Discriminative Closed Fragment Mining and Perfect Extensions in MoFa

Thorsten Meinl*, Christian Borgelt† and Michael R. Berthold‡

Abstract. In the past few years many algorithms for discovering frequent subgraphs in graph databases have been proposed. However, most of these methods are limited to finding only relatively small fragments or restrict the discovered structures in other ways, which makes them not very useful for applications in biochemistry. Recently the authors of the original gSpan algorithm have shown how the usage of closed fragments can considerably speed up their algorithm. However, the main limitation to small fragments still remains. In this paper we show how the more versatile search algorithm underlying MoFa can benefit greatly from using closed fragments as well and how the concept of perfect extensions quite naturally allows to prune the underlying search tree. We demonstrate how this results in speed-ups on the NCI's HIV database.
Keywords: closed fragments, substructure mining, graphs, molecules, pruning

1 Introduction

1.1 Motivation

Finding common features in large sets of molecules is a frequently reoccurring problem in many biological or chemical applications. Examples include drug discovery, where the goal is to identify common properties that molecules share which were identified as "active" in a so-called High-Throughput Screen. Such screens typically produce activity information for hundreds of thousands of molecules. Other examples are compound synthesis, i.e. the generation of new molecules based on so-called virtual libraries. The ability to predict chances of a successful synthesis before it is being attempted can save valuable resources. Again, results for hundreds of thousands of attempted syntheses exist from which knowledge can be derived. In all these cases there exist many possible modes of action, that is, reasons why a specific molecule interacts with the sample or why a synthesis fails or succeeds, are manifold. This makes it extremely hard to identify the right features to use. In sharp contrast to many other data mining problems, this is not a simple problem of feature reduction but really a problem of finding suitable ways to describe molecules. In the past biologists and chemists have spend much time developing just the right ways to describe molecules, ranging from simple one-dimensional measurements to enormously complex thousand dimensional descriptors:

*Chair for Programming Systems (Inf 2), University of Erlangen-Nuremberg, Martensstrasse 3, 91058 Erlangen, Germany, meinl@informatik.uni-erlangen.de

†School of Computer Science, Otto-von-Guericke-University of Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany, borgelt@iws.cs.uni-magdeburg.de

‡Department of Computer Science, University of Konstanz, 78457 Konstanz, Germany, berthold@inf.uni-konstanz.de

- simple one-dimensional descriptors measure molecular weight, number of hydrogen donors or acceptors, or rotatable bonds among many others.
- 2D descriptors model the connectivity of a molecule. Prominent examples are binary feature vectors where each bit represents a specific constellation of atoms, for example interesting substructural elements such as aromatic rings or amino groups but also simpler features such as specific atom-atom pairs. These so-called *fingerprints* can range from a few hundred to several thousand bits.
- 3D shape descriptors try to map a molecule to a 3D grid and attempt to model differences in geometry, that is their physical coverage of 3D space. Related approaches measure electrostatic properties at points in 3D space or other surface properties.

It is obvious that none of the methods above will be able to model all possible aspects of possible interactions between molecules. Sometimes simple 3D shape is sufficient – although quite often this only matters for part of a molecule, making matters complicated again since a possible similarity measure would need to weight different parts of the molecules differently. Sometimes the part of a molecule which is important can be described through the combination of a few bits of a fingerprint or by a small 2D fragment, literally a subgraph of the entire molecule.

This latest approach is particularly interesting to the chemist because the resulting model can be easily interpreted.

1.2 Mining Molecular Fragments

Finding fragments in a database of chemical structures is inherently different from the classical task of building a classification model for an arbitrary dataset. The latter focuses on finding one common model that summarizes the underlying dependencies for the entire data set. However, such an approach is futile for many chemical data sets[1] because the underlying model consists of many different modes of action, which would need to be modeled independently. In addition it is almost certain that not all data points will be explainable at all, thus making the usual target of perfect classification impossible (and unimportant!) to achieve. It is therefore much more suitable to extract individual, local models ("bits&pieces of evidence") that describe different types of chemical or biological interaction that result in the same outcome, e.g. inhibition of a certain activity in a drug discovery context.

For the extraction of frequent or discriminative fragments, various methods have been described recently. All of them are based on methods borrowed from the association rule mining community, in particular the Apriori algorithm [1] and the Eclat approach [14]. Whereas Apriori essentially implements a breadth-first search, Eclat follows a depth-first approach. The difference to the classical application of these algorithms – finding frequent occurrences of bits in large collections of high-dimensional bitvectors – can be summarized nicely when looking at the two main steps of both methods, namely Candidate Generation and Support Computation.

- *Candidate Generation*: Generating new fragments is inherently based on the previous set of smaller fragments. In a bit vector based domain such candidate generation is relatively

[1] We simplify things slightly by ignoring very focused libraries that target one single mode of action very late in the drug discovery process.

straightforward, for graphs this becomes a more challenging task, since there are potentially many different candidates to consider and it is not trivial to avoid generation of duplicates.

- *Support Computation*: Again, this step is relatively easy for bit vectors. For graphs, however, the support computation requires a test on subgraph isomorphism, which essentially requires an embedding[2] of a fragment into each molecule in the database. Subgraph embedding has been shown to be NP complete [4], so this becomes – especially for larger fragments – prohibitively expensive.

Quite a number of different approaches exist to date to find frequent fragments in molecular databases. Most of them concentrate only on a subset of the problems mentioned above, in particular most of them ignore the problem of support computation and rely on available graph embedding toolkits, which makes them applicable to finding small fragments only since graph embedding is computationally extremely expensive if not optimized carefully. Some examples are MolFea [7], FSG [8], gSpan [12], MoFa [2] and the relatively new FFSM [6]. A more detailed discussion on this class of algorithms can be found in [11]. All these algorithms have in common that they operate on graphs. Besides there are also other approaches relying for instance on methods from Inductive Logic Programming (ILP), where molecules are essentially encoded as lists of basic facts and the result is a combination of facts (usually based on first order logic) which is compatible with both, positive and negative examples [3].

In the next sections we want to concentrate on one of the graph based approaches, MoFa, and have a deeper look into it.

1.3 Mining Closed Fragments using MoFa

In the following sections we will describe how an approach presented earlier in [13] can be used to speed up MoFa considerably. The method described in [13] concentrates on so-called *closed fragments*, that is, fragments where no larger super-fragment occurs in exactly the same examples of the molecular database. Doing this allows them to prune their search tree, which achieves speedups of 1-2 orders of magnitude. However, the resulting algorithm is still restricted to the discovery of fairly small fragments. In addition, they only report results on small subsets ($\approx$ 1000 molecules) of the HIV database [9], suggesting problems with larger databases.

In this paper we show how the concept of closed fragments can be incorporated in the depth-first mining algorithm used by MoFa. We introduce *perfect extensions*, that is, extensions that do not alter the number of occurrences in the underlying database. Such perfect extensions will be executed before any other alternative is explored, which results in substantial speedups.

In the following we will summarize the algorithm underlying MoFa before we show experimental results on the National Cancer Insititute's HIV data [9].

[2] An *embedding* is the same as a subgraph isomorphism but normally the isomorphism is cached for later reuse thus the different name.

2 Fragment Mining with MoFa

As stated above, the goal of molecular fragment mining is to find discriminative fragments in a database of molecules, which are grouped into different classes, usually active or inactive. To achieve this, the algorithm presented in [2] represents molecules as attributed graphs and performs a depth first search on a tree of fragments. Stepping down one level in this search tree corresponds to extending a fragment by adding a bond and maybe an atom (no new atom is added if we close a ring for example). The important difference to other approaches is that for each fragment a list of embeddings into the available molecules is maintained. From this list, the subsequent list of embeddings for all its extensions when stepping down along a branch of the fragment tree can easily be constructed. As a consequence, expensive re-embeddings (i.e. subgraph isomorphism tests) of fragments are not necessary and we only generate fragments that occur in at least one molecule of the database. This maintenance of current embeddings is the main reason why this method outperforms other approaches – especially for larger fragments. The support of a fragment (the number of molecules it is contained in) is then determined by simply counting the number of different molecules these embeddings refer to. If the support of a fragment is high in the set of active molecules and low in the set of inactive molecules it is reported as a discriminative fragment. Note that in [2] it is also mentioned that in most cases it is sufficient to report only those fragments for which the support values are different from the previous node, thus already reporting what is denoted as *closed fragments* in [13]. The important ingredients of the algorithm are different search tree pruning methods, which can be categorized as follows:

- *size based pruning*: which simply cuts off branches when nodes represent fragments with more than a predefined number of bonds and/or atoms (and is seldomly used),
- *support based pruning*: which cuts off branches for which fragments do not have embeddings in a sufficiently large number of molecules (this corresponds to the usual support based pruning in association rule mining), and
- *structural pruning*: which is the most important and unfortunately also most complicated part. It is based on a definition of local orderings for the extensions of a fragment, which eliminates most, but not all, generations of redundant fragments. Since we generate arbitrarily formed, connected subgraphs, we need to avoid the generation of the same fragment in different branches of the search tree. In a depth first search we would not be able to delete fragments. See [2] for details.

Since for the following the traversal of the search tree is important, let us briefly discuss a small example. Figure 1 shows the amino acids clycin, cystein and serin (hydrogens and charges are neglected). The upper part of the tree (or forest if the empty fragment at the root is removed) which is traversed by our algorithm for these molecules is shown in Figure 2. The first level contains individual atoms, the second connected pairs of atoms and so on. The dots indicate subtrees that are not depicted in order to simplify the figure. The numbers next to these dots list the number of remaining fragments in these subtrees, indicating the total size of the tree.

The order in which the atoms on the first level of the tree are processed, is determined by their frequency of occurrence in the molecules. The least frequent atom type is considered first. Therefore the algorithm starts on the left by embedding a sulfur atom into the example molecules. That is, the molecules are searched for sulfur atoms and their locations are

Figure 1: The amino acids clycin, cystein and serin

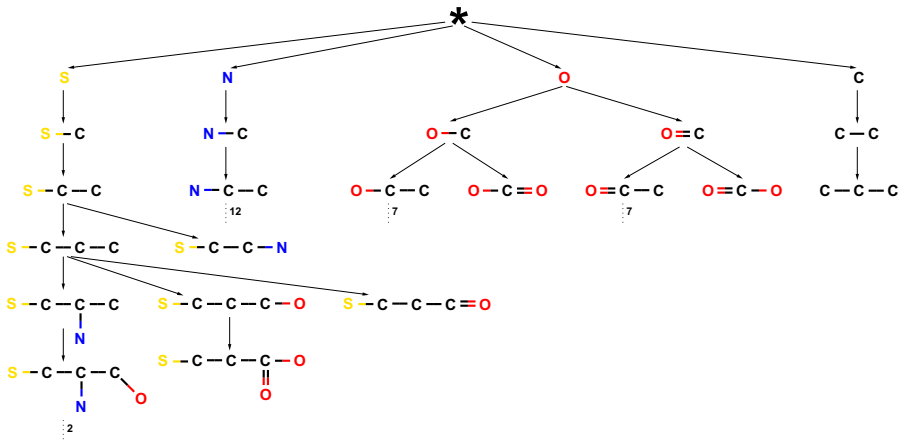

Figure 2: The tree of fragments for the amino acids example

recorded. In our example there is only one sulfur atom in cystein, which leads to one embedding of this (one atom) fragment. This fragment is then extended (depth first search) by a single bond and a carbon atom (`-C`), which produces the fragment `S-C` on the next level. All other extensions of fragments that are generated by going down one level in the tree are created in an analogous way.

If a fragment allows for more than one extension (as is the case, for instance, for the fragments `O-C` and `S-C-C-C`), we sort them according to the local ordering rules mentioned above. The main purpose of this local order is to prevent certain extensions to be generated, in order to avoid redundant search. For instance, the fragment `S-C-C-C-O` is not extended by adding a single bond to a nitrogen atom at the second carbon atom, because this extension has already been considered in the subtree rooted at the left sibling of this fragment.

Furthermore, in the subtree rooted at the nitrogen atom, extensions by a bond to a sulfur atom are ruled out, since all fragments containing a sulfur atom have already been considered in the tree rooted at the sulfur atom. Similarly, neither sulfur nor nitrogen are considered in the tree rooted at the oxygen atom, and the rightmost tree contains fragments that consist of carbon atoms only.

Up to now we only described how the search tree is organized, i.e., the manner in which the candidates for discriminative fragments are generated and the order in which they are considered. However, in an application this search tree is not traversed completely – that would be much too expensive for a real world database. Since a discriminative fragment must be frequent in the active molecules and extending a fragment can only reduce the support (because only fewer molecules can contain it), subtrees can be pruned as soon as the support falls below a user-defined threshold (support based pruning).

Discriminative fragments should also be rare in the inactive molecules, defined formally by an user-specified upper support threshold. However, this threshold cannot be used to prune the search tree: even if a fragment does not satisfy this threshold, its extension may (again

extending a fragment can only reduce the support), and thus it has to be generated. Therefore this threshold is only used to filter the fragments that are frequent in the active molecules. Only frequent fragments that satisfy this threshold are reported as discriminative fragments.

In [2] the original form of this algorithm was applied to the NCI-HIV database with considerable success. Several discriminative fragments could be found, some of which could be related to known classes of HIV inhibitors. These and other experiments have demonstrated that MoFa is usually extremely fast.

3 Perfect extension pruning

3.1 Closed fragments and perfect extensions

The concept of closed itemsets was first introduced in [10] and can easily be transferred to graphs as was recently shown in [13]. The idea behind it is quite simple: A subgraph (or fragment) is said to be closed if there does not exist a supergraph that has the same support values. Figure 3 illustrates this defintion. It shows part of the search tree MoFa creates for the

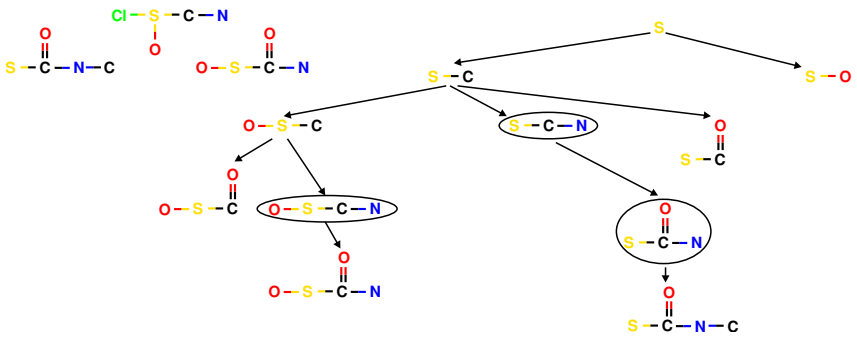

Figure 3: Part of a MoFa search tree with closed fragments being highlighted

three molecules shown in the top left corner. The tree is built following the rules explained in section 2. Three of the fragments are circled, these are closed fragments. `S-C-N` for example occurs in the same number of molecules as its ancestors `S-C` and `S`. Hence those two are not closed. On the other hand, any fragment that is bigger than `S-C-N` has a lower support, so it is indeed a closed fragment. The same holds for the other two circled structures. Obviously every complete molecule is a closed fragment as well, so we do not mention this explicitly.

The fact that not all subgraphs in the lattice are closed can now be used to prune the search tree in a dramatic fashion if the user is only interested in these closed structures. For most applications of finding fragments in molecular databases this is indeed the main focus since the user is interested in the largest, discriminative substructure and not the smallest or other intermediate ones.

To make use of this new constraint we first introduce so-called *perfect extensions*. An extension of an existing fragment is perfect, if the following three conditions hold:

1. The number of embeddings must be the same as the embeddings of the parent fragment,
2. the number of supported molecules must be the same as for the parent fragment,
3. the number of embeddings in each single molecule must be the same as for the parent fragment.

Of course, for all fragments fulfilling condition 3 the other two conditions hold as well. We mention them separately because the test for the number of embeddings is very cheap (we only have to compare the length of the embedding lists) and thus can be used as a shortcut criterion. The check for the second condition is also cheaper than the third test. Hence the test for unperfect extensions is quite fast whereas a positive result needs more computations.

The reason why the third condition is crucial will become clear once we have explained the complete pruning process in the next section.

3.2 The pruning process in MoFa

The idea underlying closed fragment pruning in MoFa is the following: After the extensions for an already discovered fragment have been created (and pruned using the standard techniques) MoFa checks if there are any perfect extensions among them. Once such a perfect extension has been found *all* other extensions are deleted and only the fragment belonging to the perfect one is extended further. If we apply this step to the search tree in Figure 3 we can prune it, resulting in the one shown in Figure 4. The highlighted fragment S-C is a perfect

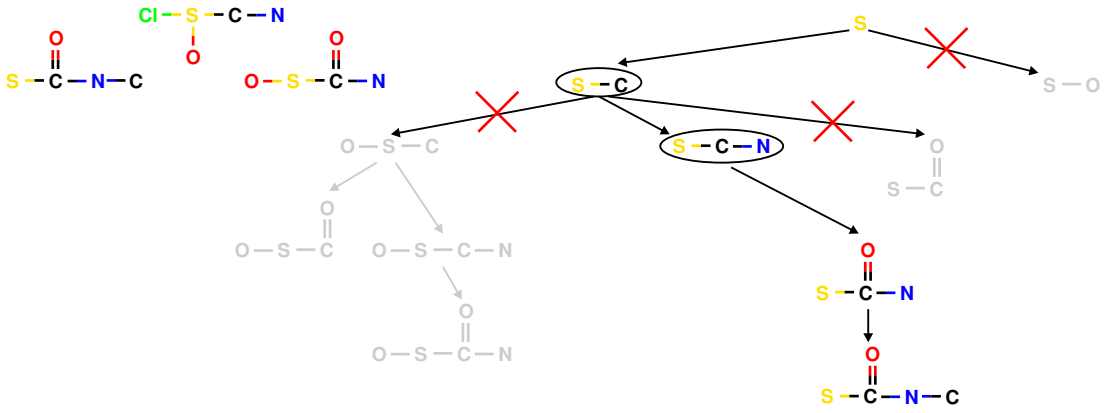

Figure 4: The pruned search tree taking into account perfect extensions.

extension of S and therefore the other branch that yields S-O is pruned. Note that S-C is a *perfect extension* of S but is not a *closed fragment*. The other circled structure S-C-N is also a perfect extension (of S-C) which is why the other two branches can also be deleted. (S-C-N is also a *closed fragment*, however, this is irrelevant for the pruning process.)

It is easy to see that even for this small example, the search tree is substantially smaller than before. However, you may have also noticed that we do not find all closed fragments anymore. The fragment O-S-C-N in the left branch is missing and also its descendant, the complete molecule. The reason for this undesirable behaviour is that the structural pruning interferes with the closed fragment pruning. We cannot add the oxygen atom to S-C-N because the last extended atom (*constrained atom*) was the carbon atom and the sulfur atom has been added to the fragment before the carbon atom. The rules of structural pruning forbid any extensions that start at atoms that have been added to a fragment before the constrained atom. Therefore we have to change these rules slightly: if branches are deleted because a perfect extension exists on the same level, the constrained atom is *not* set to the atom at which the extension took place but instead stays the same as in the parent fragment. In effect we are making sure that all extensions after the perfect extension remain applicable. If we apply this rule to the example from before the result is the new search tree as shown in Figure 5. Now it

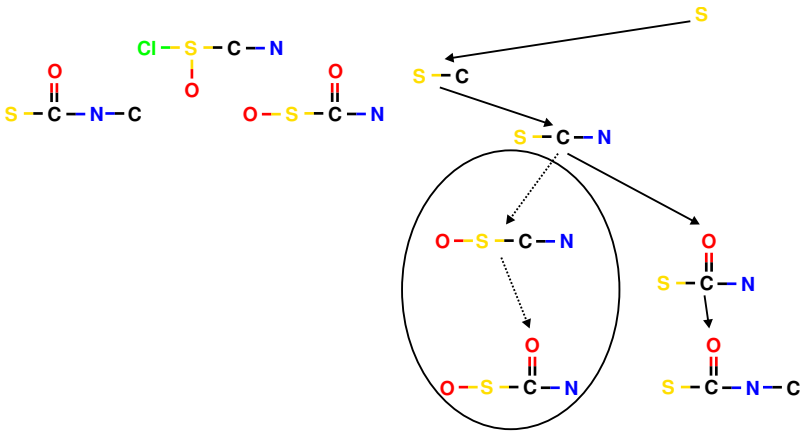

Figure 5: The new, correct search tree after the application of closed fragment pruning

is possible to add the oxygen atom to the perfect extension `S-C-N` and a new branch is initiated (circled and dashed arrows). In this branch MoFa finds the closed fragment `O-S-C-N` and the complete molecule that were missing before. Instead of looking for extensions of 11 fragments in the original search tree only 7 fragments had to be examined in the new case. How this technique accelerates the mining process on real world datasets is shown in section 4.

Besides also the size based pruning will interfere with closed fragment pruning but this does not lead to undiscovered fragments like with the structural pruning. If you cut off branches because the structures get larger than the user defined size, the biggest fragment will get reported but it need not necessarily be a closed one.

Finally we need to explain why the third condition above is indeed needed. That constraint states, that the number of embeddings *into each single molecule* has to be the same for the parent and child fragment. Let us assume that we mine on the two molecules shown on the top of Figure 6.

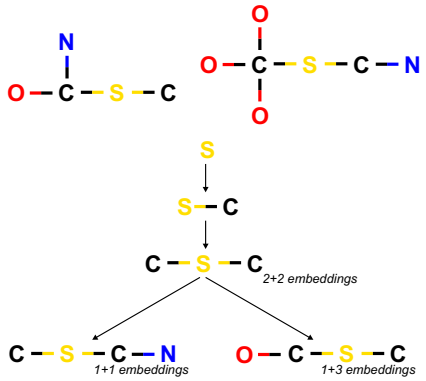

Figure 6: Non-perfect extensions.

The current fragment is `C-S-C`, which has two embeddings in the left molecule and two in the right. We can create two different extensions, `C-S-C-N` and `O-C-S-C`. The first extension does not fulfill condition 1 for perfect extensions. The second one fulfills the first two conditions but the distribution of the embeddings in the two molecules has changed. In the left molecule only one embedding remains but in the other one we now have three embeddings. If we did not have the third condition, this extension would be perfect and we would delete the other branch. But then the other fragment – which is a closed one – would never be found.

4 Experimental Results

To show the effect of *perfect extension pruning* we ran experiments on the well known HIV datasets from the NCI (National Cancer Institute [9]). As of March 2002, this library contains

43,905 molecules tested for their reaction against the HI-virus. They are grouped into three classes: 423 belong to the class CA (confirmed active), 1083 to CM (confirmed medium active) and the rest belongs to CI (confirmed inactive)[3].

The standard application of MoFa is to find discriminative fragments. That is, we would usually mine on molecules of class CA/CM of the HIV-data – the *focus* – but at the same time also carry along the embeddings in all molecules of the other, inactive class, the *complement*. Thus MoFa would in effect mine the entire dataset of nearly 44,000 molecules. However, the results presented in [13] were achieved by only mining on the classes CA and CM, in total 1503 molecules. Therefore they only find fragments that are frequent in the active (and medium active) molecules but they cannot include any upper limit on the frequency in the inactive molecules. In order to be able to present comparable results we initially also restrict the database to the structures in CA and CM and neglect all molecules in class CI. However, in the following section we will also discuss results on the entire data base.

All experiments were conducted using the ring mining feature presented in [5]. We interrupted experiments with a running time exceeding two hours (which is a more or less abritrary choice). The experiments were performed on an Athlon XP 1800+ with 1GB RAM. MoFa was run under Windows 2000 with Java 1.4.1 and a maximum heap of 750MB, gSpan[4] was run under SuSE Linux 8.0.

4.1 Results: Finding Frequent Fragments

Figure 7 shows the results on the HIV-dataset. The influence of perfect extension pruning is not noticeable with minimum support thresholds greater than 3%. However, for lower support threshold values the effect is quite dramatic. For example at 0.8% minimum occurrence, MoFa with perfect extension pruning is more than eight times faster than without. It also does not pose any problems to find even less frequent structures in reasonable time whereas without the new pruning strategy the mining process used to take more than two hours.

We also compared MoFa with gSpan on this dataset[5]. If the minimum support is higher than 3.0% gSpan is faster than MoFa (with or without perfect extension pruning) but if the threshold is lowered, MoFa outperforms gSpan. It is 30 times faster at a threshold of 2% and gSpan does not terminate within two hours for lower support values.

The reason why the impact of the new pruning strategy is not quite as impressive as reported for gSpan in [13] lies in the already very effective structural pruning. Only for relatively large fragments that have many branches perfect extension pruning accelerates the mining process noticeably.

4.2 Results: Finding Discriminative Fragments

In order to demonstrate the performance of the proposed pruning strategy in the context of mining discriminative fragments, we also performed experiments on the entire HIV database. This topic is even more interesting to the end user, as a ,,good" fragment should be frequent in one class (the *focus*) and infrequent in the other (the *complement*). Looking at the HIV data

[3]The authors in [13] mention only 1503 CA+CM molecules in the March 2003 HIV database, however, the three additional structures should not affect the results noticeably.

[4]An executable to run gSpan was kindly provided by Xifeng Yan and Jiawei Han.

[5]Keep in mind, that gSpan uses C++ code whereas MoFa is implemented in Java.

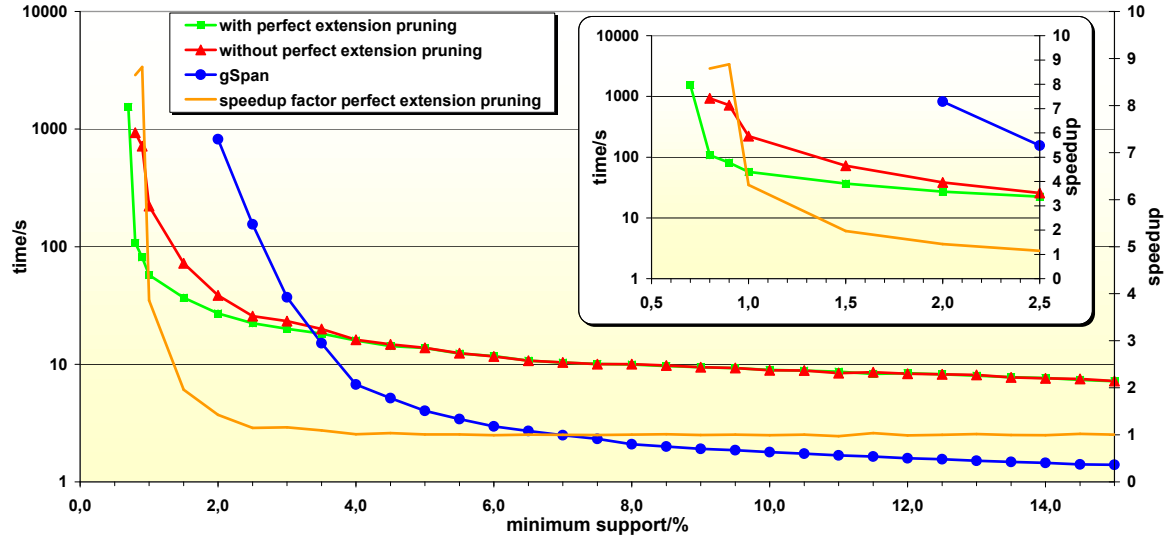

Figure 7: The mining time on the NCI-HIV (classes CA/CM) data with and without perfect extension pruning compared to gSpan. The area between 2.5 and 0.5 % is shown in greater solution in the upper right corner of the diagramm.

that means, that a good candidate fragment is often found in molecules from classes CA and CM and rarely in molecules from CI. This makes the search process more complex as you now have to search in more than 40.000 molecules.

Figure 8 shows the results. We varied the support threshold in the focus group (classes CA and CM, 1506 molecules) from 10% to 1% and fixed the threshold in the complement class (class CI, $42,400$ molecules) to 0.1%. In order to allow a comparison we also included the experiments mining for frequent fragments in classes CA and CM only.

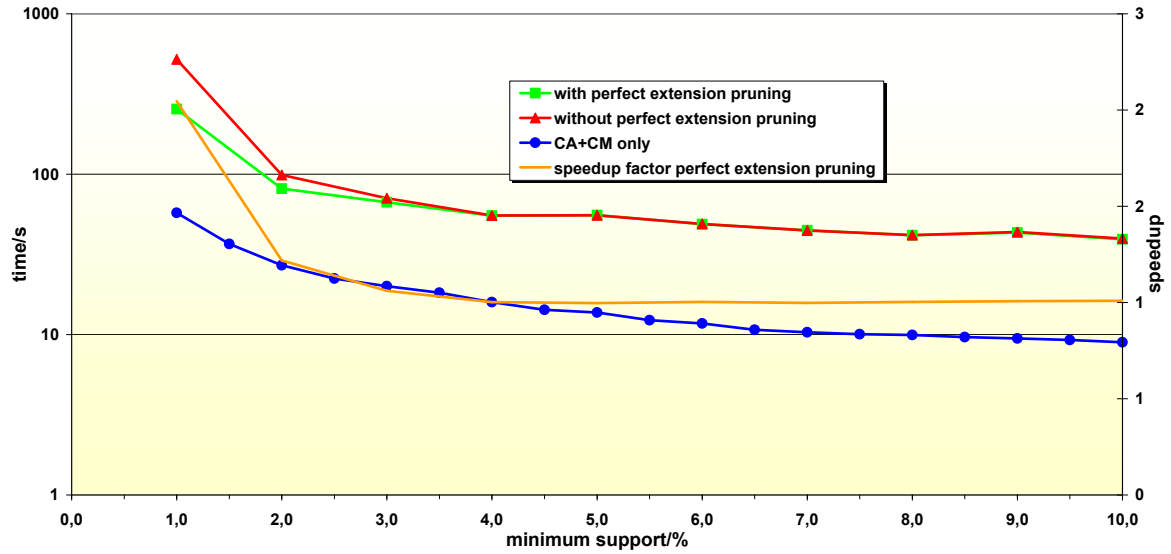

Figure 8: The mining time on the complete NCI-HIV data with and without perfect extension pruning.

Note how the behaviour stays essentially the same. For higher support thresholds, closed fragment pruning and perfect extensions do not result in a speedup. However, as before, for

thresholds below $\approx 3\%$, an ever increasing speedup is obvious.

We can not compare these results with gSpan, since the full data set was not available in a format accepted by the available software. However, we would expect a similar, if not better speed-up as before since gSpan will need to compute the frequencies on the complement database by full embeddings, whereas MoFa produces these embeddings in parallel throughout the search process.

5 Conclusion and outlook

We have shown that it is possible to mine meaningful, discriminative molecular fragments from large databases. Using an existing algorithm that employs a depth-first strategy and a sophisticated ordering scheme allows to avoid costly re-embeddings throughout the candidate growth process, which in turn enables us to find also larger fragments. Employing the closed fragment concept discussed in [13] stimulated the development of the concept of perfect extensions, which – in some cases – resulted in speed ups of several orders of magnitude. We have demonstrated how the resulting method finds discriminative fragments in molecular databases of several tens of thousands of molecules within acceptable time.

For the future it would be very interesting, to make a much more detailed analysis of the whole graph based data mining topic, comparing all of the currently known algorithms and how such special extension like closed fragments can be included in the search. Therefore one needs a standardized framework so that the performance of each individual approach is not so much dependent on the actual implementation. Another issue that could be addressed by this are the different data formats used, which imposes problems if the actual data set is not in the required form.

6 Acknowledgements

We would like to thank George Karypis and his group for interesting discussions and Jiawei Han and his group for feedback and a public domain version of their gSpan algorithm.

References

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., USA, 1993. ACM Press.

[2] C. Borgelt and M. R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Proceedings of the IEEE International Conference on Data Mining ICDM*, pages 51–58, Piscataway, NJ, USA, 2002. IEEE Press.

[3] Paul W. Finn, Stephen Muggleton, David Page, and Ashwin Srinivasan. Pharmacophore discovery using the inductive logic programming system PROGOL. *Machine Learning*, 30(2-3):241–270, 1998.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[5] H. Hofer, C. Borgelt, and M. R. Berthold. Large scale mining of molecular fragments with wildcards. In *Advances in Intelligent Data Analysis V*, number 2810 in Lecture Notes in Computer Science (LNCS), pages 380–389. Springer Verlag, 2003.

[6] Jun Huan, Wei Wang, and Jan Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. pages 549–552, 2003.

[7] S. Kramer, L. De Raedt, and C. Helma. Molecular feature mining in hiv data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 136–143. ACM Press, 2001.

[8] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proceedings of the IEEE International Conference on Data Mining ICDM*, pages 313–320, Piscataway, NJ, USA, 2001. IEEE Press.

[9] HIV antiviral screen. http://dtp.nci.nih.gov/docs/aids/aids_data.html.

[10] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. *Lecture Notes in Computer Science*, 1540:398–416, 1999.

[11] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.

[12] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the IEEE International Conference on Data Mining ICDM*, pages 51–58, Piscataway, NJ, USA, 2002. IEEE Press.

[13] Xifeng Yan and Jiawei Han. Closegraph: Mining closed frequent graph patterns. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, page (to appear). ACM Press, August 2003.

[14] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. In David Heckerman, Heikki Mannila, Daryl Pregibon, Ramasamy Uthurusamy, and Menlo Park, editors, *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 283–296. AAAI Press, 12–15 1997.