-mail: maunza@fdm.uni-freiburg.de

C. Helma
in-silico Toxicology
Altkircherstr. 4
CH-4054 Basel, Switzerland
E-mail: helma@in-silico.de

S. Kramer
Institut für Informatik/I12
Boltzmannstr. 3
D-85748 Garching b. München, Germany
E-mail: kramer@in.tum.de

the complete set of trees but significantly better than that of open trees, while feature set size is reduced by > 90 % and > 30 % compared to complete tree mining and open tree mining, respectively. Furthermore, compared to open or closed pattern mining, a large part of the search space can be pruned due to an improved statistical constraint (dynamic upper bound adjustment). This is confirmed experimentally by running times reduced by more than 60 % compared to ordinary (static) upper bound pruning. The application of our method to the largest datasets that have been used in correlated graph mining so far indicates robustness against the minimum frequency parameter, and a cross-validation run on this data confirms that the novel descriptors render large training sets feasible, which previously might have been intractable.

A C++ implementation of the mining algorithm is available at `http://www.maunz.de/libfminer-doc`. Animated figures, links to datasets, and further resources are available at `http://www.maunz.de/mlj-res`.

1 Introduction

The problem of finding frequent and class-correlated patterns in large structured databases, especially graph databases, plays an important role in several areas of science today. In medical and environmental research, for instance, algorithms are now routinely used to automatically derive structural descriptors from databases of molecular structures. These descriptors (features) are then combined statistically in so-called Structure-Activity Relationships (SARs), i.e. models correlating chemical structure with biological activity or chemical reactivity. SAR models have become indispensible tools in many areas of science, from the environmental sciences to pharmacological and toxicological applications.

Current methods for subgraph mining, however, still suffer from scalability problems and, quite related, problems with excessively large solution sets. Most of the predominant approaches employ minimum frequency and possibly statistical correlation criteria such as χ^2 values [4,6,7,10,18]. An increasing minimum frequency tends to favor a higher feature entropy, whereas statistical constraints retrieve subgraphs that are correlated with the target classes. However, the thresholds used still lead to an explosion in the number of frequent patterns, which is a significant drawback of these approaches. As a result, the size of the resulting pattern set is usually multiple times larger than the original database, rendering it useless for subgraph-based classification models and individual inspection by domain experts, at least for very large datasets. In order to build a more sparse representation with a significantly reduced number of patterns, the solution set is often constrained to open[1] or closed features. This might however prune important structural information, since only the support is considered, which may not be sufficient to adequately summarize the diversity of subgraphs.

Hence, the implementation of an explicitly structural summarization strategy (defined intensionally) should yield a higher compression ratio while still retaining all essential information and achieving better running times. Our approach [8], called *backbone refinement class mining* (*BBRC mining*), combines principles from correlated subgraph mining [4,9] with a novel strategy to ensure diversity in structure rather than in subgraph occurrence. We propose to increase structural dissimilarity in order to both increase inter-pattern entropy and decrease the number of solution patterns. A natural property of tree-shaped subgraphs

[1] We decided to use the term 'open' instead of 'free' (which is more common in the data mining literature), because 'free tree' is used as a synonym for 'unrooted tree' in graph theory, see also section 2.1.

(the backbone) is used to represent classes, which renders the set suitable for computational models even for large-scale datasets. We employ the chemical domain as our application area, i.e., we seek fragments of chemical compounds that are associated with a classification endpoint, e.g., carcinogenicity or genotoxicity. The approach is currently limited to the class of tree-shaped fragments. However, in typical databases, they make up about 80-85 % of substructures (see figure 1).

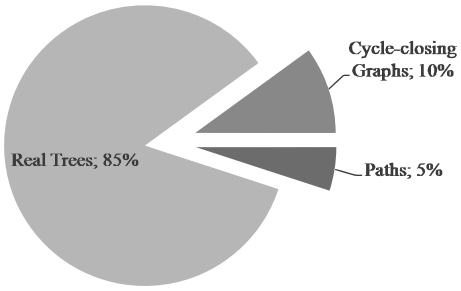


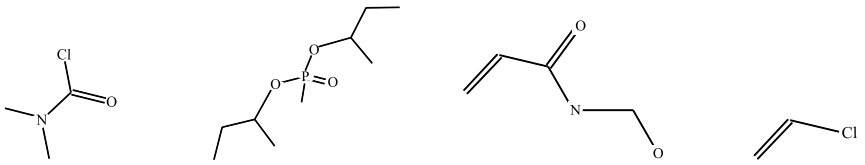
Fig. 1 Typical fragment type distribution in chemical databases. Up to 85 % of substructures are tree-shaped.

1.1 Organization of Paper and Summary of Contributions

Before we conclude this section, we give an outline of the paper and summarize its contributions:[2]

- First, we provide some background in graph theory, structural alerts, and related work (section 2).
- Subsequently, we give an intuition why backbone refinement class features lead to a set of features suitable for classification purposes and explain the reasons for their structural diversity and their efficient computation in practice (sections 3.1 and 3.2).
- Next, we present an algorithm for mining backbone refinement class features using efficient structure- and significance-based pruning (sections 3.3 and 3.4).
- We compare the number of backbone refinement class features to the complete set of subtrees in the search space of rooted perfect binary trees. This shows theoretically the degree of compression of such features (section 4).
- We evaluate the co-occurrence and similarity of backbone refinement class features. The evaluation shows that these features form a well-distributed, structurally dissimilar set of descriptors. This section includes a comparison to the ORIGAMI patterns by Al Hasan *et al.* [2] (section 6).
- We present an empirical analysis of the computation of such descriptors, considering time efficiency, solution set size and predictivity. The method is compared to complete and open trees (Bringmann *et al.* [4]), as well as to the stochastic local search method by Rückert and Kramer [13] (section 7).
- Finally, we present the results of a large-scale analysis with the largest dataset that has been used in correlated graph mining (section 8) so far.

[2] Note that the present paper extends a previous paper [8] by new sections 2.2, 3.2, 4, 6, and extended sections 1, 3.4 and 9. Moreover, some of the material from the previous paper is left out in the present paper.



(a) Acyl halide (b) Ester of phosphonic acid (c) N-methylol derivative (d) Halogenated olefin

Fig. 2 Structural alerts from the Begnini/Bossa rulebase implemented in ToxTree [3].

2 Background and Related Work

2.1 Background: Graph Theory

We assume a graph database $R = (\mathbf{r}, \Sigma, a)$, where $\mathbf{r}$ is a set of graphs, $\Sigma \neq \emptyset$ is a totally ordered set of labels and $a : \mathbf{r} \to \{0,1\}$ is a function that assigns a truth value to every graph in the database (binary classification). Graphs with the same classification are collectively referred to as *target classes*. Every graph $r \in \mathbf{r}$ is a *labeled, undirected graph*, i.e. a tuple $r = (V, E, \Sigma, l)$, where $V \neq \emptyset$ is a finite set of nodes and $E \subseteq V = \{\{v_1, v_2\} \in \{V \times V\}, v_1 \neq v_2\}$ is a set of edges and $l : V \cup E \to \Sigma$ is a label function. The set of all labeled, undirected graphs is referred to as G. An ordered set of pairwise different nodes $\{v_1, \ldots, v_m\}$ is a path between v_1 and v_m, if $\{v_i, v_{i+1}\} \in E, i \in \{1, \ldots, m-1\}$. The *length* of a path is defined as the number of edges it contains. We only consider connected graphs here, i.e., there is a path between each two nodes in the graph. The graph $r = (V, E, \Sigma, l)$ is *double connected*, if there exist two distinct paths between a pair of nodes $\{v_i, v_j\} \in E$. A node v is adjacent to an edge $e = \{e_1, e_2\}$, if $v = e_1$ or $v = e_2$. The number of distinct edges that a node is adjacent to is called its degree. A *subgraph* r' of r is a subset of vertices of r and some edges of r with both endpoints in r', s.t. r' is connected. We denote the subgraph relation by $\subseteq$. If $r' \subseteq r$, then r' is said to *cover* r. This induces a partial order on graphs, the more-general-than relation $\preceq$: for any graphs r, r', s,

$$r' \preceq r, \text{ if } r \subseteq s \Rightarrow r' \subseteq s. \tag{1}$$

The subset of $\mathbf{r}$ that a graph r covers is referred to as the *occurrences* of r, its size as *support* of r in $\mathbf{r}$, denoted by $supp(r, \mathbf{r})$. It is called *open* (*closed*), if for all s with $supp(r, \mathbf{r}) = supp(s, \mathbf{r})$ it holds that $r \subseteq s$ ($r \supseteq s$).

2.2 Background: Structural Alerts

In a typical toxicological setting it is desirable to identify substructures that discriminate between toxic and nontoxic molecules. Commonly they are referred to as *structural alerts*. Figure 2 shows the first four structural alerts by Begnini and Bossa, taken from the carcinogenicity and mutagenicity rulebase implemented in *ToxTree* [3]. Note the variation in size and shape of the patterns, which is quite representative for the whole rulebase. The halogenated olefin contains merely two bonds, whereas the ester is a branched, rather large structure.

In *ToxTree*, those (and many other) patterns are implemented in a decision tree, leading to immediate classification as 'active' in case the candidate molecule contains any of them as a substructure. The fragments reflect relevant biochemical knowledge and are crucial to

screening purposes and consolidation of experimental results in the chemical industry. The feedback we receive from toxicological experts suggests that structural (dis)similarity is of higher interest for rating and classifying chemicals than e.g. topological descriptors.
A use case that has become of vital importance is the prediction of properties of substances without relevant testing information, commonly referred to as *read across* approach. The REACH guidance[3] clearly states that all new and existing chemicals manufactured or imported in the EU at quantities $> 1t/a$ will have to be re-evaluated under REACH legislation and that animal testing has to be omitted wherever possible. Instead, equivalent methods including QSARs and data from structurally related substances have been chosen to be mandatory steps in the process. It is generally believed that structural alerts will play a pivotal role in driving those assessments by filling the gaps in the data.
How should a data mining method be designed that identifies such fragments? If we select them according to statistical criteria (e.g. all substructures that exceed a χ^2 threshold), we will find a large number of redundant substructures that are only minor variants of the same basic motif. Intuitively, we may want to reduce the number of alerts by choosing only the most significant representative from a class of structurally related fragments. Choosing a class representative with the highest statistical significance has several advantages over alternative approaches to reduce the solution space. First of all, we can identify features with the highest discriminatory power for toxic and nontoxic compounds. All other features have lower discriminatory power, because they are either too general or too specific. Open sets may be too general to discriminate efficiently, while closed sets are frequently too specific. Secondly, we can achieve a significant reduction of the search results, without compromising classification accuracy. The reduced number of fragments makes it easier for toxicological experts to inspect search results and to interpret possible molecular mechanisms. Finally, our approach is to use structural information rather than occurrences. This can avoid artifacts resulting from co-occurrences of substructures, but it is also possible to identify cases where a small modification of the structure leads to a large modification of the activity (provided that the training set has a sufficient number of examples).

2.3 Related Work

For an introduction to general tree mining, both theory and applications, see the overview by Munz et al. [20], which targets methods employing minimum frequency as a selection criterion. Enumerating all frequent subtrees in a database can be done efficiently by using *depth-sequences*, see section 3.1. In order to summarize the mined features in the unsupervised setting, i.e., when no target class information is available, most methods consider (only) the support of features. For instance, the solution can be restricted to open (closed) subgraphs of various types (see, e.g., the work of Yan and Han [19]). By definition, these techniques represent refinements with identical occurrences by the most general (the most specific) pattern. They can be easily integrated into graph mining with minimum support.
Rückert and Kramer [13] presented a solution based on occurrence lists, which aims for extensionally diverse sets of structural features. Stochastic local search (SLS) is used to optimize the dissimilarity of features, more specifically, to minimize the dot product between occurrence vectors. The main drawback of the approach is the excessively long running time of SLS to find small sets of diverse features.

[3] REACH: Registration, Evaluation, Authorization and Restriction of Chemicals of the European Union, see `http://guidance.echa.europa.eu/guidance_en.htm`.

Generally, the common strategy in most of the current approaches is to represent a large part of frequent patterns by representatives that form a summary of their occurrences. However, ignoring the wealth of structural information may be a drawback for three main reasons:

1. Occurrence-based representation is not directly related to structure. For instance, refinements with reduced support are not observed together, which might prune important data or could lead to redundant patterns.
2. It is not necessary to distinguish between different subgraph types for occurrence-based methods, e.g., paths are refined to trees arbitrarily without changing the mining strategy. Given the variability of graphs, structural invariants may be desirable.
3. Occurrence summarization methods are forced to mine a representative for any frequent support level, which could impact performance.

The subset of closed subgraphs for which no frequent supergraphs exist is called *maximal* (also known from version space theory as *the positive border*). Al Hasan *et al.* [2] employ sparse representations of maximal frequent subgraphs obtained by sampling. The approach aims for structural diversity (*orthogonality*) of the sampled features while enforcing a certain level of structural similarity (*representativeness*) at the same time. It is a supervised method that repeatedly evaluates candidate sets using a score function.
In contrast to occurrence-based representations, such as open/closed features, we propose to partition the search space directly by structure, as outlined in section 3. The ORIGAMI patterns are defined structurally, and therefore bear some similarity to our descriptors. In this paper, we compare the latter favorably to both methodologies. To support our claims, we will put forward theoretical and empirical evidence for the potential of our method to provide a compressed representation of frequent and significant patterns, and to significantly improve classification accuracy over occurrence-based techniques. Moreover, we show that a structure-aware search can be implemented efficiently and that the method is scalable to datasets larger than those previously considered in correlated class mining.

3 Backbone Refinement Class Mining

We introduce the idea of backbone refinement classes (BBRCs). Then, we formally define our concepts and discuss the actual implementation.

3.1 A New Class of Substructures

Let $p = \{v_1, \ldots, v_m\} \in P$ be a path, then its sequence is defined as the string $l(v_1)\ l((v1,v2))\ \ldots\ l((v_{m-1}\ v_m))\ l(v_m)$, obtained by concatenating node and edge labels along the path. This can be done in a similar fashion for (real) trees by encoding them as *depth-sequences*, following a depth-first walk through the tree, thus building a sequence of nodes and edges with the special property of annotating every node explicitly with its depth. Trees can obviously have several depth sequences.

To enumerate trees uniquely (double-free), Nijssen and Kok [10] use a *canonical depth sequence* formulation that can be used to define a partial order among trees. Descending one level in the partial order is equivalent to adding a node and an edge. Siblings in the partial order can then be compared lexicographically to guarantee double-free enumeration. Every prefix in a canonical depth sequence is also a canonical depth-sequence of the subtree it represents, which guarantees completeness. Moreover, it is efficiently possible to decide

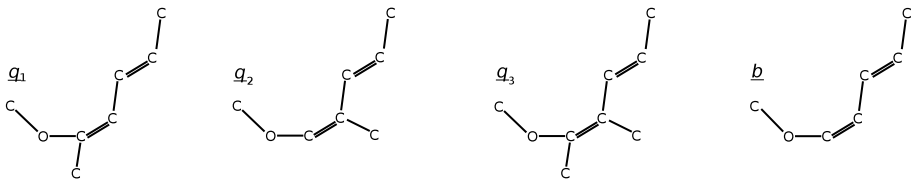


Fig. 3 Three example trees q_1, q_2, and q_3 with the same backbone b. It holds that $q_1 \preceq_b q_2$ and $q_3 \preceq_b q_2$, but neither $q_1 \preceq_b q_3$ nor $q_3 \preceq_b q_1$. Therefore, q_1 and q_3 are not in the same backbone refinement class.

whether the extension of a canonical depth sequence by an edge and node (immediate refinement) will be canonical.

Every tree $t \in T$ has a unique *backbone* $b(t)$, which is defined as the longest path $p \subseteq t$ with the lowest sequence. Nijssen and Kok exploit this and re-label nodes/edges to make the backbone appear always appear first in the canonical ordering for every tree. This allows them to start the search with paths and refine trees from them. An (immediate) tree refinement of $t \in T$ is an addition of an edge and a node to t, such that the result t' is still acyclic, i.e. $t' \in T$. A *backbone refinement* is a tree refinement that is backbone-preserving, i.e. $b(t') = b(t)$.

The following idea is central to backbone refinement class mining: A given backbone spans a maximal tree, i.e., no tree refinement may be added to that tree without changing the backbone. Since every tree has exactly one backbone, by maximally tree-refining all backbones, the partial order of trees is disjointly partitioned across different backbones.

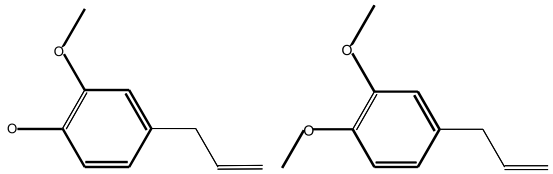
The classes we use are induced by the conjunction of backbone and refinement, hence the name *backbone refinement classes* (*BBRCs*). For example, in figure 3, q_1 : C-C=C-C=C(-C)-O-C and q_2 : C-C=C-C(-C)=C-O-C are in different classes, but q_3 : C-C=C-C(-C)=C(-C)-O-C is in the respective classes of both q_1 and q_2 [4]. Assume that the search starts with the backbone b : C-C=C-C=C-O-C from Figure 3. Assume further that the search then refines to q_1. It may further refine to q_3. However, to enumerate q_2, it will have to backtrack to the backbone position and branch and initiate a new class. Since q_3 is a supergraph of q_1 and q_2, it is in both classes (but it is only enumerated once due to the enumeration strategy).

3.2 The Intuition behind Backbone Refinement Class Mining

According to section 3.1, backbone refinement classes partition the search space structurally, and in section 2.2 we sketched an approach to reduce feature set size based on most significant representatives of such classes. Thus, we propose to mine exactly one representative for every backbone refinement class (if it contains at least one pattern satisfying minimum frequency and significance constraint). We give an example where structural summarization directly detects information that other methods may miss. Figure 4 depicts two structurally similar compounds: eugenol, figure 4(a), and methyleugenol, figure 4(b). Eugenol is used widely as flavoring agent and in medicine as a local antiseptic and anesthetic. Methyleugenol, however, is "*reasonably anticipated to be a human carcinogen*"[5]. In experiments with rodents, methyleugenol induced cancer in multiple organs, especially in

[4] See `http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html` for notation.

[5] See `http://ntp.niehs.nih.gov/ntp/roc/eleventh/profiles/s109meth.pdf` for details.



(a) Eugenol (4-allyl-2-methoxyphenol, Pubchem CID 3314)

(b) Methyleugenol (4-allyl-1,2-dimethoxybenzene, Pubchem CID 7127)

Fig. 4 Small structural modifications turn a harmless substance (left) into a carcinogen (right). The backbones are marked bold.

the liver.

Note that the structure graph of the latter is a refinement of the former. Such deviations might therefore not be detected by methods that represent sets of features by most specific or most general features. In contrast, our structural method is able to detect this case: Due to the backbone change between the two structures, the structures belong to different backbone refinement classes[6] and would therefore be represented differently. It can be argued that the backbone change in the example was selected on purpose. In practice, however, we found it useful in a surprisingly large number of cases. Also, other structural invariants are conceivable, e.g. constraints on branching.

We select the most significant representative from every class induced by the structural criterion. Figure 5 schematically depicts the general search space for two BBRCs starting at the same root node (not disjoint, see section 3.1), which may be a backbone or a real tree. They branch into different directions, corresponding to mutually exclusive sets of features with respect to "$\preceq$" and meet at the maximal tree the corresponding backbone is able to span (note that in general this maximal tree need not exist).

Classes may spread across many support levels, due to their structural definition. However, class representatives are more likely in the lighter regions of figure 5, i.e., patterns with medium support tend to be most significant. Due to the convexity of the χ^2 function, patterns with very low or high support either have too little weight, or their occurrences' class distribution resembles closely the overall database distribution. More formally, for high-support patterns, the weight factor $\frac{x}{n}$ is close to 0, while for low-support patterns y is close to m, and $x-y$ is close to $n-m$, where x (y) is the number of instances (active instances) covered by the pattern, and n (m) is the number of instances (active instances), see section 3.4 for the exact definitions. To summarize, the dichotomy between classes leads to a sparse (by taking only the maximal class member) and structurally diverse (by the structural definition of classes) selection of features. Most BBRC features are sampled from the middle of the search space, where the majority and the most diverse patterns reside. This distribution also allows for early pruning of the search as well as robustness against increasing minimum frequency since there is no need to visit every frequent support level (see section 7.1.1).

3.3 Backbone Refinement Class Mining

We now formally define our concepts. Undirected, labeled graphs are partially ordered. Let $P \subseteq G$ be the set of acyclic connected graphs with degree at most 2 (paths) and let $T \subseteq G$ be

[6] Our approach is currently limited to tree-shaped fragments, but that should not invalidate the argument.

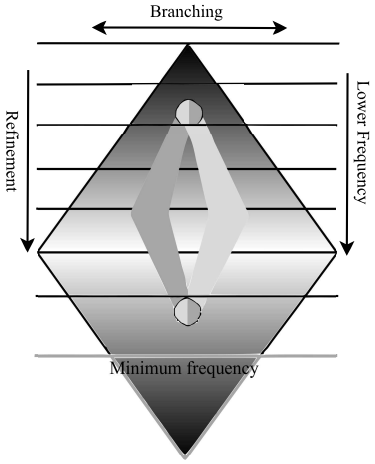


Fig. 5 BBRC search space (case 1) spanning multiple support levels (horizontal lines) as opposed to occurrence-based methods. BBRC features are more probable in lighter regions.

the set of acyclic connected graphs (trees). It holds that

$$P \subset T \subseteq G. \tag{2}$$

Definition 1 (Backbone Refinement Classes) We are considering the *Backbone Refinement Classes* of backbone $b \in P$, denoted by $BBRC_b = \{BBRC_{b_1}, \ldots, BBRC_{b_n}\}$, where each $BBRC_{b_i}$ is the set of trees that are backbone refinements of each other with respect to b, i.e. for all $t, t' \in BBRC_{b_i}$ the following conditions hold:

1. $b(t) = b(t')$ and
2. $t \preceq t'$ or $t' \preceq t$.

The backbone refinement relation given by conditions 1. and 2. is denoted by $\preceq_b$. The set of all backbone refinement classes for a graph database R is called $BBRC_R$.

It follows directly from the definition, that backbone refinement classes are in general not disjoint for the same backbone. More precisely, given two BBRCs $BBRC_{b_i}$ and $BBRC_{c_j}$, they are associated in one of two possible ways: $BBRC_{b_i}$ and $BBRC_{c_j}$ are either

1. not disjoint, iff $b = c$, or
2. disjoint, iff $b \neq c$.

For case 1, members of the two classes $BBRC_{b_i}$ and $BBRC_{c_j}$ are enumerated such that the refinement process starts at the same backbone, but at some later point branches into different directions. It is possible that $BBRC_{b_i}$ and $BBRC_{c_j}$ have a common maximal supergraph (see figures 3 and 5). For case 2, no common elements exist, since b and c occupy disjoint parts of the search space.

The objective of Backbone Refinement Class Representative Mining is to find the most significant representative for each backbone refinement class, formally:

Definition 2 (Backbone Refinement Class Representative Mining) Given a graph database R, a user-defined minimum support f and user-defined minimum χ^2 value u, for all $B \in BBRC_R$, find the most significant $t \in B$ that is *frequent*, i.e. $supp(t,R) \geq f$, and *significant* with respect to occurrence in the target classes, i.e. $\chi^2_t \geq u$. If more than one such patterns exist, return the most general one.

We modified the graph miner Gaston [10] to support BBRC mining[7]. Gaston first enumerates all path refinements, and only thereafter starts enumerating tree refinements growing from all paths, thereby prohibiting backbone changes while applying tree refinements recursively. Furthermore, Gaston uses a very efficient canonical representation for graphs. Specifically, no refinement is enumerated twice. To allow for efficient pruning, we keep BBRCs disjoint once they have branched, i.e., given two BBRCs A and B with feature $a \in A$ and $b \in B$, with $a \not\preceq b$ and $b \not\preceq a$, every c with $a \preceq c$ and $b \preceq c$ is put either in A or B, whichever BBRC is enumerated first. For example, q_3 would be either assigned to the class of q_1 or q_2 in figure 3. The same holds true for the maximal tree in figure 5. However, any pattern more general than the branch point can represent A or B or both.

3.4 Dynamic Upper Bound Pruning

Here, we use the χ^2 distribution test (checking the adherence of a variable to a given distribution) instead of the independence test (checking the statistical independence of variables). For a pattern t, it is defined as

$$\chi^2_t(x,y) = \frac{(y - \frac{xm}{n})^2}{\frac{xm}{n}} + \frac{(x - y - \frac{x(n-m)}{n})^2}{\frac{x(n-m)}{n}}, \tag{3}$$

where $x = supp(t,\mathbf{r})$, and $y \leq x$ is the number of *actives* in the support of t, i.e., with label (say) '1'. Similarly, $n = |\mathbf{r}|$, and $m \leq n$ is the total number of actives in $\mathbf{r}$. It is possible to calculate an upper bound for the χ^2 values of refinements t' of a pattern t (see the work by Morishita and Sese [9]), defined as:

$$\chi^2(x(t'),y(t')) \leq max\ \{\ \chi^2(y(t),y(t)),\ \chi^2(x(t)-y(t),0)\ \}. \tag{4}$$

Following their notation, we use $x(q)$ and $y(q)$ to emphasize the dependency of x and y on q. This upper bound has the antimonotonic property and can thus be used for pruning the refinements of a structure, if their upper bound falls below a given threshold. Using a static, user-defined upper bound threshold u is referred to as *static upper bound pruning* [4]. To speed up the search, we may increase this threshold:

Definition 3 (Dynamic Upper Bound Pruning) For any frequent pattern t, let χ^2_t and $\chi^2_{u,t}$ denote the χ^2 value for t and χ^2 upper bound for refinements of t, respectively. Let $u_{max}(t) = max\{\chi^2(t',R) \mid t' \preceq_b t\}$. Then, if $u_{max}(t) > u$, the user-defined upper-bound threshold u may be increased to $u_{max}(t)$, since we only search for the maximal class element.

The method for mining backbone refinement classes using dynamic upper bound pruning is shown in algorithms 1 and 2. The procedure is invoked as $p.init()$ for any path p of length 1 with $\chi^2_p \geq u$. As can easily be seen, procedure $init()$ starts the search procedure $expand()$ with the longest, i.e., non-refineable, paths and subsequently backtracks to the shorter ones. Thus, every path p satisfying the user defined minimum frequency and significance constraint serves as a backbone.
The tree search procedure *expand()* in algorithm 2 works as follows: Lines 1-4 output the maximal element, if no further refinements are available. Otherwise, for every refinement on the same level, a new class is instantiated for every refinement q'. In line 6, *cmax*, the maximal χ^2 value seen so far (including q') is calculated. Line 7 then implements dynamic

[7] We used version 1.1 (with embedding lists), see `http://www.liacs.nl/~snijssen/gaston/`.

Algorithm 1 *init*():
Calculation of paths as candidate backbones.

Input: A global user defined significance threshold u, a global user defined minimum frequency m
Output: Immediate refinements of current structure that satisfy minimum frequency and significance bounds.

```
1: for all q' ∈ this.ImmediatePathRefinements do
2:     if χ²_q' ≥ u ∧ q'.frequency > m then
3:         q'.init()
4:     end if
5: end for
6: for all q' ∈ this.ImmediateTreeRefinements do
7:     if χ²_q' ≥ u ∧ q'.frequency > m then
8:         q'.expand(q', χ²_q') // q' is a tree
9:     end if
10: end for
```

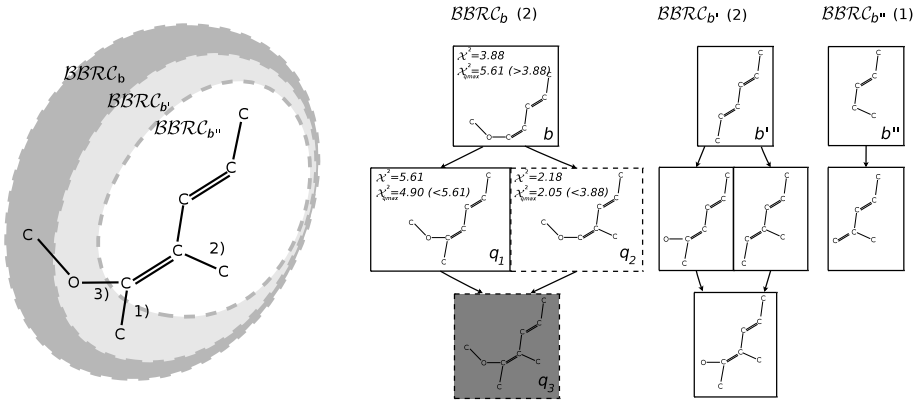
Algorithm 2 $expand(q_{max}, \chi^2_{q_{max}})$:
Implementation of backbone refinement class mining via dynamic upper bound pruning.

Input: A tree q_{max} with significance $\chi^2_{q_{max}} > u$ above the global user defined significance threshold u, a global user defined minimum frequency m, a global variable $updated = true$
Output: The most significant backbone refinement of q_{max}.

```
1: if this.ImmediateTreeRefinements == ∅ { then
2:     print q_max
3:     updated=false
4: end if
5: for all q' ∈ this.ImmediateTreeRefinements do
6:     cmax = max(χ²_q', χ²_qmax)
7:     if χ²_u,q' ≥ cmax ∧ q'.frequency > m then
8:         if χ²_qmax < χ²_q' then
9:             q_max = q'
10:            χ²_qmax = χ²_q'
11:            updated = true
12:        end if
13:        q'.expand(q_max,χ²_qmax)
14:    else
15:        if updated then
16:            print q_max
17:            updated = false
18:        end if
19:    end if
20: end for
```

upper bound pruning by checking q''s upper bound value against *cmax*. If it is lower, the search is truncated and the maximal pattern at that point is output (lines 15-18). Otherwise, it is updated and the iteration continues. Therefore, after including a subgraph as a feature, all the refinements that are in backbone refinement relation to it are pruned away.

Figure 6 visualizes the refinement process for tree q_3 by continuing the example from figure 3. Figure 6(a) indicates BBRC boundaries due to condition 1. from definition 1 by dashed borders, while figure 6(b) shows how the three different non-refineable paths in q_3



(a) q_2

(b) Paths b, b', b'' and associated BBRCs.

Fig. 6 Left: Backbone Refinement Classes for q_2 from figure 3 with numbered edges 1), 2), and 3). Dashed borders indicate their boundaries due to condition 1. from definition 1. Right: Associated search space for non-refineable paths b, b', and b''. Insignificant (95% $\sim$ 3.84) nodes are dashed, pruned nodes are gray.

are used as backbones while the structure is explored. BBRC sizes are given in brackets. Specifically,

– $BBRC_b$ contains two backbone refinement classes corresponding to the exclusive inclusion of either edge 1) or 2), $BBRC_{b_1}$ and $BBRC_{b_2}$, with 4 subgraphs in total: $BBRC_{b_1} = \{b, q_1, q_3\}$, $BBRC_{b_2} = \{b, q_2, q_3\}$.
We detail the expansion and pruning process for $BBRC_b$. Note the associated χ^2 and upper bound ($\chi^2_{q_{max}}$) values in figure 6(b).
Feature b has a χ^2 value of 3.88, and no refinement of b can have a χ^2 value > 5.61. Assume that b is the most discriminative pattern seen so far. Then, $\chi^2_{q_{max}} = 3.88$. Assume further that the original treshold given by the user was $u = 3.84$ ($\approx 95\%$ significance for 1 degree of freedom). Since $5.61 > 3.88$, we expand b.
Feature q_1 is expanded before q_2, since it has the lower canonical depth sequence. It has a χ^2 value of 5.61, increasing the upper bound threshold $\chi^2_{q_{max}}$ to that value, thus making q_1 representative for $BBRC_{b_1}$. However, it has an upper bound value of 4.90, which is below the current threshold of 5.61. Thus it is not expanded.
Feature q_2, the right child, has a χ^2 value of 2.18. Since it is not in backbone refinement relation to the left child, it initiates a new class. However, it can not be representative of that class, since b has the higher χ^2 value. We therefore mark it with a dashed border (Even if b had a lower value, it could not be a representative, since its significance is below u). Thus, b will be the current representative for $BBRC_{b_2}$. Also, it has an upper bound value of 2.05, which is below the current threshold of 3.88. Thus it is not expanded (additionally, in our implementation, q_3 would have already been considered as refinement of q_1 and not be enumerated a second time).
In summary, q_3 (gray) will never be reached, due to dynamic upper bound pruning. More specifically, although it could be a significant feature judging from the q_1 position (its χ^2 value could be $> u$), the upper bound threshold has increased too far already, making q_1 the final representative for $BBRC_{b_1}$ already at that point. For $BBRC_{b_2}$, the representative is b.

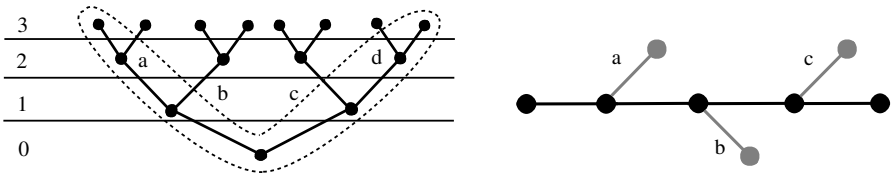
- $BBRC_{b'}$ contains two backbone refinement classes corresponding to the exclusive inclusion of either edge 1) or 3), $BBRC_{b'_1}$ and $BBRC_{b'_3}$, with 4 subgraphs in total:
  $BBRC_{b'_1} = \{$C-C=C-C=C-C, C-C=C-C(-C)=C-C, C-C=C-C(-C)=C(-O)-C$\}$,
  $BBRC_{b'_3} = \{$C-C=C-C=C-C, C-C=C-C=C(-O)-C, C-C=C-C(-C)=C(-O)-C$\}$.
- $BBRC_{b''}$ contains a single backbone refinement class with 2 subgraphs:
  $BBRC_{b''} = \{$C-C=C-C-C, C-C=C-C(=C)-C$\}$.

The class members are enumerated by subjecting paths b, b', and b'' from figure 6(b) to algorithm 2. After backtracking, the same concept is applied to the other (shorter) paths in q_2.

To assess the amount of overhead incurred by our method, running time analysis was performed by comparing its profile to that of the original Gaston algorithm. The results indicate that our algorithm is about 6 % of the time concerned with χ^2 and upper bound calculations, and about 3 % with additional control overhead due to the more sophisticated *expand*() routine. Since running time gains of over 60 % were obtained (see section 7), it may be concluded that the additional effort is justified.

4 Compression of Feature Set Size using Backbone Refinement Classes

This section contains a formula for calculating the number of backbone refinement classes in a rooted perfect binary tree. A *rooted perfect binary tree* is a perfectly balanced binary tree with a designated root node of degree 2. An example of height 3 is shown in figure 7(a). The number of BBRCs is compared to the complete set of subtrees containing the root



(a) (b)

Fig. 7 Left: Rooted perfect binary tree with height 3. A longest path β^* of length 6 has been marked by dashes. It has branches $B_{\beta^*} = \{a,b,c,d\}$, where the subtrees induced by b and c have longest paths of length $\sigma(b) = \sigma(c) = 2$. The path β^* induces $\rho(\beta^*) = 4! = 24$ backbone refinement classes.
Right: A backbone with branches a,b,c (gray) attached.

node. We intend to theoretically show the amount of compression possible with BBRCs. Of course, in real datasets, the search space will most likely not have the perfect binary property. However, we observed that trees for typical carcinogenicity and mutagenicity databases have a branching factor < 2. Therefore, it seems legitimate to simplify the theoretical setting and use binary trees for the estimation of compression. Furthermore, it is intuitive that the relationships described in the following will also hold for higher branching factors.

4.1 Branches and Induced Backbone Refinement Classes

Consider a backbone β of a certain length such as the one in figure 7(b) with length 4. A branch (gray) is either present or not present (figure 7(b) shows all branches present). The number of branches is $\sigma(\beta) = length(\beta) - 1$, where $length(\beta)$ is the number of edges in β. Let the branches be labeled $a, b, c, \ldots$. Consider the set B of branches $\{a, b, c, \ldots\}$ and all its

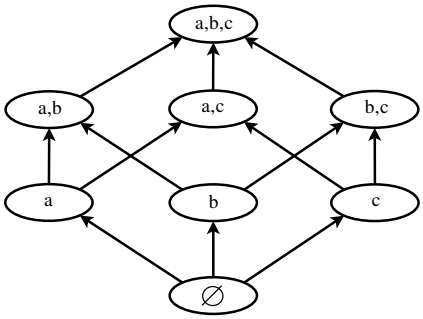


Fig. 8 Set graph corresponding to the subsets of $\{a, b, c\}$ and the partial order induced by the subset relation.

subsets including the empty set. The subsets can be partially ordered in a directed set graph according to "$\subset$", such as shown in figure 8 for the set $\{a, b, c\}$.

Note that there is alway a node corresponding to the full set of branches (with *indeg* $\sigma(\beta)$ and *outdeg* 0), as well as a node corresponding to the empty set (with *indeg* 0 and *outdeg* $\sigma(\beta)$), where *in(out)deg* of a node is the number of edges going into (emanating from) that node.

Lemma 1 *The number of paths from the empty set node to the full set node in the set graph of a backbone equals the number of backbone refinement classes induced by this backbone. It is* $\rho(\beta) = \sigma(\beta)!$.

Proof Starting at the empty set node, after having traversed i edges, there are $\sigma(\beta) - i$ ways to add exactly one item to the current set, which in combination yields $\sigma(\beta)!$ different paths. Let $B_1, \ldots, B_n$ denote all the subsets of B, including the full and empty sets. Associate $x \in B_i$ with the following meaning: x is present on the backbone. It follows that two sibling nodes in the set tree induce two different backbone refinement classes, since all elements in one class contain a branch that no element from the other class contains (no two trees are refinements of each other).
On the other hand, the subgraph relation between a child and a parent node does not induce a new backbone refinement class, since all branches on the parent are still present on the child. □

4.2 Number of Backbone Refinement Classes

Lemma 2 *The number of backbone refinement classes induced by a subtree of height h with root node on the backbone of a rooted perfect binary tree is recursively given by*

$$b(h) = (h-1)! + (h-1)\sum_{i=1}^{h-1} b(i). \tag{5}$$

Proof From Lemma 1: The number of backbone refinement classes induced by a longest path β of this branch is $\rho(\beta) = (h-1)!$.
Then, for every branch b_i of the branches $b_1, \ldots, b_{h-1}$ of β, we recursively add its number of induced classes. Every branch can be combined uniquely with the $h-2$ other branches, or combined with none, thus appearing in a total of $h-1$ induced classes. □

We are now in the position to state the result of this section.

Theorem 1 *The number of backbone refinement classes in the search space of rooted perfect binary trees of height h is recursively given by*

$$B(1) = 1 \tag{6}$$

$$B(h) = \sum_{i,j=2}^{h} 2^{i-1}2^{j-1} \left[(i+j-2)! + (i-2)\sum_{s=1}^{i-1} b(s) + (j-2)\sum_{t=1}^{j-1} b(t) \right], \qquad h \geq 2 \tag{7}$$

Proof There are $\sum_{i,j=2}^{h} 2^{i-1}2^{j-1}$ paths containing the root in a rooted perfect binary tree of height h. For each pair (i, j), the corresponding path has $i+j-2$ subtree inducing branches (because of the missing branch at the root node) and $(i+j-2)!$ induced backbone refinement classes. Then, for every branch, we add its number of induced classes. On each side of the root, every branch can be combined uniquely with the $i-2$ and $j-2$ other branches, respectively. □

4.3 Number of Subtrees

Szekely and Wang [15] showed that a rooted perfect binary tree with height h has exactly

$$F(h) = \lfloor q^{2^{h+1}} \rfloor - 1 \tag{8}$$

non-empty subtrees containing the root, where $q \approx 1.502837$.

4.4 Comparison of BBRC and Tree Set Sizes

For different values of height h we calculate the number of backbone refinement classes ($B(h)$) according to Theorem 1, as well as the complete tree set size ($F(h)$) according to the result of Szekely and Wang. Figure 9 compares the values for heights 1 to 8. Clearly, F grows much faster than B. Note that there is a double exponential in F which does not occur in B. The log-scaled chart in figure 9 reveals this extra exponential growth of F, while B is nearly linear on the investigated interval. In summary, we have a compression by backbone refinement class descriptors of at least an order of magnitude compared to the full set of subtrees.

h	B(h)	F(h)
1	1	4
2	8	25
3	632	676
4	60 280	458 330
5	11 922 296	2.101e+11
6	4.126e+09	4.413e+22
7	2.138e+12	1.947e+45
8	1.537e+15	3.792e+90

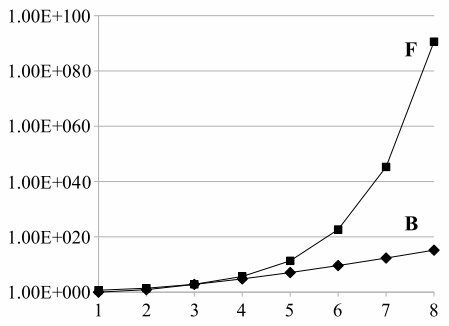


Fig. 9 Comparison of BBRC and full subtree set sizes for heights 1 to 8 (log-scaled). Clearly, F grows much faster than B.

5 Datasets

For the experiments in sections 6 and 7, we used four chemical datasets with binary target classification obtained from the Carcinogenic Potency Database (CPDB) at `http://potency.berkeley.edu/cpdb.html`, version 08/04/29.:

- *Salmonella* Mutagenicity (SM, 388 active / 810 compounds)
- Rat Carcinogenicity (RC, 459 active / 1145 compounds)
- Mouse Carcinogenicity (MoC, 428 active / 927 compounds)
- Multicell Call (MuC, 553 active / 1067 compounds).

For the large-scale analysis in section 8, we performed experiments on parts of the NCI Yeast Anticancer Drug Screen datasets at `http://dtp.nci.nih.gov/yacds/download.html` (April 2002 release). This dataset reports growth inhibition of yeast strains when exposed individually to a large number of compounds as compared to solvent only:

- AC-One (stage 0): Total of 87,264 compounds, 12,068 active (active, if growth inhibition of at least 70 % in at least one strain)
- AC-All (stage 0): Total of 87,264 compounds, 5,777 active (active, if growth inhibition of at least 70 % in all strains)
- AC-All (stage 1): Total of 10,924 compounds at the high dose (50 microM), 5,433 active (active, if growth inhibition at least 70 % in all strains)

6 Diversity and Representativeness

In order to back up our intuition from section 3.2, we use the *Salmonella* mutagenicity (SM) dataset to visualize and analyze the distribution of BBRC features on molecular data in a 2-D embedding based on co-occurrence and entropy, indicating occurrence-based similarity. The other three CPDB datasets yield similar results (all available at `http://www.maunz.de/mlj-res`). We also investigate structural similarity for all CPDB datasets and compare the results to a previously described feature type sampled from the positive border of the search space.

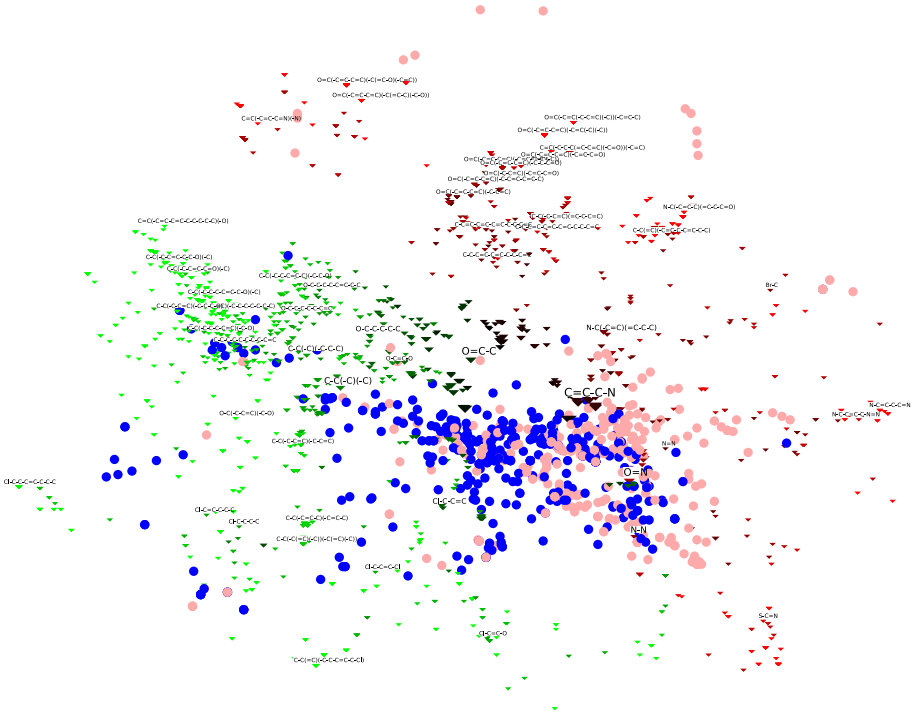


Fig. 10 Euclidean embedding of features and molecules based on co-occurrence and entropy for the *Salmonella* mutagenicity dataset. A flash-animated version is available at `http://www.maunz.de/mlj-res`.

6.1 Co-Occurrence and Entropy

In this section, we use the method of Schulz *et al.* to embed BBRC features and molecules into a 2-D plane [14]. The point coordinates are influenced by:

- Feature-molecule co-occurrence: Features are placed close to the molecules they instantiate.
- Feature-feature co-occurrence: Co-occurring features are placed close to each other.
- The entropy of patterns induced by the target classes of molecules (active/inactive).

For the feature-molecule co-occurrences, an instantiation matrix is used. The negative entropy of the features is multiplied on this matrix to associate molecules with the features that are most significant for the determination of their class, where the entropy is defined as

$$H(q) = -\sum_{c\in\{0,1\}} p_{c,q}\, log\, p_{c,q}, \tag{9}$$

and $p_{c,q}$ is the empirical (induced) probability of feature q being instantiated in target class c. Then, all co-occurrence values are combined in a likelihood function.
Figure 10 shows the Euclidean embedding: The data points consist of features (triangles) and molecules (circles), the distances are found by a local optimization procedure maximizing the log-likelihood of the data. The lower the entropy of a feature the brighter its color, indicating its potential to discriminate between classes (green for active, red for inactive).

We observe that differently colored features are nearly perfectly separated along a diagonal from top left to bottom right. Also, there are very few non-discriminating features (mainly in the center). The features are well distributed across the plane, with few clusters. Thus, BBRC features have a low co-occurrence and high entropy, otherwise there would be more clusters. They are also highly discriminative, as indicated by color brightness. The molecule classes (blue for active, salmon for inactive) are also well separated, following the feature distribution to some extent. The main agglomeration of molecules (below the center) is widely stretched, especially for the active class (green). There are some remarkably distinguished and dense molecule groups in the outer parts, which always consist of members of the same class. Those groups are especially well characterized. This visual evidence, together with the statistical measures from our earlier paper [8], suggest that instances are well-covered by BBRC features.

In summary, it can be stated that the 2-D embedding of molecules and features, combining co-occurrences and entropy information, indicates that BBRC features are, from this perspective, suitable candidates for classification tasks.

6.2 Structural Similarity

We compare BBRC features to the class of "α-*orthogonal*, β-*representative patterns*" introduced by Al Hasan *et al.* and their ORIGAMI approach [2] in terms of structural similarity. They defined the similarity between two features p and q as

$$sim(p,q) = \frac{|m_{pq}|}{max(|p|,|q|)}, \tag{10}$$

where m_{pq} is the maximum common (edge-induced) subgraph of p and q and $|x|$ is the number of edges of x. Therefore *sim* is the size of the maximum common subgraph relative to the larger graph. Al Hasan *et al.* investigated α-orthogonal patterns with pairwise similarity values with $\alpha \in [0.15, 0.35]$. They obtained such feature sets by sampling from the positive border (*maximal patterns*). Running times reported in their analysis were $\approx 2750s$ ($= 45m48s$) for a set of 1000 features obtained from a database of 1084 compounds using a minimum frequency threshold of 2.3 % (pp. 9-10).
Figure 11 shows boxplots of the pairwise BBRC feature similarity distribution for the four CPDB datasets. It gives the median (bold bar in the middle), the upper and lower quartile (box boundaries, comprising 50 % of the data) and therefore also the inter-quartile distance. We also used minimum frequency 2.3 %, individually calculated for each dataset. The boxplots show that, except for the *Salmonella* mutagenicity dataset, BBRC features had a median similarity of < 0.4, and that, in two cases, median values were inside the ORIGAMI interval. The running time for mining the BBRC features on the four datasets were $0.22s$, $0.34s$, $0.24s$, and $0.37s$ for mining 519, 413, 263, and 447 features, respectively.

In summary, the tradeoff between mining time and structural diversity seems favorable. The diversity was obtained with a simple structural criterium in an unsupervised fashion. Additionally, they can be derived efficiently enough to be even used in "on demand"-settings, i.e. in ad-hoc situations.

7 Feature Count, Efficiency of Calculation and Expressiveness

This section investigates the usefulness of backbone refinement class features as descriptors in classification tasks with a focus on efficiency. For fixed minimum frequency and signif-

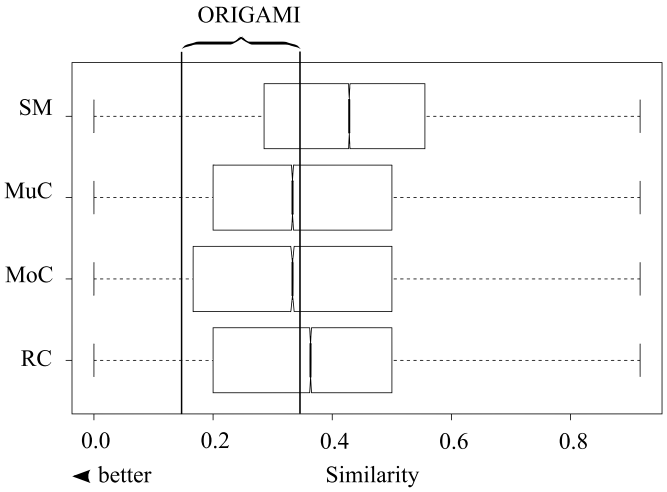


Fig. 11 Pairwise BBRC features similarity distribution for the CPDB datasets.

The median is in the ORIGAMI interval $[0.15, 0.35]$ in two and ≤ 0.4 in three out of four cases.

icance thresholds, we compare them to the complete set of subtrees and open trees (Bringmann *et al.* [4]), as well as to the stochastic local search method by Rückert and Kramer [13].

7.1 Descriptor Computation and Predictivity

We evaluated four types of fragment descriptors according to feature count, running time, predictive accuracy as well as to sensitivity and specificity for the detection of active compounds. The four types are:

1. All Linear Fragments
2. Significant Trees: trees that are frequent and have a minimum χ^2 significance of 95 %.
3. Open Trees: the most general of the trees with the same occurrences from 2.
4. BBRC Representatives: the most significant representatives of the backbone refinement classes from 2.

It should be pointed out that 3. and 4. form a summarization of the features in 2. For the tree-shaped descriptors (2.-4.), we employed a minimum frequency of 6 to avoid excessive numbers of subgraphs, which is well below 1 % of the data set size in all cases. For the linear subgraphs (1.), we applied no minimum frequency and no significance threshold. However, refinement was stopped at frequency 1, i.e., a fragment with single occurrence was included in the set of fragments but not further refined.
Fragment types 1., 2., and 4. were calculated with the proposed approach. For the open trees (3.) we used the method by Bringmann *et al.* [4][8].
Mined subgraphs were evaluated in a leave-one-out crossvalidation. For each fold, significance value calculation and feature selection were done from scratch to avoid information flow between folds. Prediction was performed with a nearest-neighbor technique, using Tanimoto similarity, where each substructural feature was quantified by its χ^2 significance value [5]. A training compound was selected as neighbor, if its similarity to the query structure

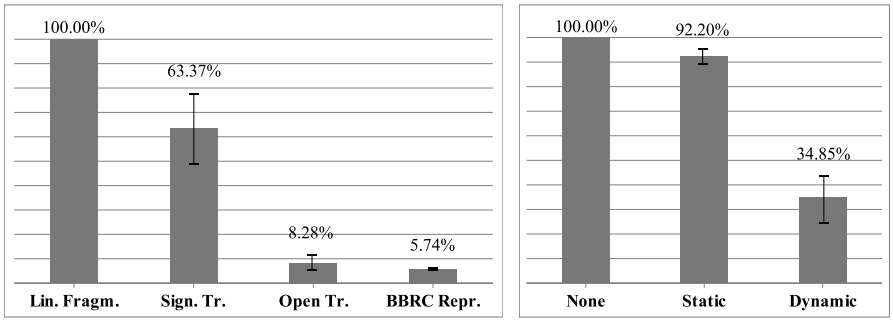
[8] The authors kindly provided us with a binary of their algorithm `sfgm`, pointing out that it may not be optimized for speed and uses a breadth-first search technique known to be memory demanding.

	SM	RC	MoC	MuC
1. All Linear Fragments	48 259	86 300	49 816	70 802
2. Significant Trees	27 093	94 991	22 395	29 970
3. Open Trees	8062	4569	1937	5122
4. BBRC Representatives	2715	5183	3083	3636

Table 1 Feature set sizes for All Linear Fragments, Significant Trees and Open Trees and BBRC Representatives for the four CPDB datasets.

Pruning Type	Applicable To	SM		RC		MoC		MuC	
		s	%	s	%	s	%	s	%
None	1,2,3,4	2.63	100.00	21.23	100.00	3.71	100.00	5.17	100.00
Static	2,3,4	2.55	96.97	21.11	99.43	2.98	80.32	4.76	92.07
Dynamic	4	0.44	16.73	6.63	31.22	2.13	57.41	1.76	34.04

Table 2 Comparison of running time for mining BBRC Representatives using different pruning techniques for the four CPDB datasets.



(a) Feature count mean reduction (b) Runtime mean reduction

Fig. 12 Mean values of Tables 1 and 2, taken for the four CPDB datasets. Feature set size is reduced by $\geq 30\%$ and running time by $\geq 60\%$ compared to Open Trees / Static Upper Bound Pruning.

exceeded 0.3 (this cut-off value was used as default throughout our experiments), and its contribution to the prediction was weighted by its similarity. If no neighbor could be identified, no prediction was made. Otherwise, a minimum of five neighbors was used. Besides the actual classification, a confidence value based on mean neighbor similarity was calculated for every single prediction. Aromatic rings in the structures were represented in Kekulé notation with alternating single and double bonds. In this study, we consider for efficiency reasons only the core chemical structure without hydrogen atoms. Hydrogens attached to fragments can be inferred from matching the fragments back to the training structures.

7.1.1 Effectiveness

Table 1 compares the resulting fragment set sizes for the different fragment types and Table 2 indicates the mining times we obtained for BBRC representatives with different statistical metric pruning techniques. More specifically, those times correspond to BBRC mining using no statistical pruning, static upper bound pruning and dynamic upper bound prun-

	SM			RC		
	all %	AD %	wt. %	all %	AD %	wt. %
1. All Linear Fragments	75.0	77.8	83.0	63.7	67.0	77.5
2. Significant Trees	74.6	**80.7**	**86.8**	64.4	70.0	81.8
3. Open Trees	**75.5**	80.6	84.5	64.5	68.7	80.0
4. BBRC Representatives	74.6	79.4	85.4	**67.2**	**70.4**	**82.2**
	MoC			**MuC**		
	all %	AD %	wt. %	all %	AD %	wt. %
1. All Linear Fragments	67.6	72.7	79.9	69.3	72.4	79.6
2. Significant Trees	**73.3**	75.7	**83.7**	**71.9**	**75.6**	83.5
3. Open Trees	71.5	74.4	80.8	70.2	73.5	81.3
4. BBRC Representatives	71.7	**76.5**	82.0	70.3	74.1	**84.9**

Table 3 Descriptor accuracy for the CPDB datasets, obtained with leave-one-out crossvalidation. Bold figures indicate the best results.

ing, respectively. Mining times for open trees using `sfgm` were as follows: 1,899s (SM), 28,537s (RC), 1,744s (MoC), and 2,594s (MuC). Figure 12 summarizes the information from Tables 1 and 2 by mean values of relative feature count and running time reduction across the different datasets. These empirical findings suggest that fragment set sizes may be reduced by 94 %, 91 % and 31 % through the use of BBRC representatives compared to linear subgraphs, significant trees and open fragments, respectively. Furthermore, the application of dynamic upper bound adjustment is associated with a reduction in running time by 63.34 % and 60.92 % compared to using no statistical pruning and static upper bound pruning, respectively. Considering that open trees cannot use dynamic upper bound adjustment, static upper pruning applies to them (see also section 8.2 and the work of Bringmann *et al.* [4]). The "no pruning" setting corresponds to ordinary fragment search with only minimum frequency as antimonotonic constraint, i.e., to the original Gaston implementation. Performance comparisons in this class-blind setting, e.g., to gSpan, can be found in the literature [11, 17].

7.1.2 Predictivity

This section compares accuracy, sensitivity and specificity of descriptors 1.-4. in a binary classification task. Sensitivity (specificity) measures the proportion of target class positives (target class negatives) which are correctly identified as such. Accuracy is the overall fraction of correct predictions. The figures reported in this section are derived as follows: *all* includes all unweighted predictions, *AD* (or *Applicability Domain predictions*) considers the top 80 % unweighted predictions as ranked by confidence [5], whereas *wt.* includes again all predictions, but this time the contribution of every prediction is weighted by its associated confidence value. The rationale for this measure is that errors of high-confidence predictions should be penalized more heavily than errors of low-confidence predictions. Therefore, *wt.* combines and aggregates both types of information into one measure.
Table 3 compares the accuracy values of BBRC representatives to all linear fragments, significant trees, and open significant trees. Table 3 shows that tree-shaped subgraphs always perform better than linear subgraphs, whether for all or AD predictions or for weighted accuracy. BBRC representatives outperform open trees in 10 out of 12 cases. The mean accuracy difference between BBRC representatives and significant trees is -0.27 $\pm$1.47, whereas it is 1.1 $\pm$ 1.44 compared to open trees and 2.77 $\pm$ 1.66 compared to linear fragments, respec-

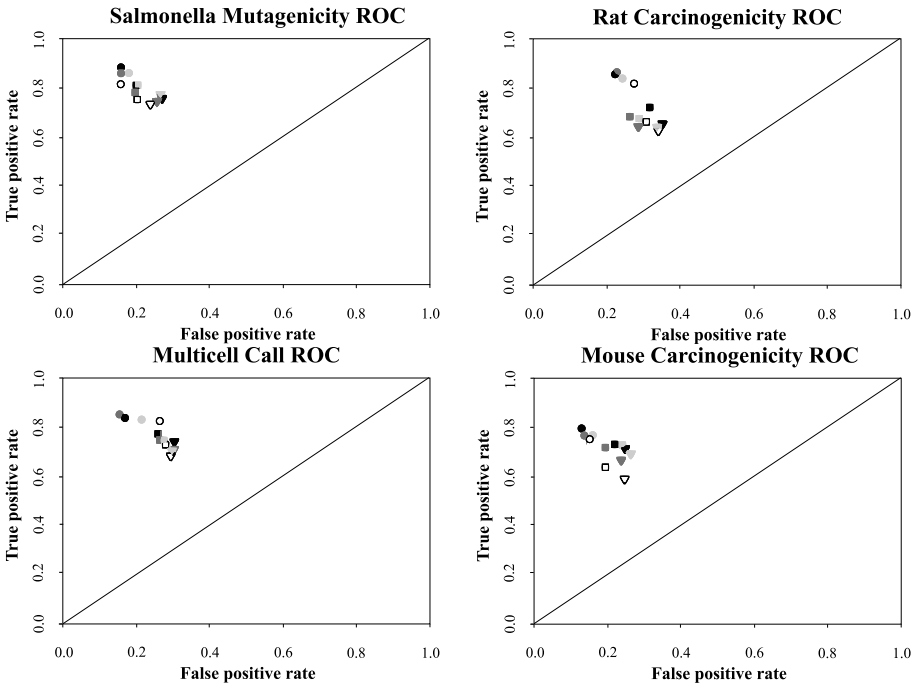


Fig. 13 ROC plots for the CPDB datasets comparing Significant Trees (black), BBRC Representatives (dark gray), Open Trees (light gray) and All Linear Fragments (hollow). Circles denote predictions weighted by confidence, squares predictions in applicability domain, and triangles all predictions.

	all	AD	wt.	#Features	Time
NCTRER	0.74	0.79	0.82	1782	0.65
Bloodbarr	0.72	0.75	0.84	616	0.22
Yoshida	0.55	0.59	0.58	377	0.16

Table 4 Accuracy table for datasets used in the study of Rückert and Kramer obtained by 10-fold crossvalidation.

tively. A paired t-test on the accuracy values revealed that BBRC representatives perform significantly better than open trees ($t = 2.65$, $df = 11$, p-value = 0.02267), while no significant difference between BBRC representatives and the complete set of trees ($t = 0.65$, $df = 11$, p-value = 0.5302) was observed. Figure 13 compares the four different types of descriptors in ROC space, showing the differences in sensitivity and specificity for the prediction of active compounds. There is a trend for better values when tree-shaped fragments are used, clearly signalling the higher information load present in those descriptors. BBRC representatives seem to exhibit a lower false alarm rate compared to open trees. Indeed, a paired t-test on the false positive ratios confirmed that BBRC representatives significantly improved on specificity ($t = -4.60$, $df = 11$, p-value = 0.00077). A significant difference in sensitivity could not be detected. All four test results were confirmed with Wilcoxon signed rank tests [16]. We also evaluated backbone refinement class representatives on the three datasets from the study of Rückert and Kramer [13]. To render the results comparable, we also used 6 % of the dataset size as minimum frequency threshold. After performing 10-

fold crossvalidation, we obtained the accuracies shown in Table 4. Apart from the Yoshida dataset, the results seem to be competitive to the respective figures from the original publication. In particular within the applicability domain, results are very similar to those of the SLS method. The table also gives the mean number of features and the running time for feature calculation per fold. In terms of running times, the method proposed here is much faster, as the construction of the trie for the SLS method typically takes a few minutes, whereas the SLS run itself may take hours [13].

In summary, BBRC representatives exhibit a significantly higher specificity compared to open trees, while overall accuracy seems to be competitive to significant trees. They are more sparse than open trees (reducing feature count), and dynamic upper bound adjustment is multiple times more effective than the static method used for open tree mining in terms of running times.

8 Large-Scale Analysis

We employed publicly available large-scale databases (see section 5) to assess several advanced aspects of BBRCs beyond running time, feature count and predictivity. To the best knowledge of the authors, AC-One (stage 0) is the largest labeled dataset that has been considered in correlated graph mining. The analysis investigates different minimum frequency thresholds and the impact of aromatic perception, i.e., whether the miner uses special labels for carbons and edges that are part of an aromatic ring, or whether Kekulé notation employing alternating single and double bonds is used. Finally, we report on a crossvalidation run using those large-scale data. All experiments were carried out on the same computer as in section 7.

8.1 Dataset Coverage and Class Sizes

Given that BBRCs are partially ordered, it is likely that their representatives occur more frequently than at minimum frequency, since, in general, they are not the most specific fragment of their respective classes (see section 2.3). Ideally, this should give a good coverage of the dataset with few descriptors of a relatively high minimum frequency. Such a reduced set would then allow for fast, memory saving computational models as well as for interpretable representations of the (de)activating substructures of a training set. In particular, for use in a predictive model, it is necessary to assess the coverage and representativeness of descriptors. First, we examined the highest minimum frequency of 200 for the AC-One (stage 0) and AC-All (stage 0) datasets to see how well they would cover the data set, i.e., assessed numbers of descriptors per compound, and compared them to the respective figures for a minimum frequency of 100. It clearly can be observed that the mean coverage does not change much. For instance, the AC-One (stage 0) dataset with aromatic perception has a mean log value of 1.650 for a minimum frequency of 200, and a mean log value of 1.673 for a minimum frequency of 100. A comparison of the median values (1.623 vs. 1.633) showed an even smaller difference indicating a skew to the right for a minimum frequency of 100. In contrast, the effects of missing aromatic perception were much greater. The AC-One (stage 0) dataset without aromatic perception has a mean log value of 1.878 for a minimum frequency of 200 and a mean log value of 1.896 for a minimum frequency of 100. Similar results were obtained for the AC-All (stage 0) data (details omitted).
The conclusion here is that a lower minimum frequency does not increase coverage for the

	AC-one (stage 0)	AC-all (stage 1)
Significant Trees	1 190 763	291 729
Open Trees	?	216 206
Maximal Trees	556 673	148 562
BBRC Representatives	31 450	14 381

Table 5 BBRC feature counts for large-scale datasets AC-One (stage 0) and AC-All (stage 1). '?' indicates that computation terminated with an error.

.

majority of compounds, as it might be expected from the feature count differences presented in the previous section. In contrary, the normality of the corresponding density curves (not shown) indicate clearly that coverage is handled more appropriately by the higher threshold. In this setting, for AC-One (stage 0) [AC-All (stage 0)], the mean number of descriptors per compound is about $10^{1.63} \approx 42.6$ [$10^{1.61} \approx 40.7$] with aromatic ring perception and about $10^{1.91} \approx 81.28$ [$10^{1.82} \approx 66.07$] without.
In summary, consistent with the intuition from section 3.2, the method proved robust in terms of descriptor coverage for the higher threshold, allowing for fast mining with higher frequency thresholds. Moreover, the results provide additional evidence for the coverage capacity of tree-shaped descriptors.

8.2 Crossvalidation

To conclude our experiments, we report on two large-scale crossvalidation runs using aromatic perception. Since a balanced dataset is vital for a nearest-neighbor approach, and in view of the results from earlier sections, we extracted a subset of AC-one (stage 0), composed of all actives and an equal number of inactives sampled randomly from the dataset (2*11,700 = 23,400 compounds). The second run, on AC-All (stage 1), used all actives and inactives (in total, 5,248+5,300 = 10,548 compounds). We compared the number of BBRC representatives to the feature counts of other summarization methods. This time we additionally investigated the set sizes of *maximal patterns* (the positive border as implied by minimum frequency and significance constraints; see the work by Al Hasan *et al.* [2]). For AC-All (stage 1), the extraction of open trees with `sfgm` took $>$ 10h, BBRC representatives took 1m13s. For AC-one (stage 0), the `sfgm` system computing open trees terminated with an error, while BBRC representatives took 4m52s. Table 5 shows BBRC representatives had a very condensed representation of below $\leq 5\%$; maximal and open patterns achieved a reduction up to only $\sim 50\%$; thus, BBRC representatives reduce feature counts much more drastically for these large-scale datasets than for the smaller validation datasets (see section 7).
Indeed, BBRC representatives turned out to be the only practically useful feature type for validation on the (quite powerful) computer we used. A prediction included the derivation of the training set similar to the query instance based on features occurring in the compounds and the calculation of the prediction in the same fashion as in Section 7. With open trees, we obtained impractical prediction times of $> 60s$, whereas BBRC representatives gave a mean of $4.7s$ and $11.1s$, respectively. Also, RAM usage was unacceptable with open trees. Table 6 shows the validation results we obtained.

In summary, we consider the class of BBRC representatives a promising candidate for large-scale structural datasets, rendering them computationally feasible. Judging from our

	All	AD	Weighted
AC-one (st. 0)	68.4	71.9	78.3
AC-all (st. 1)	67.7	71.2	77.5

Table 6 Validation results for large-scale datasets AC-One (stage 0) and AC-All (stage 1), using BBRCs.

results in section 7, we can also be quite confident about the quality of predictions compared to other types of subgraph features.

9 Conclusion

We introduced backbone refinement classes (BBRCs), a particularly useful class of subgraphs for mining databases of chemical compounds. Due to their definition, they are structurally distributed in search space as opposed to features resulting from purely occurrence-based methods. They can be calculated efficiently due to their distribution around the medium-frequent patterns. Thus, not all frequency levels above minimum frequency have to be searched.

We derived a formula for calculating the number of backbone refinement classes for the special class of rooted perfect binary trees. A comparison to the complete set of subtrees containing the root revealed at least an order of magnitude difference. It seems reasonable to assume similar relations also for different types of trees.

We investigated occurrence-based similarity as well as structural similarity of BBRC features, indicating low co-occurrence and high inter-feature entropy, as well as structural dissimilarity. Specifically, we compared the method to a class of features sampled from the positive border supervised by structural dissimilarity. We were able to show that BBRC features fall into the target interval of that method two out of four times and that their similarity is < 0.4 three out of four times. The time needed to calculate those features was $< 1s$, compared to $\approx 45m$ for the other method. Thus, a large fraction of structurally similar features are left out, which cannot be guaranteed by occurrence summarization methods, such as open or closed subgraphs.

The experimental results revealed that the expressiveness of backbone refinement class representatives is significantly higher than that of open trees, because a lower number of features is associated with better accuracy, mainly due to higher specificity, reducing false alarms in classification tasks. The method proves to be highly efficient compared to mining significant and open trees, dramatically reducing running time and number of features mined. For smaller datasets, such as the CPDB databases, it seems even suitable for ad-hoc use, e.g. on-demand calculations.

In our experiments on the largest labeled set of chemical compounds used so far in class-correlated graph mining, we showed that BBRCs can be computed within reasonable time and used in simple predictive learning schemes. In view of the drastic decrease in numbers, they may also be suitable for identifying so-called structural alerts, i.e., chemical substructures that are thought to be particularly toxic.

References

1. Opentox: A Predictive Toxicology Framework. URL `http://www.opentox.org`
See also: Hardy, B., Douglas, N., Helma, C., *et al.*: Collaborative Development of Predictive Toxicology Applications Fifth International Symposium on Computational Methods in Toxicology and Pharmacology Integrating Internet Resources (CMTPI 2009), to appear., Taylor & Francis.

2. Al Hasan, M., Chaoji, V., Salem, S., Besson, J., Zaki, M.: Origami: Mining Representative Orthogonal Graph Patterns. ICDM 2007. Seventh IEEE International Conference on Data Mining Pp. 153–162 (2007)
3. Benigni, R., Bossa, C.: Structure Alerts For Carcinogenicity, and the Salmonella Assay System: A Novel Insight Through the Chemical Relational Databases Technology. Mutation Research/Reviews in Mutation Research 659(3), 248 – 261 (2008). DOI DOI:10.1016/j.mrrev.2008.05.003
4. Bringmann, B., Zimmermann, A., Raedt, L.D., Nijssen, S.: Don't Be Afraid Of Simpler Patterns. In: Proceedings 10th PKDD, pp. 55–66. Springer-Verlag (2006)
5. Helma, C.: Lazy Structure-Activity Relationships (Lazar) for the Prediction of Rodent Carcinogenicity and Salmonella Mutagenicity. Molecular Diversity pp. 147–158 (2006)
6. Jahn, K., Kramer, S.: Optimizing gSpan for Molecular Datasets. In: Proceedings of the Third International Workshop on Mining Graphs, Trees and Sequences (MGTS-2005) (2005)
7. Kramer, S., De Raedt, L., Helma, C.: Molecular Feature Mining in HIV Data. In: KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 136–143. ACM, New York, NY, USA (2001). DOI http://doi.acm.org/10.1145/502512.502533
8. Maunz, A., Helma, C., Kramer, S.: Large-Scale Graph Mining Using Backbone Refinement Classes. In: KDD '09: Proceedings Of The 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Pp. 617–626. ACM, New York, NY, USA (2009). DOI http://doi.acm.org/10.1145/1557019.1557089
9. Morishita, S., Sese, J.: Traversing Itemset Lattice with Statistical Metric Pruning. In: Symposium on Principles of Database Systems, pp. 226–236 (2000)
10. Nijssen, S., Kok, J.N.: A Quickstart in Frequent Structure Mining Can Make a Difference. In: KDD '04: Proceedings Of The Tenth ACM SIGKDD International Conference On Knowledge Discovery And Data Mining, pp. 647–652. ACM, New York, NY, USA (2004)
11. Nijssen, S., Kok, J.N.: Frequent Subgraph Miners: Runtime Don't Say Everything. In: Proceedings of the International Workshop on Mining and Learning with Graphs (MLG 2006, pp. 173–180 (2006)
12. Richard, A.M.: Role Of Computational Chemistry in Support of Hazard Identification (Id): Mechanism-Based SARs. Toxicology Letters 79(1-3), 115 – 122 (1995). DOI DOI:10.1016/0378-4274(95)03363-P
13. Rückert, U., Kramer, S.: Optimizing Feature Sets for Structured Data. In: Stan Matwin and Dunja Mladenic, editors, 18th ECML. Springer (2007)
14. Schulz, H., Kersting, C., Karwath, A.: ILP, the Blind, and the Elephant: Euclidean Embedding of Co-Proven Queries. 19th International Conference on Inductive Logic Programming (ILP 2009) URL `http://www.cs.kuleuven.be/~dtai/ilp-mlg-srl/dokuwiki/doku.php?id=paper:ilp:33`
15. Szkely, L., Wang, H.: On Subtrees of Trees. Advances in Applied Mathematics 34(1), 138 – 155 (2005). DOI DOI:10.1016/j.aam.2004.07.002
16. Wilcoxon, F.: Individual Comparisons By Ranking Methods. Biometrics Bulletin 1(6), 80–83 (1945)
17. Wörlein, M., Meinl, T., Fischer, I., Philippsen, M.: A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, ffsm, and Gaston. In: Proceedings of PKDD, pp. 392–403 (2005)
18. Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02), p. 721. IEEE Computer Society, Washington, DC, USA (2002)
19. Yan, X., Han, J.: Closegraph: Mining Closed Frequent Graph Patterns. In: KDD '03: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 286–295. ACM, New York, NY, USA (2003). DOI http://doi.acm.org/10.1145/956750.956784
20. Y. Chi, R.R. Muntz, S. Nijssen, J.N. Kok. Frequent subtree mining - an overview (2001)