Don't Be Afraid of Simpler Patterns

Björn Bringmann, Albrecht Zimmermann, Luc De Raedt, and Siegfried Nijssen

Institute of Computer Science, Machine Learning Lab, Albert-Ludwigs-University Freiburg, Georges-Köhler-Allee 79, 79110 Freiburg, Germany
{bbringma, azimmerm, deraedt, snijssen}@informatik.uni-freiburg.de

Abstract. This paper investigates the trade-off between the expressiveness of the pattern language and the performance of the pattern miner in structured data mining. This trade-off is investigated in the context of correlated pattern mining, which is concerned with finding the k-best patterns according to a convex criterion, for the pattern languages of itemsets, multi-itemsets, sequences, trees and graphs. The criteria used in our investigation are the typical ones in data mining: computational cost and predictive accuracy and the domain is that of mining molecular graph databases. More specifically, we provide empirical answers to the following questions: how does the expressive power of the language affect the computational cost? and what is the trade-off between expressiveness of the pattern language and the predictive accuracy of the learned model? While answering the first question, we also introduce a novel stepwise approach to correlated pattern mining in which the results of mining a simpler pattern language are employed as a starting point for mining in a more complex one. This stepwise approach typically leads to significant speed-ups (up to a factor 1000) for mining graphs.

1 Introduction

Whereas initially the data mining community focused on mining simple pattern languages, such as itemsets, there has been a recent shift towards mining more and more complex pattern languages, such as sequences, trees and graphs [1,2,3]. This is often motivated by challenging applications domains such as chemoinformatics and network analysis, which can naturally be modeled as graphs. Whereas experience gained while mining simpler representations, such as the exploitation of the *apriori*-property [4], has been transferred into structured data mining, little research has been devoted to the trade-off between expressiveness of the pattern language and the performance of the pattern miner. Within the field of computer science, insight into the trade-off between expressiveness and performance has been of critical importance, and therefore also seems essential to the field of data mining and its applications.

The setting that we shall employ is that of correlated pattern mining, where one is given a dataset divided into two classes, and the aim is to find the k-best patterns expressed within a pattern language $\mathcal{L}$ according to some statistical criteria, such as significance, or information gain [5,6]. Correlated pattern miners typically employ a branch-and-bound technique. Because graph miners are today

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 55–66, 2006.

the most prominent representatives of structured data mining systems, and their application has been to a large extent targeted towards molecular applications in chemo-informatics such as structure activity relationship prediction [7], our experimental investigation targets such applications. The pattern languages $\mathcal{L}$ considered are those of *itemsets*, *multi-itemsets*, *sequences*, *trees*, and *connected graphs*. Even though there are many possible performance criteria in data mining, computational cost and predictive performance are certainly the most prominent ones, which we will therefore also adopt in our study. W.r.t. computational cost, the question is how the computational cost is affected by growing expressiveness of the language. While answering this question, we also introduce the technique of stepwise correlated pattern mining, which first finds the k best patterns in the simpler language, and then employs the score of the k-th best pattern as a bound while mining the more expressive language in a branch-and-bound algorithm. In some of our experiments, the stepwise approach leads to speed-ups of a factor 1000 for large molecular datasets. To gain insight into the second trade-off, which is concerned with predictive performance, we employ the typical approach of using the k best patterns as features in a classifier. This then allows us to construct a predictive model in the form of a decision tree, rule-set or naive Bayes model, and hence, to evaluate the predictive performance of the resulting model and thus, pattern language.

The paper is structured as follows: In the next section we introduce the notations used throughout the rest of the paper. In section 3, we describe correlation measures and the branch-and-bound approach used to solve the mining task. The experiments and their results are described and discussed in detail in section 4. In the last section we conclude and point towards related and future work.

2 Pattern Languages

In this section, we introduce the various pattern languages employed in our study, and discuss the relationship among them.

Definition 1 (Graphs). *An* undirected, labeled graph $\mathcal{G}(\mathbb{V}, \mathbb{E}, \lambda, \Sigma)$ *consists of a finite set* $\mathbb{V}$ *of vertices, a set* $\mathbb{E} \subseteq \{\{u, v\} | u, v \in \mathbb{V}, u \neq v\}$ *of edges, an alphabet* Σ *and a labeling function* $\lambda : (\mathbb{V} \cup \mathbb{E}) \to \Sigma$.

A graph $\mathcal{G}(\mathbb{V}, \mathbb{E}, \lambda, \Sigma)$ is called *connected* iff $\forall u, v \in \mathbb{V}$ there exists a sequence of vertices $\exists v_1, ..., v_n \in \mathbb{V}$ with $v_1 = u$ and $v_n = v$ and $\{v_x, v_{x+1}\} \in \mathbb{E}$.

Definition 2 (Trees). *A (free)* tree *is a connected graph with* $|\mathbb{V}| = |\mathbb{E}| + 1$.

Connected graphs that are not trees thus have cycles and are called *cyclic graphs*.

Definition 3 (Sequences). *A* sequence *is a tree (and hence a graph) where no vertex has more than two edges (i.e. no branches),* $\forall v \in \mathbb{V} : \left|\{\{v, u\} | u \in \mathbb{V}\}\right| \leq 2$.

Definition 4 (Multi-itemset). *A* multi-itemset *is a graph* $\mathcal{G}(\mathbb{V}, \mathbb{E}, \lambda, \Sigma)$ *where the set of edges is empty,* $\mathbb{E} = \emptyset$.

Definition 5 (Itemset). *A multi-itemset is called* itemset *iff* $\forall v_a, v_b \in \mathbb{V} : (v_a = v_b) \vee (\lambda(v_a) \neq \lambda(v_b))$.

Perhaps the definitions of multi-itemset and itemset are a bit unusual, but they are convenient for showing the relationship among the different pattern types. It is useful to use the following notation for the pattern languages: $\mathcal{L}_G$, the set of all graphs; $\mathcal{L}_C$, the set of all connected graphs; $\mathcal{L}_T$, the set of all trees; $\mathcal{L}_S$, the set of all sequences; $\mathcal{L}_M$, the set of all multi-itemsets; and $\mathcal{L}_I$, the set of all itemsets. From the definitions introduced above, it directly follows that

Proposition 1. $\mathcal{L}_I \subset \mathcal{L}_M \subset \mathcal{L}_G$ *and* $\mathcal{L}_S \subset \mathcal{L}_T \subset \mathcal{L}_C \subset \mathcal{L}_G$

At the same time, the subgraph isomorphism relation can be used as the *covers* and *generality* relation to structure the search:

Definition 6 (Graph Isomorphism). *Two Graphs* $\mathcal{G}(\mathbb{V},\mathbb{E},\lambda,\Sigma)$,$\mathcal{G}'(\mathbb{V}',\mathbb{E}',\lambda',\Sigma)$ *are called* isomorphic *if there exists a bijective function* $\varphi : \mathbb{V} \rightarrow \mathbb{V}'$ *such that:* $\forall v \in \mathbb{V} : \lambda(v) = \lambda'(\varphi(v)) \wedge\ \mathbb{E}' = \{\{\varphi(v_1), \varphi(v_2)\} | \{v_1, v_2\} \in \mathbb{E}\} \wedge\ \forall \{v_1, v_2\} \in \mathbb{E} : \lambda(\{v_1, v_2\}) = \lambda'(\{\varphi(v_1), \varphi(v_2)\})$

Definition 7 (Subgraph). *Given two graphs* $\mathcal{G}(\mathbb{V}, \mathbb{E}, \lambda, \Sigma)$, $\mathcal{G}'(\mathbb{V}', \mathbb{E}', \lambda', \Sigma)$, $\mathcal{G}'$ *is called a* subgraph *of* $\mathcal{G}$ *iff* $\mathbb{V}' \subseteq \mathbb{V} \wedge \mathbb{E}' \subseteq \mathbb{E} \wedge \forall v \in \mathbb{V}' : \lambda(v') = \lambda(v) \wedge \forall e \in \mathbb{E}' \lambda(e') = \lambda(e)$

A graph $\mathcal{G}'$ is subgraph-isomorphic to another graph $\mathcal{G}$ if and only if it has a subgraph $\mathcal{S}$ that is isomorphic to $\mathcal{G}'$. The subgraph isomorphism relation can be applied to the more specific pattern languages and results in a natural notion of coverage or generality for these languages as well.

3 Correlated Pattern Mining

A correlation measure compares the expected frequency of the joint occurence of a pattern and a certain class value to the observed frequency. If the resulting value is larger than a given threshold the deviation is considered statistically significant and there is evidence for a causal relationship between the pattern and the class.

	c_1	c_2	
ϕ	p	n	$p+n$
$\neg\phi$	$P-p$	$N-n$	$\|D\|-(p+n)$
	P	N	$\|D\|$

Fig. 1. A contingency table

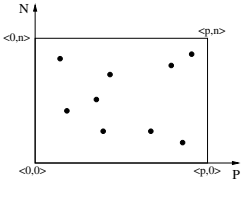

Fig. 2. Convex hull of future $\langle p', n' \rangle$

We organize the observed frequencies in a contingency table, cf. Figure 1. Let ϕ be the pattern and c_1, c_2 the two classes. P and N denote the total number

of instances having class c_1, c_2, respectively, in dataset D. The frequencies of those two classes amongst the instances covered by the pattern are referred to as p and n. Since p and n are sufficient for calculating the value of a correlation measure on this table, we view these measures as real-valued functions on $\mathbb{N}^2$ for the remainder of this paper.

Correlation measures are neither monotone nor anti-monotone, hence the search becomes more difficult than in frequent pattern mining. However, if the correlation measure is convex, an upper bound on the score for the refinements of a given pattern can be calculated, which enables the use of an effective branch-and-bound algorithm.

Convex functions like χ^2 and *Information Gain* (see [5] for a proof) take their extreme values at the points forming the convex hull of their domain.

For a pattern inducing the tuple $\langle p, n\rangle$, specialization will decrease p, n, both, or none. This leads to points lying within the rectangle shown in Figure 2. Evaluating the correlation measure at $\langle p, 0\rangle$ and $\langle 0, n\rangle$ will give an upper bound on the value the measure can take for specializations of the current pattern. For an in-depth discussion of upper bound calculation we refer the reader to [5,6].

Correlated pattern miners now compute either the k best patterns within a language of patterns $\mathcal{L}$ or else finds all patterns within $\mathcal{L}$ whose correlation measure exceeds a certain threshold. Furthermore, to avoid redundancy in the resulting solution set, one typically employs patterns that are *free*, i.e., it is not allowed that two patterns ϕ_1 and ϕ_2 occur in the solution set for which ϕ_2 is a refinement of ϕ_1 and both patterns reach the same score.

The solutions to the first task can be conveniently modeled using

$$Th_k(\mathcal{L}, D) = \{\phi \in \mathcal{L} | free(\phi, D), \phi \textit{ among the k-best patterns w.r.t. } score(\phi, D) \textit{ in } \mathcal{L}\}$$

where D denotes the dataset under consideration, and *score* the correlation measure considered. Solutions to the second type of query are represented as

$$Th_>(\mathcal{L}, D, t) = \{\phi \in \mathcal{L} | free(\phi, D) \wedge score(\phi, D) > t\}$$

It will be convenient to combine the notations and introduce:

$$Th_k(\mathcal{L}, D, t) = \{\phi \in \mathcal{L} | free(\phi, D) \text{ and } score(\phi, D) > t \text{ and } \phi \textit{ is amongst the k-best patterns w.r.t. } score(\phi, D) \text{ in } \mathcal{L}\}$$

In the light of the hierarchies of languages introduced above, it is now instructive to look at some straightforward properties of these solution sets.

Proposition 2. *Whenever $\mathcal{L}_1 \subseteq \mathcal{L}_2$, then for all datasets D and thresholds t: $Th_>(\mathcal{L}_1, D, t) \subseteq Th_>(\mathcal{L}_2, D, t)$.*

This property actually states that whenever $\mathcal{L}_1 \subseteq \mathcal{L}_2$, it will be the case that solutions found within $\mathcal{L}_1$ will also be solutions within the more expressive pattern language $\mathcal{L}_2$. Another useful property is

Proposition 3. *Whenever $\mathcal{L}_1 \subseteq \mathcal{L}_2$, then for all datasets D: $Th_k(\mathcal{L}_2, D) \cap \mathcal{L}_1 \subseteq Th_k(\mathcal{L}_1, D)$.*

It states that the solutions of a top k query in a larger language $\mathcal{L}_2$ may contain some top k solutions from a less expressive language $\mathcal{L}_1$.

These properties actually motivate the stepwise correlated pattern mining algorithm, which we will now sketch. Algorithm 1 assumes a given hierarchy of pattern languages $\mathcal{L}_1 \subset \mathcal{L}_2 \subset \mathcal{L}_3 \ldots \mathcal{L}_n$, a dataset D, and a value for k. The goal is to find $Th_k(\mathcal{L}_n, D)$.

Algorithm 1. Stepwise Correlated Pattern Mining.

$t_0 := -\infty$;
for i = 1 to n **do**
  compute $Th_k(\mathcal{L}_i, D, t_{i-1})$
  $t_i := \min_{p \in Th_k(\mathcal{L}_i, D, t_{i-1})} score(p, D)$
return $Th_k(\mathcal{L}_i, D, t_i)$

The idea underlying the stepwise algorithm is that one first searches for the k best patterns in the simpler language $\mathcal{L}_i$, and then records the score of the k-th best solution found in $\mathcal{L}_i$. This score must be lower than or equal to the score of all solutions in $Th_k(\mathcal{L}_{i+1}, D)$, and can therefore be used as a threshold when searching for the solutions at the next level. Given that correlated pattern miners employ a branch-and-bound algorithm, this is likely to result in additional pruning when searching $\mathcal{L}_{i+1}$. This process is then iterated until the final language $\mathcal{L}_n$ is considered.

It is easy to see that the stepwise correlated pattern mining algorithm will produce exactly the same results as directly computing the $Th_k(\mathcal{L}_n, D)$. The question however is whether this stepwise technique is more efficient than the direct approach. In the experimental section, we shall provide evidence that this typically is the case for the hierarchies of pattern languages considered, and that the speed-up can be significant (up to a factor of about 1000).

4 Experimental Evaluation

In this section, we experimentally answer the following questions:

Q1 Does the stepwise correlated pattern miner speed up the correlated pattern mining process for the mining of graphs?
Q2 How does the expressiveness of the pattern language influence the running times of the correlated pattern miner?
Q3 How does the expressiveness of the pattern language influence the predictive performance of the models learned using correlated patterns as features ?

4.1 Experimental Set-Up

In all experiments, we employed a correlated pattern miner to compute$Th_k(\mathcal{L}, D)$ for the values $k = 1, 10, 100$ and 1000, and the languages $\mathcal{L} = \mathcal{L}_I, \mathcal{L}_M, \mathcal{L}_S, \mathcal{L}_T$ and $\mathcal{L}_C$. For the mining step we implemented two correlated pattern miners. For

the itemsets and multi-itemsets a simple APRIORI-SMP [5]-like implementation was used. For sequences, trees and graphs a modified GSPAN [2] implementation, able to mine correlated patterns, was employed. The correlation measure used was χ^2 and the starting minimum threshold for the first step in the stepwise mining procedure and all direct runs was 3.84.

To guarantee that we find all k-best free patterns, it is essential that for every large graph, all its subgraphs are enumerated first. Otherwise, it might happen that the addition of a small subgraph to a set of k-best patterns necessitates the removal of several k-best supergraphs, and, consequently, one could not guarantee that we find all k-best free patterns. It can be shown however that the enumeration schemes that we are using (both the stepwise approach, and GSPAN's method for enumerating graphs) have this desirable property. Furthermore, the properties of the DFS-Encoding used for graphs in GSPAN allowed in a straightforward manner to restrict the mining process to only trees (sequences) by not allowing structures with cycles (branches).

All running time experiments were performed on a 2.8GHz Machine with 2GB of main memory running Linux.

The two evaluation criteria employed were, on the one hand, the computation time needed to compute $Th_k(\mathcal{L}, D)$ and, on the other hand, the predictive accuracy. As directly evaluating the predictive accuracy of a correlated pattern is typically not very interesting, we rather evaluate the predictive accuracy of a set of patterns $Th_k(\mathcal{L}, D)$ indirectly. This is realized by employing these patterns as features in binary vectors describing the datasets to the WEKA toolkit. We then employed WEKA's [8] implementations of RIPPER [9], C4.5 [10], a SUPPORT VECTOR MACHINE, and NAÏVE BAYES to build classifiers whose accuracies can be measured. All accuracy estimates were obtained using ten-fold cross-validation, and the results were compared against each other w.r.t. significance. The WEKA experimenter environment was used for this task since accuracy averaging and significance testing is automated in this tool.

4.2 Datasets

For the experimental evaluation, we used four different real-world datasets, namely the *NCI HIV* dataset [3], a biodegradability dataset [11], and two mutagenicity datasets [12,7]. All these datasets contain a number of graphs describing chemical compounds. Atoms are represented as vertices and labeled with the atom type, whereas bonds between atoms are represented as edges, also labeled with the type of edge (which can be single bond, double bond or aromatic bond).

To answer the first two questions, we performed running time experiments on the **HIV** and on the **Mutagenicity I** datasets. Due to the small size of the other two datasets, running times measured are in the range of a few seconds at most, making differences unreliable. We used 5 different setups of the HIV dataset, namely *active vs. inactive* (CA vs CI), *active vs. moderate* (CA vs CM), *moderate vs. inactive* (CM vs CI), *active vs. moderate and inactive* (CA vs CMCI), and *active and moderate vs. inactive* (CACM vs CI). For each of the 6 setups, we mined the k-best patterns for $k = 1, 10, 100$, and 1000.

Table 1. Characteristics of the used datasets

Name	Total Size	Number of Classes	Class sizes
HIV	41768	3 (CA,CM,CI)	CA: 417, CM: 1069, CI: 40282
Biodegradability	328	2 (BD, NBD)	BD: 185, NBD: 143
Mutagenicity I	4337	2 (M,NM)	M: 2401, NM: 1936
Mutagenicity II	684	2 (M,NM)	M: 341, NM: 343

4.3 Stepwise Correlated Pattern Mining Q1

Our first question was whether the stepwise approach would speed up the mining process compared to the direct approach. The experiments performed show that the stepwise approach was up to 2800 times faster with an average of 202.81 ($\sigma = 600.66$) on all settings. Some of the results are shown in Table 2. The direct approach for $k = 1000$ in the *CS vs. CM* on the **HIV** dataset took more than 48 hours[1]! Figure 3 shows an explanation for the acceleration of the mining process. After evaluating only 600 sequences, the threshold reaches a value of 769. When mining trees (resp. graphs) directly, a similar threshold is reached after analyzing 2700 (resp. 4200) patterns. The quickly rising threshold enforced by mining stepwise allows a much more efficient pruning than the direct approach.

However, we observed some contrary cases that require further explanation. First, when comparing the direct and the stepwise approach on multi-itemset mining ($\mathcal{L}_M$), the stepwise approach does not help in terms of speed-up. Fortunately, it does not hurt either. Furthermore, on the **Mutagenicity I** dataset the stepwise approach was always slightly slower than the direct approach with the worst case for $k = 10$. Finally the direct mining of the 1000-*best* patterns on the *CA vs. CI* **HIV** dataset was 13.41% faster than the stepwise mining. In this case, the sequence-mining step could raise the threshold to 18.62 only which does not help much in the next mining step, especially considering the score of the 1000th-best pattern in $\mathcal{L}_C$ with 796.82. Furthermore, each step has to rebuild the *k-best* list since only the threshold is reused. This can be time consuming and might - as in this case - have the effect that the stepwise approach is slower than the direct approach. This could be improved by reusing not only the threshold obtained by a mining step, but the whole *k-best* list.

4.4 Running Time Q2

This second question considers the influence of the expressiveness of the pattern language on the runtime. More precisely: How are the running times distributed among the steps for mining $\mathcal{L}_S$ to $\mathcal{L}_T$ to $\mathcal{L}_C$? In principle, mining $\mathcal{L}_S$ could increase the threshold t_S to such a high level that mining $\mathcal{L}_T$ using t_S is faster than the mining of $\mathcal{L}_S$. The experiments in Table 3 show that this is never the case. The time spent in the tree mining step is on average 22 times as high as the sequence mining step. The graph mining step, on the other hand, is never

[1] Before it ran out of memory.

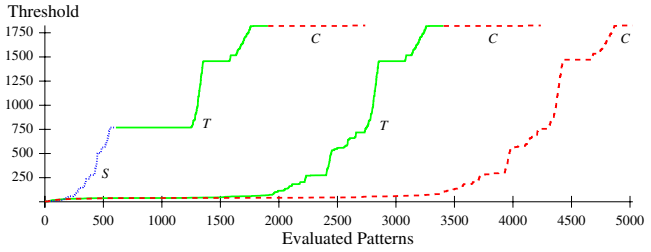

Fig. 3. Threshold vs. evaluated patterns for the stepwise and the direct approach on *HIV CA vs. CM* with $k = 100$. Mining $\mathcal{L}_S$ (dotted) followed by $\mathcal{L}_T$ (solid) and then $\mathcal{L}_C$ (dashed) evaluates much less patterns than mining $\mathcal{L}_T$ followed by $\mathcal{L}_C$. Mining $\mathcal{L}_C$ directly has to evaluate the most patterns.

much more expensive than the tree mining step. We also performed experiments on itemsets and multi-itemsets, which showed that itemsets were much faster than multi-itemsets. The results are not listed as other experiments showed that these features are not competitive.

Table 2. Running times of the stepwise and the direct approach for mining connected graphs ($\mathcal{L}_C$)

Approach	k	Muta I	CA vs CM	CACM vs CI	CA vs CI	CM vs CI	CA vs CMCI
direct	1000	0:05.16	>48h	7:48.21	0:19.54	4:07.15	0:23.51
stepwise	1000	0:06.39	0:03.57	5:11.19	0:22.59	3:02.15	0:23.15
direct	10	0:00.12	1:47.25	7:04.48	0:07.42	3:02.26	0:09.31
stepwise	10	0:00.22	0:00.14	0:05.63	0:01.03	0:07.18	0:01.10

Table 3. Running times of the stepwise approach for mining $\mathcal{L}_S, \mathcal{L}_T$, and $\mathcal{L}_C$, respectively. ($k = 1, 100$ are not listed).

Approach	k	Muta I	CA vs CM	CACM vs CI	CA vs CI	CM vs CI	CA vs CMCI
$\mathcal{L}_S$	1000	00:00.11	00:00.07	00:01.42	00:01.15	00:01.37	00:01.18
$\mathcal{L}_T$	1000	00:04.37	00:03.47	05:08.16	00:21.57	02:57.67	00:22.15
$\mathcal{L}_C$	1000	00:06.39	00:03.57	05:11:19	00:22.59	03:02.15	00:23.15
$\mathcal{L}_S$	10	00:00.03	00:00.04	00:00.56	00:00.36	00:00.57	00:00.38
$\mathcal{L}_T$	10	00:00.12	00:00.09	00:03.06	00:01.11	00:04.15	00:01.16
$\mathcal{L}_C$	10	00:00.22	00:00.14	00:06.13	00:01.03	00:07.18	00:01.10

4.5 Predictive Performance Q3

A typical usage of patterns found via a data mining approach is to employ them as features for describing instances to a propositional machine learning algorithm. Since mining more complex structures requires more time, as can be seen above, naturally the question arises whether this increase in mining effort

Table 4. Accuracy results on the HIV and Mutagenicity I datasets

(a) HIV *CA* vs *CM*

$\mathcal{L}_I$	$\mathcal{L}_M$	$\mathcal{L}_S$	$\mathcal{L}_T$	$\mathcal{L}_C$	k	Algorithm
72.88	74.16	76.78	**77.06**	**77.06**		J48
72.88	74.16	76.78	**77.06**	**77.06**	1	JRip
72.88	74.16	76.78	**77.06**	**77.06**		SVM
72.88	74.16	76.78	**77.06**	**77.06**		NB
72.88	74.16	**76.45**	76.28	76.28		J48
72.71	74.13	76.35	**77.06**	**77.06**	10	JRip
72.82	74.16	76.45	**76.85**	**76.85**		SVM
72.08	*73.02*	76.52	**76.65**	**76.65**		NB
73.05	74.16	**77.46**	77.06	77.06		J48
71.91	73.49	76.38	**77.06**	**77.06**	100	JRip
72.72	74.56	**77.09**	76.72	76.72		SVM
69.38	*72.21*	76.31	**76.58**	**76.58**		NB
73.05	*75.54*	**83.21**	*75.95*	*75.95*		J48
71.91	*74.84*	**81.29**	*76.52*	*76.42*	1000	JRip
72.72	*74.50*	**81.97**	*75.84*	*75.81*		SVM
69.38	*72.11*	**77.83**	76.65	76.65		NB

(b) Mutagenicity I

k	$\mathcal{L}_I$	$\mathcal{L}_M$	$\mathcal{L}_S$	$\mathcal{L}_T$	$\mathcal{L}_C$
	60.89	**61.15**	*58.20*	*58.40*	*58.40*
1	60.89	**61.15**	*58.20*	*58.40*	*58.40*
	60.89	**61.15**	*58.20*	*58.40*	*58.40*
	60.89	**61.15**	*58.20*	*58.40*	*58.40*
	61.40	*61.07*	**70.96**	*68.55*	*68.55*
10	*61.15*	*60.92*	**70.96**	*68.50*	*68.53*
	61.43	*60.45*	**70.19**	*68.52*	*68.49*
	61.17	*61.06*	**67.44**	*59.12*	*59.12*
	62.00	*65.67*	**76.37**	*70.94*	*70.79*
100	*61.33*	*65.11*	**74.06**	*70.74*	*70.89*
	62.17	*59.59*	**72.39**	70.23	70.17
	61.21	*60.24*	**70.08**	*65.67*	*65.90*
	62.00	*61.30*	**79.73**	*74.83*	*74.68*
1000	*61.33*	*60.89*	**76.40**	*72.23*	*71.66*
	62.17	*67.24*	**80.14**	*71.24*	*70.61*
	61.21	*60.80*	**71.70**	*57.61*	*57.33*

is matched by an increase of the quality of the description derivable by using the found patterns (**Q3**).

The first evaluated setting involved the HIV *active* and *moderately active* datasets. Accuracy estimates are show in Table 4(a). In this table, and all following ones, the best value an algorithm achieves for a given k is shown in **bold** numbers. All values that are significantly worse at $\sigma = 5\%$ are *italicized*. Itemsets never manage to capture enough information to be useful as features in this case and multi-itemsets also perform worse than the structured representations even though the difference is not significant except in the case $k = 1000$. As for the structured patterns - sequences, trees, and connected graphs - there is no significant difference for $k = 1, 10, 100$. For $k = 1000$, sequences significantly outperform trees and graphs except when used with the NAÏVE BAYES classifier which is overwhelmed by the number of features.

The second dataset was first used in a machine learning setting in [12]. Accuracy results on this dataset are summarized in Table 4(b) Except when only one feature is used, sequences are the most successful pattern type w.r.t. accuracy of the learners using those as features. The improvement in comparison to the other feature classes is significant except for the SVM when $k = 100$.

On the *biodegradability* dataset, there is no significant difference in accuracy for $k = 1$ and $k = 10$ as shown in Table 5(a). All representations more complex than pure itemsets perform very similarly to each other for the other two settings, with the notable differences that the SVM makes far better use of the sequence-type feature than of tree and graph features for $k = 100$ and that trees perform better than graphs in J48 and SVM for $k = 1000$. Finally, on the second mutagenicity dataset, for which results are reported in Table 5(b),

Table 5. Accuracy results on the Biodegradability and Mutagenicity II datasets

(a) Biodegradability

$\mathcal{L}_I$	$\mathcal{L}_M$	$\mathcal{L}_S$	$\mathcal{L}_T$	$\mathcal{L}_C$	k	Algorithm
67.05	**67.38**	65.25	65.25	65.25		J48
67.05	**67.38**	65.25	65.25	65.25	1	JRip
67.05	**67.38**	65.25	65.25	65.25		SVM
67.05	**67.38**	65.25	65.25	65.25		NB
65.51	68.43	**72.42**	72.27	72.27		J48
64.45	68.44	**74.56**	74.42	74.42	10	JRip
67.05	67.34	71.96	**72.43**	**72.43**		SVM
67.05	68.13	**74.09**	73.95	73.95		NB
65.51	72.58	**76.22**	71.96	71.81		J48
64.45	76.41	**77.89**	72.73	72.73	100	JRip
67.05	75.32	**80.04**	*71.96*	*71.81*		SVM
67.05	**69.04**	68.93	66.80	66.80		NB
65.51	73.45	74.10	**76.07**	71.20		J48
64.45	77.45	76.36	**78.95**	*72.42*	1000	JRip
67.05	74.89	76.08	**78.66**	*73.02*		SVM
67.05	**68.79**	64.65	63.13	62.98		NB

(b) Mutagenicity II

k	$\mathcal{L}_I$	$\mathcal{L}_M$	$\mathcal{L}_S$	$\mathcal{L}_T$	$\mathcal{L}_C$
	64.47	**64.91**	64.32	64.32	64.32
1	64.47	**64.91**	64.32	64.32	64.32
	64.47	**64.91**	64.32	64.32	64.32
	64.47	**64.91**	64.32	64.32	64.32
	65.86	*64.91*	**70.46**	**70.46**	**70.46**
10	*66.08*	*62.27*	**70.39**	**70.39**	**70.39**
	65.93	*62.79*	**70.39**	**70.39**	**70.39**
	64.62	64.47	**67.82**	**67.82**	**67.82**
	65.86	*64.39*	**70.90**	70.39	70.53
100	*66.08*	*64.75*	**71.48**	71.12	71.12
	65.93	*65.50*	**72.72**	71.71	71.78
	64.25	*66.08*	**72.07**	71.05	71.05
	65.86	*65.12*	71.63	71.55	**72.06**
1000	*66.08*	*67.87*	74.41	72.95	**74.48**
	65.93	*69.14*	**75.95**	74.99	75.13
	64.25	*64.77*	71.26	72.50	**72.87**

itemsets and multi-itemsets are significantly outperformed by the more complex representations. These in turn show no clear best or worst language class among themselves. The results of all experiments are surprising in that the use of more expressive pattern languages than $\mathcal{L}_S$ does not seem to pay off in terms of predictive accuracy. In all settings, sequences were at least as informative as trees and graphs when representing molecules. Yet sequences are much easier to handle and to compute than trees and graphs. On the other hand, the information stored in itemsets and multi-itemsets is typically not precise enough to be useful for a propositional learner. To gain more insight into the underlying reasons for these findings, we set up a further experiment in which we selected the $k = 1000$ best patterns in $\mathcal{L}_C \cup \mathcal{L}_M$ and classified them as (cyclic) graphs, trees, sequences, multi-itemsets and itemsets.

The two charts in Figure 4 show the distribution of the patterns, as well as the number of such patterns, their average score, and the standard deviation. Most of the experiments resulted in a chart similar to the one shown in Figure 4 (right) where the vast majority are trees, and the best scoring pattern is also a tree. In two exceptions like in Figure 4 (left), the multisets scored surprisingly well whereas in most of the other cases no single multiset or itemset ever appeared among the 1000 best. In two cases, the highest scoring structure was a sequence, and even though the sequences were much less frequent, they scored comparative to the trees. Furthermore, graphs (with cycles) usually had very low scores. A feasible explanation for these results might be in the fact that a graph contains far more sub-trees than sub-sequences and cyclic sub-graphs, and hence, that correlated pattern miners have to process much more trees (with similar scores) than sequences or cyclic graphs.

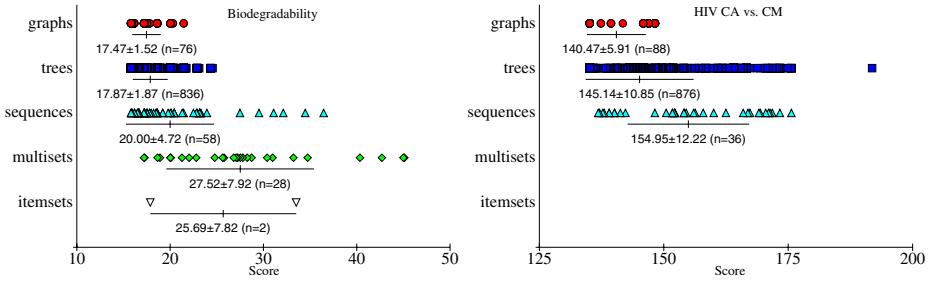

Fig. 4. Distribution of the $k = 1000$ best patterns for Biodegradability (left) and HIV CA vs. CM (right)

5 Conclusions

We have presented an empirical evaluation of the influence of the expressiveness of the pattern language on the performance of correlated graph miners. Perhaps the most surprising result of this study was that the use of more complex patterns such as cyclic graphs and trees does not necessarily lead to a better accuracy. Indeed, the best results obtained were by using sequential patterns, which are far easier to compute. A further result of our investigation was the introduction of a novel stepwise approach to correlated pattern mining, in which one first searches for k correlated patterns in a simpler language and then employs the score of the k-th best pattern as a lower bound for finding patterns at the next level of expressiveness. This stepwise approach led in virtually all cases to a significant speed-up, sometimes with a factor of up to 1000.

This work is related to the QUICKSTART approach by Nijssen and Kok [13] where a monotone constraint was considered. In this paper we study a branch-and-bound search using a convex constraint. This leads to further important differences. In our opinion, a constrained pattern mining algorithm consists of several elements: an algorithm that determines which candidates should be evaluated as it is not known yet if they satisfy the constraint, an algorithm that removes duplicate candidates and finally an algorithm that performs this evaluation. In the QUICKSTART approach, the last two steps were optimized. In this paper, we intend to optimize the first part. By considering a set of simple patterns first, we hope to find a threshold that allows for more pruning. Even though there were also speed-ups in the QUICKSTART approach, the magnitude was certainly not comparable to the factors obtained in correlated pattern mining. Finally, it would even be possible to combine both approaches. Concerning the influence of the predictive performance, related questions have arisen for instance in the kernel community, cf. [14], where different graph kernels take into account different types of information. Also, in the chemo-informatics community, cf. [15], it is often argued that one should take into account 3D information about the compounds in addition to the 2D graph structure. Doing this within our framework would be an interesting question for further research. It would

also be interesting to repeat our investigation in other domains than computational chemistry.

Acknowledgments. We are grateful to Andreas Karwath for providing the datasets. Furthermore, we like to thank the anonymous reviewers which helped us to improve the paper. The work was partially supported by the IQ project (EU grant IST-FET FP6-516169).

References

1. Zaki, M.: Efficiently mining frequent trees in a forest. In Hand, D., Keim, D., Ng, R., eds.: KDD. (2002) 71–80
2. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: ICDM. (2002) 721–724
3. Kramer, S., De Raedt, L., Helma, C.: Molecular feature mining in HIV data. In: KDD. (2001) 136–143
4. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB. (1994) 487–499
5. Morishita, S., Sese, J.: Traversing itemset lattices with statistical metric pruning. In: PODS. (2000) 226–236
6. Zimmermann, A., De Raedt, L.: Corclass: Correlated association rule mining for classification. In Suzuki, E., Arikawa, S., eds.: DS. (2004) 60–72
7. Helma, C., Cramer, T., Kramer, S., De Raedt, L.: Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. Journal of Chemical Information and Computer Systems **44** (2004) 1402–1411
8. Frank, E., Hall, M., Trigg, L.E., Holmes, G., Witten, I.H.: Data mining in bioinformatics using Weka. Bioinformatics **20** (2004) 2479–2481
9. Cohen, W.W.: Fast effective rule induction. In Prieditis, A., Russell, S.J., eds.: ICML. (1995) 115–123
10. Quinlan, J.R.: C4.5: Programs for Machine Learning. (1993)
11. Blockeel, H., Dzeroski, S., Kompare, B., Kramer, S., Pfahringer, B., Laer, W.V.: Experiments in predicting biodegradability. Appl. Art. Int. **18** (2004) 157–181
12. Kazius, J., Nijssen, S., Kok, J., Back, T., IJzerman, A.: Substructure mining using elaborate chemical representation. Journal of Chemical Information and Modeling **46** (2006) 597–605
13. Nijssen, S., Kok, J.N.: A quickstart in frequent structure mining can make a difference. In: KDD. (2004) 647–652
14. Horváth, T., Gärtner, T., Wrobel, S.: Cyclic pattern kernels for predictive graph mining. In: KDD. (2004) 158–167
15. Wale, N., Karypis, G.: Acyclic subgraph-based descriptor spaces for chemical compound retrieval and classification. Technical report, Univ. Minnesota (2006)