

# Large-Scale Graph Mining Using Backbone Refinement Classes

Andreas Maunz  
Freiburg Center for Data  
Analysis and Modeling (FDM)  
Hermann-Herder-Str. 3  
D-79104 Freiburg i. Breisgau,  
Germany  
maunza@fdm.uni-  
freiburg.de

Christoph Helma  
in-silico Toxicology  
Altkircherstr. 4  
CH-4054 Basel, Switzerland  
helma@in-silico.de

Stefan Kramer  
Institut für Informatik/I12,  
Technische Universität  
München  
Boltzmannstr. 3  
D-85748 Garching b.  
München, Germany  
kramer@in.tum.de

## 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 occurrences, which is characteristic for open or closed fragment mining. In the experiments, the proposed method reduces feature set sizes by >90 % and >30 % compared to complete tree mining and open tree mining, respectively. Evaluation using cross-validation runs shows that their classification accuracy is similar to the complete set of trees but significantly better than that of open trees. Compared to open or closed fragment mining, a large part of the search space can be pruned due to an improved statistical constraint (dynamic upper bound adjustment), which is also confirmed in the experiments in lower running times compared to ordinary (static) upper bound pruning. Further analysis using large-scale datasets yields insight into important properties of the proposed descriptors, such as the dataset coverage and the class size represented by each descriptor. A final cross-validation run confirms that the novel descriptors render large training sets feasible which previously might have been intractable.

A C++ implementation is available at <http://www.maunz.de/libminer-doc/>.

## Categories and Subject Descriptors

G.2.2 [Graph Theory]: Graph Algorithms, Trees; H.2.8 [Database Applications]: Data Mining, Statistical Databases

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'09, June 28–July 1, 2009, Paris, France.

Copyright 2009 ACM 978-1-60558-495-9/09/06 ...\$5.00.

## General Terms

Theory, Experimentation, Performance

## 1. INTRODUCTION AND BACKGROUND

Current methods for subgraph mining 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  $\chi^2$  values [6, 8, 13, 2, 5]. An increasing minimum frequency tends to favor a higher entropy of a pattern occurrence. Statistical constraints retrieve subgraphs that are correlated with the target classes. However, the thresholds used usually 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 often multiple times the size of the original database, which makes it unusable for subgraph-based classification models, at least for very large datasets.

In order to build a more sparse representation with a significantly reduced number of solution patterns, the pattern set is often constrained to open<sup>1</sup> or closed features. This might however prune important structural information, since only the support is considered. In this study, we propose a method to increase inter-pattern entropy and reduce pattern set size by increasing structural dissimilarity. In order to reduce the space of frequent and significant patterns, we use a natural property of tree-shaped subgraphs (the backbone) to represent classes, which renders the set usable 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.

Various types of subgraph patterns, such as paths, trees, and general subgraphs, have been investigated for databases of molecular graphs so far [2]. For the databases used in this study, however, pattern sets contain about 5-10% paths, 80-85% real trees and 10% real cycle-closing graphs, which may be quite representative for many applications. Although there exists an efficient method for mining subgraphs shaped as outerplanar graphs (a strict generalization of trees which

<sup>1</sup>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.

may contain cycles [4]), we restrict ourselves to the largest fragment class, because there is some evidence that cycle-closing graphs may not generally improve performance in SAR models [2].

## 1.1 Problem Formulation

### Graph Databases.

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  $a : \mathbf{r} \rightarrow \{0, 1\}$  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 *labelled, 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  $l : V \cup E \rightarrow \Sigma$  is a label function. The set of all labelled, undirected graphs is referred to as  $\mathcal{G}$ . A set of nodes  $\{v_1, \dots, v_m\}$  is a path between  $v_1$  and  $v_m$ , if  $\{v_i, v_{i+1}\} \in E, i \in \{1, \dots, m-1\}$ . 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  $r, r', s$ ,

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

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  $\text{supp}(r, \mathbf{r})$ . It is called *open* (*closed*), if for all  $s$  with  $\text{supp}(r, \mathbf{r}) = \text{supp}(s, \mathbf{r})$  it holds that  $r \subseteq s$  ( $r \supseteq s$ ).

### Backbone Refinement Classes.

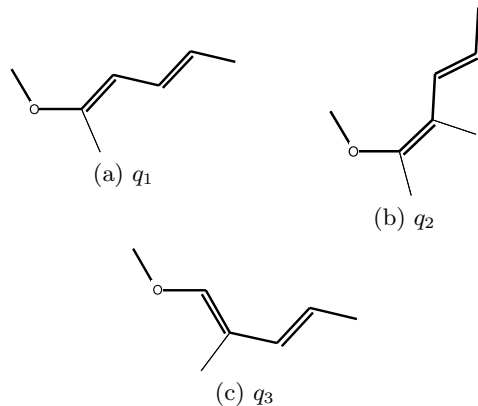
Undirected, labelled graphs are partially ordered. Let  $\mathcal{P} \subseteq \mathcal{G}$  be the set of not double connected graphs with degree at most 2 (paths) and let  $\mathcal{T} \subseteq \mathcal{G}$  be the set of not double connected graphs (trees). It holds that

$$\mathcal{P} \subset \mathcal{T} \subseteq \mathcal{G}.$$

Let  $p = \{v_1, \dots, v_m\} \in \mathcal{P}$  be a path, then its *sequence* is defined as the string  $l(v_1)l((v_1, v_2)) \dots l((v_{m-1}, v_m))l(v_m)$ , obtained by concatenating node and edge labels along the path. Every tree  $t \in \mathcal{T}$  has a *backbone*  $b(t)$ , which is defined as the longest path  $p \subseteq t$  with the lowest sequence according to a lexicographic ordering (described by Nijssen and Kok [8]). A (immediate) tree refinement of  $t \in \mathcal{T}$  is an addition of an edge and a node to  $t$  s.t. the result  $t'$  is still not double connected, i.e.  $t' \in \mathcal{T}$ . A *backbone refinement* is a tree refinement that is backbone preserving, i.e.  $b(t') = b(t)$ .

We are considering the *Backbone Refinement Classes* of  $b \in \mathcal{P}$ , denoted by  $\text{BBRC}_b = \{\text{BBRC}_{b_1}, \dots, \text{BBRC}_{b_n}\}$ , where each  $\text{BBRC}_{b_i}$  is the set of trees that are backbone refinements of each other with respect to  $b$ , i.e. for all  $r, r' \in \text{BBRC}_{b_i}$  it holds that

1.  $b(r) = b(r')$  and
2.  $r \preceq r'$  or  $r' \preceq r$ .



**Figure 1: Three example trees with the same backbone 'C1C2C1C2C101C' (edges are labelled by bond order) in bold. 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.**

We denote the backbone refinement class relation by  $\preceq_b$ . Note that the classes are not disjoint for the same backbone (but they are across different ones). For example, in Figure 1,  $q_1$  and  $q_3$  are in different classes, but  $q_2$  is in the respective classes of both  $q_1$  and  $q_3$ . The set of all backbone refinement classes for a graph database  $R$  is called  $\text{BBRC}_R$ . The objective of Backbone Refinement Class Representative Mining is to find the most significant representative for each backbone refinement class:

### Definition: Backbone Refinement Class Representative Mining (BBRC Mining).

Given a graph database  $R = (\mathbf{r}, \Sigma, a)$ , a user-defined minimum support  $m$  and user-defined minimum  $\chi^2$  value  $p$ , for all  $B \in \text{BBRC}_R$ , find the most significant  $t \in B$  that is *significant*, i.e.  $\chi^2(t, \mathbf{r}) \geq p$ , and *frequent*, i.e.  $\text{supp}(t, \mathbf{r}) \geq m$ .

The complexity of BBRC mining is upper-bounded by the complexity of regular tree mining [8]. We will however show that our approach decreases running times significantly for practical applications.

## 1.2 Related Work

In the following, we review the literature for reducing solution sets in graph mining. For the unsupervised setting, i.e., when no target class is available, methods can only take into account 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 [14]). 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 [10] 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 method is the excessively long running time of SLS to find small sets of diverse features.

The subset of closed subgraphs for which no frequent supergraphs exist is called *maximal* (also known from version

space theory as *the positive border*). Hasan *et al.* [1] employ sparse representations of maximal frequent subgraphs obtained by sampling. The approach aims for structural diversity of the mined features while enforcing a certain level of representativeness at the same time.

The common strategy in most of the 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: First, this type of representation is not directly related to structures but solely to their occurrence, which might prune important data. Second, it is not necessary to distinguish between different subgraph types (e.g., paths are refined to trees arbitrarily without changing the mining strategy). Third, occurrence summarization methods are forced to mine a representative for any frequent support level, which could impact performance. Hence, the implementation of an explicitly structural pruning criterion (defined intensionally) should yield a higher compression ratio with less information loss and better running time.

To obtain diverse patterns homogeneously distributed in structural space, we combine principles from correlated subgraph mining [2, 7] with a novel approach to ensure diversity in structure rather than in subgraph occurrences. We compare the method to an occurrence summarization approach (in this case to open fragments) in terms of feature count, information load, and representativeness.

## 2. METHODS

### 2.1 Statistical metric pruning

Pruning by minimum frequency can be solved with depth first search methods. N.B.: If each subgraph occurred in half of the graphs, a set of  $k$  subgraphs could in principle distinguish  $2^k$  bins of graphs. Since a higher global minimum frequency raises frequencies in at least one target class as well, it leverages a subgraph’s discriminative potential for classification tasks.

Subgraphs that are correlated significantly with the target class can be filtered with statistical measures. Due to the absence of both monotonicity and antimonotonicity, significance values cannot be used for antimonotonic (nor monotonic) pruning directly, however, the convexity of the  $\chi^2$  function allows to derive a related measure for antimonotonic pruning.

In our work, this is done as initially suggested by Morishita and Sese [7], however, we use the  $\chi^2$  distribution test (checking the adherence of a variable to a given distribution) instead of the independence test (checking the statistical independence of variables). In the following, we briefly outline our use of the  $\chi^2$  distribution test and its upper bound for pruning.

	$q$	$all$
$active$	$y$	$m$
$inactive$	$x - y$	$n - m$
$\Sigma$	$x$	$n$

Table 1: Contingency table for subgraph  $q$ .

For a given subgraph  $q$ , a  $2 \times 2$  contingency table counts the occurrences, depending on whether the covered compounds are active or inactive (see Table 1). More specif-

ically,  $y$  is defined as  $|\{r \in \mathbf{r} \mid covers(q, r) \wedge a(r)\}|$ , the count for active compounds containing  $q$ ,  $x$  is defined as  $|\{r \in \mathbf{r} \mid covers(q, r)\}|$ , the count for all compounds containing  $q$ . During calculation of  $\chi^2$  significance values, a weight  $\frac{x}{n}$  is assigned to every candidate pattern  $q$ , corresponding to its support. Subgraphs with low support get a low weight, while subgraphs with high support are unlikely to deviate much from the sample mean. The regions of interesting patterns therefore lie “somewhere in the middle”.

It holds that  $0 \leq y \leq n$  and  $0 \leq x - y \leq n$ . We are now able to check whether the distribution of  $q$  differs significantly from the distribution of all subgraphs. The  $\chi_d^2$  function for distribution testing is defined as

$$\chi_d^2(x, y) = \frac{(y - \frac{xm}{n})^2}{\frac{xm}{n}} + \frac{(x - y - \frac{x(n-m)}{n})^2}{\frac{x(n-m)}{n}}, \quad (1)$$

(see the work of Morishita and Sese [7]). Following their notation, we use  $x(q)$  and  $y(q)$  to emphasize the dependency of  $x$  and  $y$  on  $q$ . They showed that, for any subgraph  $q$ ,  $x(q)$  and  $y(q)$  allow to calculate an upper bound for the  $\chi_d^2$  value of  $q'$ , for all refinements  $q'$  with  $q \preceq q'$ :

$$\chi_d^2(x(q'), y(q')) \leq \max \{ \chi_d^2(y(q), y(q)), \chi_d^2(x(q) - y(q), 0) \}$$

Therefore, the upper bound measure is antimonotonic and may be used for pruning the search space with fragment refinement methods. Bringmann *et al.* [2] demonstrated the application to graph mining while comparing the expressiveness of different subgraph types (paths, trees, and graphs). In the following, this pruning technique with a static, user-defined upper bound threshold is referred to as *static upper bound pruning*.

### 2.2 Dynamic Upper Bound Pruning

#### Enumeration scheme.

The partial order  $\mathcal{P} \subset \mathcal{T} \subseteq \mathcal{G}$  of graphs allows for partitioning the subgraph mining process into three different phases or levels according to the type of subgraph mined. The approach proposed here is concerned only with the set  $\mathcal{T}$ . Thus, no refinement yields a double connected graph and we only consider path and tree refinements.

We modified the graph miner Gaston [8] to support BBRC mining<sup>2</sup>. Two specific properties allow for an efficient BBRC mining implementation on top of Gaston:

- Predominantly, modern graph miners enumerate subgraphs canonically. For the implementation of the Minimum DFS code used in gSpan see also Yan and Han [13]. Gaston uses a similar canonical code representation for graphs that enables checking for allowable path and tree refinements in constant time. Specifically, no refinement is enumerated twice, e.g.,  $q_2$  in Figure 1 (for details, see the work of Nijssen and Kok [8, 9]).
- Gaston first enumerates all frequent path refinements  $\mathcal{P}_{|R}$ , and only thereafter starts enumerating all tree refinements  $\mathcal{T} \setminus \mathcal{P}_{|R}$  growing from all  $p \in \mathcal{P}_{|R}$ . This is performed by using  $p$  as backbone of the innate tree, i.e., by prohibiting backbone changes while applying tree refinements recursively.

<sup>2</sup>We used version 1.1 (with embedding lists), see <http://www.liacs.nl/~snijssen/gaston/>.

ALGORITHM 1. *expand()* function

**Input:** A tree  $q_{max}$  with significance  $\chi^2(q_{max}, \mathbf{r}) > u$  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  $q_{max}, \mathbf{r}$ .

```

1 function expand(Tree  $q_{max}$ , Float  $\chi^2(q_{max}, \mathbf{r})$ ) {
2   If this.ImmediateTreeRefinements ==  $\emptyset$  {
3     print  $q_{max}$ 
4     updated=false
5   }
6   foreach Tree  $q' \in$  this.ImmediateTreeRefinements {
7      $cmax = \max(\chi^2(q', \mathbf{r}), \chi^2(q_{max}, \mathbf{r}))$ 
8     if  $\chi_u^2(q', \mathbf{r}) \geq cmax$  {
9       if  $\chi^2(q_{max}, \mathbf{r}) < \chi^2(q', \mathbf{r}) \wedge q'.frequency > m$  {
10         $q_{max} = q'$ 
11         $\chi^2(q_{max}, \mathbf{r}) = \chi^2(q', \mathbf{r})$ 
12        updated = true
13      }
14       $q'.expand(q_{max}, \chi^2(q_{max}, \mathbf{r}))$ 
15    }
16    else {
17      if updated {
18        print  $q_{max}$ 
19        updated = false
20      }
21    }
22  }

```

Figure 2: Implementation of backbone refinement class mining via dynamic upper bound pruning.

*Dynamic Upper Bound Adjustment.*

Let  $u$  be the user defined minimum  $\chi^2$  value. For any frequent subtree  $q \subseteq r \in \mathbf{r}$ , let  $\chi^2(q, \mathbf{r})$  and  $\chi_u^2(q, \mathbf{r})$  denote the  $\chi^2$  value and upper bound value for  $q$ , respectively, both found by the  $\chi^2$  distribution test for  $q$ . Let  $u_{max}(q) = \max\{\chi^2(p, \mathbf{r}) | p \preceq_b q\}$ , the highest  $\chi^2$  value seen so far in this backbone refinement class. Then, if  $u_{max}(q) > u$ ,  $u$  may be increased to  $u_{max}(q)$ , since we only search for the maximum class element (*dynamic upper bound adjustment*).

Every path  $p$  satisfying the user defined minimum frequency and significance constraint serves as backbone. The search starts with the longest —i.e., non-refinable—paths and subsequently backtracks to the shorter ones. The algorithm for mining the representatives of  $p$ 's refinement classes using dynamic upper bound pruning is shown in Figure 2. The procedure is invoked as  $p.expand(p, \chi^2(p, \mathbf{r}))$ .

Lines 2-5 output the maximum 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 7,  $cmax$ , the maximum  $\chi^2$  value seen so far (including  $q'$ ) is calculated. Line 8 then implements dynamic upper bound pruning by checking  $q'$ 's upper bound value against  $cmax$ . If it is lower, the search is truncated and the maximum at that point is output (lines 16-20). Otherwise, it is updated and the iteration continues.

Figure 3 visualizes the refinement process for tree  $q_2$  from

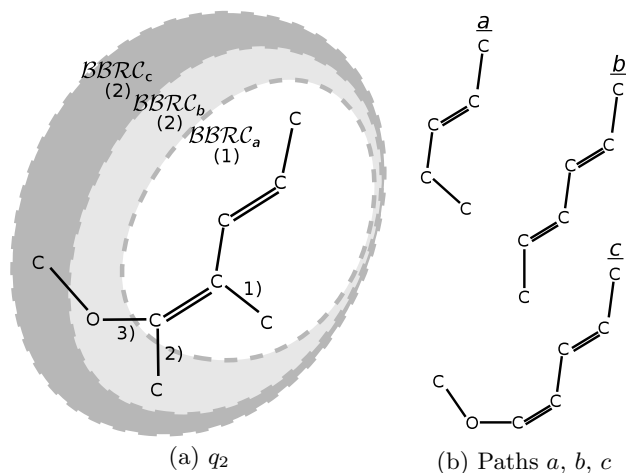


Figure 3: Backbone Refinement Classes for paths  $a$ ,  $b$ , and  $c$ .

Figure 1(b). It shows how the three different non-refinable paths in  $q_2$  are used as backbones while the structure is explored. The paths are marked bold and all carbon node labels have been made explicit. The borders of the backbone refinement classes are outlined, their sizes are given in brackets. Specifically,

- $BBRC_c$  contains two backbone refinement classes corresponding to the exclusive inclusion of either edge 1) or 2). The two classes comprise 4 subgraphs in total, namely the backbone as well as the subgraphs  $q_1$ ,  $q_2$ , and  $q_3$  (see Figure 1).
- $BBRC_b$  contains two backbone refinement classes corresponding to the exclusive inclusion of either edge 1) or 3). The two classes include 4 subgraphs in total.
- $BBRC_a$  contains a single backbone refinement class with 2 subgraphs.

The class members are enumerated by subjecting  $a$ ,  $b$ , and  $c$  to the algorithm in Figure 2. After backtracking, the same concept is applied to the other (shorter) paths in  $q_2$ .

### 3. EXPERIMENTS

In this section we will investigate BBRCs in terms of running time, feature count and expressiveness. First, we test on smaller structure-activity relationship (SAR) data. Then, we proceed with large-scale testing on the largest labelled dataset of structures tested so far. All calculations were carried out on an 8 processor 2.4 GHz Intel Xeon system with 16G of RAM running Linux 2.6.24.

#### 3.1 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: all trees that are frequent and have a minimum  $\chi^2$  significance of 95 %.

	SM No	RC No	MoC No	MuC No
1. Lin. Frag.	48,259	86,300	49,816	70,802
2. Sign. Trees	27,093	94,991	22,395	29,970
3. Open Trees	8,062	4,569	1,937	5,122
4. BBRC Repr.	2,715	5,183	3,083	3,636

**Table 2: Comparison of feature set sizes for All Linear Fragments, Significant Trees and BBRC representatives.**

- Open Trees: the most general representatives of all trees with the same occurrences from 2.
- 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.

We used four chemical datasets obtained from the Carcinogenic Potency Database (CPDB)<sup>3</sup>, 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) and Multicell Call (MuC, 553 active / 1067 compounds).

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.* [2].<sup>4</sup>

The mined subgraphs were evaluated in a leave-one-out cross-validation. 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  $\chi^2$  significance value [3]. A training compound was selected as neighbor, if its similarity to the query structure 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 (aromatic perception is discussed in section 3.2).

### 3.1.1 Effectiveness

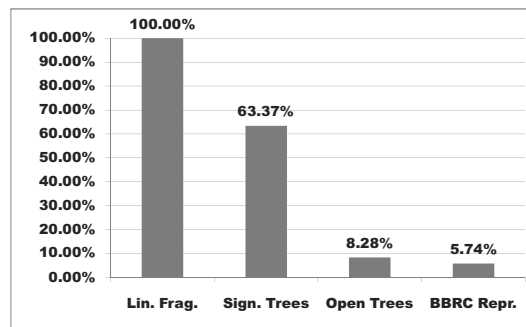
Table 2 compares the resulting fragment set sizes for the different fragment types and Table 3 indicates the mining

<sup>3</sup><http://potency.berkeley.edu/cpdb.html>

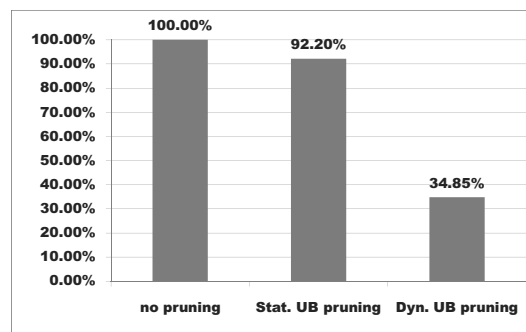
<sup>4</sup>The authors kindly provided us with a binary of their algorithm *sfgm*, pointing out that it may not be optimized for speed and uses a breadth-first search technique known to be memory demanding.

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 pruning, 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 4 summarizes the information from Tables 2 and 3 by mean values of relative feature count and running time reduction across the different datasets.



(a) Feature count mean reduction



(b) Runtime mean reduction

**Figure 4: Mean values of Tables 2 and 3, taken over the datasets SM, RC, MoC, and MuC.**

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 3.2.2 and the work of Bringmann *et al.* [2]). 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 [9, 12].

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  $\chi^2$  and upper bound calculations, and about 3% with additional control overhead due to the more sophisti-

	Applies to	SM		RC		MoC		MuC	
		s	Fraction	s	Fraction	s	Fraction	s	Fraction
No statistical pruning		2.63	100.00%	21.23	100.00%	3.71	100.00%	5.17	100.00%
Static UB pruning	Sign. & Open Trees	2.55	96.97%	21.11	99.43%	2.98	92.06%	4.76	93.85%
Dynamic UB pruning	BBRC Repr.	0.44	16.73%	6.63	31.22%	2.13	57.41%	1.76	34.04%

**Table 3: Comparison of running time for mining BBRC Representatives using different pruning techniques.**

cated *expand()* routine. Since running time gains of over 60% were obtained, it may be concluded that the additional effort is justified.

### 3.1.2 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 [3], 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 4 compares the accuracy values of BBRC representatives to all linear fragments, significant trees, and open significant trees. Table 4 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, respectively. 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 5 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 [11].

We also evaluated backbone refinement class representatives on the three datasets from the study of Rückert and Kramer [10]. To render the results comparable, we also used 6 % of the dataset size as minimum frequency threshold. After performing 10-fold cross-validation, we obtained the ac-

	all	AD	wt.	#feat. (time)
NCTRER	0.74	0.79	0.82	1,782 (0.65s)
Bloodbarr	0.72	0.75	0.84	616 (0.22s)
Yoshida	0.55	0.59	0.58	377 (0.16s)

**Table 5: Accuracy table for datasets used in the study of Rückert and Kramer [9]**

curacies shown in Table 5. 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 [10].

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.

## 3.2 Large-Scale Analysis

We employed publicly available large-scale databases to assess several advanced aspects of BBRCs beyond running time, feature count and predictivity. More precisely, we performed experiments on parts of the NCI Yeast Anticancer Drug Screen datasets<sup>5</sup> (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 (growth inhibition of at least 70 % in at least 1 strain)
- AC-All (stage 0): Total of 87,264 compounds, 5,777 active (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 (growth inhibition at least 70 % in all strains)

To the best knowledge of the authors, AC-One (stage 0) and AC-All (stage 0) are the largest labelled datasets that have been considered in correlated graph mining. The analysis investigates different minimum frequency thresholds and the impact of aromatic perception, i.e., whether the miner

<sup>5</sup><http://dtp.nci.nih.gov/yacds/download.html>

	<i>SM</i>			<i>RC</i>			<i>MoC</i>			<i>MuC</i>		
	all	AD	wt.	all	AD	wt.	all	AD	wt.	all	AD	wt.
1. Lin. Frag.	75.0	77.8	83.0	63.7	67.0	77.5	67.6	72.7	79.9	69.3	72.4	79.6
2. Sign. Trees	74.6	<b>80.7</b>	<b>86.8</b>	64.4	70.0	81.8	<b>73.3</b>	75.7	<b>83.7</b>	<b>71.9</b>	<b>75.6</b>	83.5
3. Open Trees	<b>75.5</b>	80.6	84.5	64.5	68.7	80.0	71.5	74.4	80.8	70.2	73.5	81.3
4. BBRC Repr.	74.6	79.4	85.4	<b>67.2</b>	<b>70.4</b>	<b>82.2</b>	71.7	<b>76.5</b>	82.0	70.3	74.1	<b>84.9</b>

Table 4: Descriptor accuracy for the CPDB datasets, obtained with leave-one-out crossvalidation. Bold figures indicate the best results.

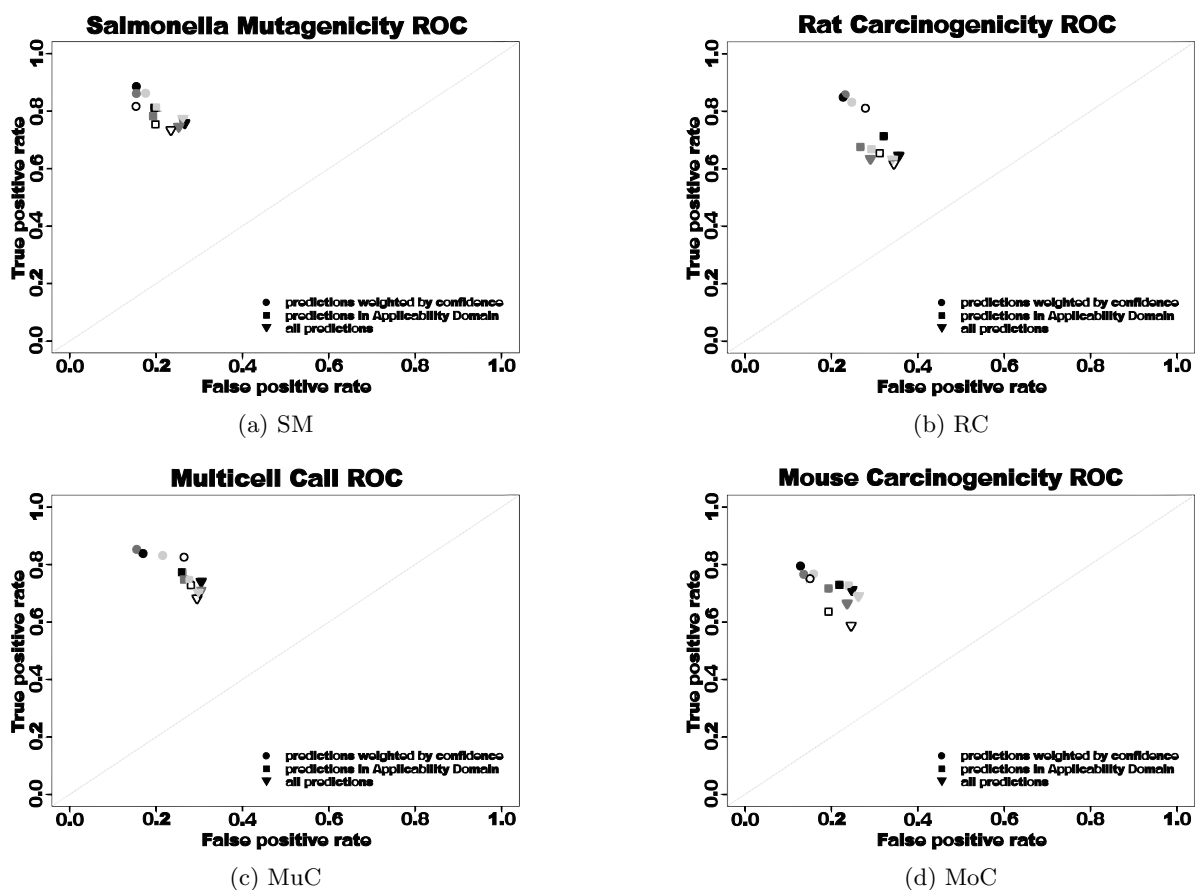


Figure 5: ROC plots for the CPDB datasets comparing Significant Trees (black), BBRC Representatives (dark gray), Open Trees (light gray) and All Linear Fragments (hollow).

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. All calculations were carried out on the same computer as in section 3.

Section 3.2.1 gives an overview of dataset coverage and BBRC class sizes, and in section 3.2.2 we report on a cross-validation run using those large-scale data.

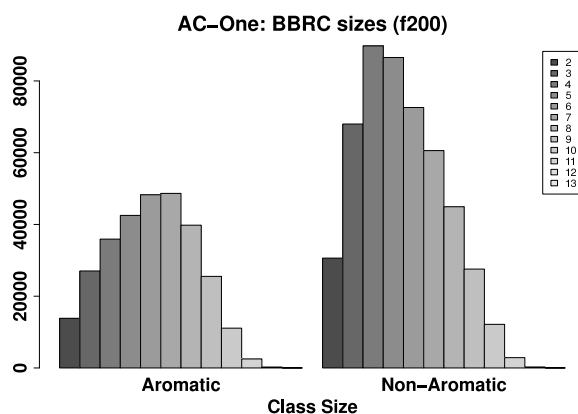
### 3.2.1 Dataset Coverage and Class Sizes

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. 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).



(a) BBRC size for minimum frequency 200

Figure 6: BBRC sizes for AC-One (stage 0)

The conclusion here is that a lower minimum frequency does not increase coverage for the 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.

The higher coverage for the non-aromatic setting raises the question whether this is an expression of low representativeness, i.e., whether backbone refinement classes are smaller for the non-aromatic setting. We assessed backbone refinement class sizes via static upper bound pruning, thus putting any significant and frequent tree into exactly one class and counting them. The frequency distribution of classes with different sizes in Figure 6 shows clearly that the non-aromatic setting produces far more classes of the same size, and moreover, the mean class size is significantly lower than for the aromatic setting (a skew to the right for the non-aromatic setting). Therefore, class size is indeed smaller, and this may be attributed to a lower expressiveness.

In summary, 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. Finally, the aromatic setting seems to produce a more sparse and representative collection of patterns.

### 3.2.2 Cross-validation

To conclude our experiments, we report on two large-scale cross-validation 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 ( $2 \cdot 11,700 = 23,400$  compounds). The second run, on AC-All (stage 1), used all actives and inactives (in total,  $5,248 + 5,300 = 10,548$  compounds).

	AC-one (stage 0)	AC-all (stage 1)
Sign. Trees	1,190,763	291,729
Open Trees	?	216,206
Max. Trees	556,673	148,562
BBRC Repr.	31,450	14,381

Table 6: Feature counts for AC-One (stage 0) and AC-All (stage 1)

We compared the number of BBRC representatives to the feature counts of other summarization methods. This time we investigated additionally the set sizes of maximum patterns (the positive border as implied by the minimum frequency and 95 % significance constraints; see also the work by Hasan *et al.* [1]).

For AC-All (stage 1), the extraction of open trees with `sfgm` took >10h, BBRC representatives took 1m13s. For AC-one (stage 0), open trees did not finish, while BBRC representatives took 4m52s.

Table 6 shows that 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 3).

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 3. 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 7 shows the validation results we obtained.

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

**Table 7: Validation results for AC-One (stage 0) and AC-All (stage 1), using BBRCs**

In summary, we consider the class of BBRC representatives a promising candidate for large-scale structural datasets, rendering them computationally feasible. Judging from our results in section 3, we can also be quite confident about the quality of predictions.

## 4. CONCLUSION

In the paper, we introduced backbone refinement classes (BBRCs), a particularly useful class of subgraphs for mining databases of chemical compounds. Due to their formal properties, BBRCs can be mined efficiently and searched by existing graph mining systems like Gaston with only minor modifications. The overall method proves to be highly efficient compared to mining significant and open trees, dramatically reducing running time and number of features mined.

Moreover, 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 mined tree structures form a sparse and structurally diverse pattern collection. Their inter-entropy seems to be beneficial for SAR systems using tree-shaped features in that a large fraction of structurally similar features are left out, which cannot be guaranteed by occurrence summarization methods, such as open or closed subgraphs.

In our experiments on the largest labelled 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 dramatic decrease in numbers, they may be also suitable for identifying so-called structural alerts, i.e., chemical substructures that are thought to be particularly toxic.

## 5. ACKNOWLEDGEMENTS

The authors thank Björn Bringmann for providing a binary and friendly cooperation in dataset testing, and Ulrich Rückert for providing datasets. The research was (partially)

supported by the EU seventh framework programme under contract no Health-F5-2008-200787 (OpenTox).

## 6. REFERENCES

- [1] M. Al Hasan, V. Chaoji, S. Salem, J. Besson, and M. Zaki. Origami: Mining Representative Orthogonal Graph Patterns. *ICDM 2007. Seventh IEEE International Conference on Data Mining*, pages 153–162, Oct. 2007.
- [2] B. Bringmann, A. Zimmermann, L. de Raedt, and S. Nijssen. Don't Be Afraid of Simpler Patterns. In *Proceedings 10th PKDD*, pages 55–66. Springer-Verlag, 2006.
- [3] C. Helma. Lazy Structure-Activity Relationships (lazar) for the Prediction of Rodent Carcinogenicity and Salmonella Mutagenicity. *Molecular Diversity*, pages 147–158, 2006.
- [4] T. Horváth, J. Ramon, and S. Wrobel. Frequent Subgraph Mining in Outerplanar Graphs. In *KDD '06: Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 197–206, 2006.
- [5] K. Jahn and S. Kramer. Optimizing gSpan for Molecular Datasets. In *Proceedings of the Third International Workshop on Mining Graphs, Trees and Sequences (MGTS-2005)*, 2005.
- [6] S. Kramer, L. De Raedt, and C. Helma. Molecular feature mining in HIV data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 136–143, New York, NY, USA, 2001. ACM.
- [7] S. Morishita and J. Sese. Traversing Itemset Lattices with Statistical Metric Pruning. In *Symposium on Principles of Database Systems*, pages 226–236, 2000.
- [8] S. Nijssen and J. N. Kok. 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*, pages 647–652, New York, NY, USA, 2004. ACM.
- [9] S. Nijssen and J. N. Kok. Frequent Subgraph Miners: Runtimes Don't Say Everything. In *Proceedings of the International Workshop on Mining and Learning with Graphs (MLG 2006)*, pages 173–180, 2006.
- [10] U. Rückert and S. Kramer. Optimizing Feature Sets for Structured Data. In *Stan Matwin and Dunja Mladenic, editors, 18th ECML*. Springer, 2007.
- [11] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.
- [12] M. Wörlein, T. Meil, I. Fischer, and M. Philippsen. A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston. In *Proceedings of PKDD*, pages 392–403, 2005.
- [13] X. Yan and J. Han. gSpan: Graph-Based Substructure Pattern Mining. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 721, Washington, DC, USA, 2002. IEEE Computer Society.
- [14] X. Yan and J. Han. CloseGraph: Mining Closed Frequent Graph Patterns. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 286–295, New York, NY, USA, 2003. ACM.