Advanced Pruning Strategies to Speed Up Mining Closed Molecular Fragments[*]

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

Thorsten Meinl
Computer Science Department 2
University of Erlangen-Nuremberg
Martenstraße 3, 91058 Erlangen, Germany
meinl@cs.fau.de

Michael R. Berthold
Dept. of Computer and Information Science
University of Konstanz
78457 Konstanz, Germany
berthold@inf.uni-konstanz.de

Abstract – *In recent years several algorithms for mining frequent subgraphs in graph databases have been proposed, with a major application area being the discovery of frequent substructures of biomolecules. Unfortunately, most of these algorithms still struggle with fairly long execution times if larger substructures or molecular fragments are desired. In this paper we describe two advanced pruning strategies —* equivalent sibling pruning *and* perfect extension pruning *— that can be used to speed up the MoFa algorithm (introduced in [2]) in the search for closed molecular fragments, as we demonstrate with experiments on the NCI's HIV database.*

Keywords: molecular fragment, closed fragment, graph mining, pruning, perfect extension.

1 Introduction

A frequent task in biochemistry is the search for common features in large sets of molecules. Examples are drug discovery, where one is interested in identifying properties shared by molecules that have been classified as "active" w.r.t., for example, the protection of human cells against a virus, and compound synthesis, where one tries to identify properties that enable or inhibit the synthesis of new molecules, so that one can predict the chances for a successful synthesis.

Since the features one may use to describe molecules are manifold, there are approaches in abundance, ranging from simple one-dimensional measurements to highly complex thousand-dimensional descriptors. The molecular weight and the number of hydrogen donors or acceptors are examples of simple features. More complex ones include binary vectors, which can be several thousand bits long with each bit representing a specific constellation of atoms like aromatic rings or amino groups, as well as shape descriptors that try to capture geometric properties of a molecule.

In this paper we focus on an approach that models molecules as attributed graphs, thus taking the connection structure, though not the 3-dimensional structure into account. The resulting set of graphs is then searched for common subgraphs, that is, molecular fragments, that appear with a user-specified minimum frequency. For this approach several algorithms have been proposed recently, with many of them based on methods developed for association rule mining. In particular, the Apriori algorithm [1] and the Eclat algorithm [13] are taken as starting points. The general ideas of these algorithms can be transferred to molecular substructure mining, even though the fact that the input consists of graphs instead of sets poses some problems. Examples of algorithms developed in this way are MolFea [6], FSG [7], gSpan [12], MoFa [2] and FFSM [5]. Other approaches rely, for instance, on principles from inductive logic programming [3] and describe the graph structure by logical expressions.

A common problem of all approaches is that the task is highly complex and therefore careful optimization is necessary to achieve bearable execution times. In this paper we consider optimizations for the MoFa algorithm [2], which consist in two advanced pruning strategies, namely *equivalent sibling pruning* and *perfect extension pruning*. These methods can lead to considerable speed-ups in the search for closed molecular fragments, with the latter being particularly useful when the considered fragments become larger.

2 Molecular Fragment Mining

As stated above, the algorithm presented in [2] represents molecules as attributed graphs. It then carries out a depth first search on a tree of fragments (Eclat-based approach).

[*]**0-7803-8566-7/04/$20.00 © 2004 IEEE.**

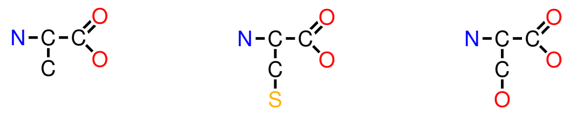
Figure 1: The amino acids clycin, cystein and serin.

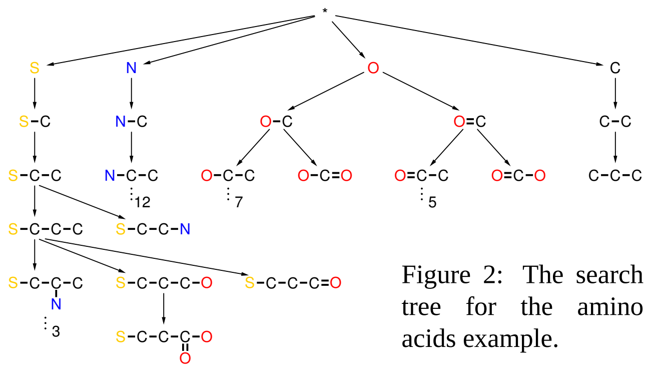
Figure 2: The search tree for the amino acids example.

Going down one level in this search tree means extending a fragment by adding a bond and maybe an atom to it. For each fragment a list of embeddings into the available molecules is maintained, from which the lists of embeddings of its extensions can easily be constructed. As a consequence, expensive re-embeddings of fragments are not necessary. The support of a fragment, that is, the number of molecules it is contained in, is then determined by simply counting the number of different molecules the 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.

The important ingredients of the algorithm are different search tree pruning methods, which can be grouped into three categories: *size based pruning*, *support based pruning*, and *structural pruning*. Among these the latter is the most important and most complicated one. It is based on a set of rules that defines a local order of the extensions of a fragment, which is used to avoid redundant searches. (See [2] for details.)

Since the way in which the search tree is traversed is very important for the following explanations, let us briefly discuss a simple example. Figure 1 shows the amino acids clycin, cystein and serin with hydrogens and charges neglected. The upper part of the tree (or forest if the empty fragment at the root is removed) that is traversed by our algorithm for these molecules is shown in Figure 2. The first level contains individual atoms, the second connected pairs and so on. The dots indicate subtrees that are not depicted in order to keep the figure understandable. The numbers next to these dots state the number of fragments in these subtrees, giving an indication of 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 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 (see [2] for details). The main purpose of this local order is to prevent certain extensions to be generated, in order to avoid redundant searches. 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 (leftmost branch). 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 frequent 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 reported fragment must be frequent 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). Furthermore, the depth of the tree may be restricted, thus limiting the size of the fragments to be generated (size based pruning).

It should be noted that the approach can easily be extended to find *discriminative fragments*, i.e., fragments that are frequent in the active molecules and infrequent in a complementary set of inactive molecules, defined formally by user-specified lower and upper support thresholds, even though the latter cannot be used to prune the search tree. (See [2] for details.)

3 Advanced Pruning Strategies

As described in [2], the structural pruning scheme employed in MoFa cannot rule out all redundant search. However, even though there is no simple way to suppress *all* redundant search, there are some possibilities for improvement, two of which we describe in this section. The first we call *equivalent sibling pruning*, which checks for equivalent children of a search tree node, the second is *perfect extension pruning*, which follows only one branch of the search tree if the corresponding extension satisfies certain criteria.

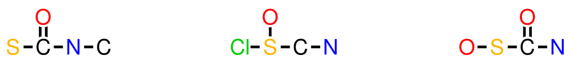

Figure 3: Some (fictitious) example molecules.

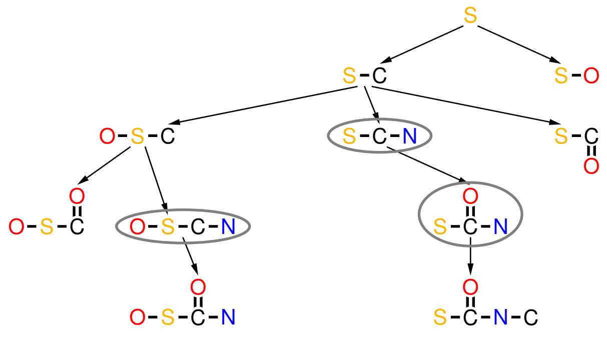

Figure 4: Search tree for the molecules in Figure 3.

Note that the first can be applied even if the search is not restricted to closed fragments, while the second presupposes that non-closed fragments can be discarded.

3.1 Closed Molecular Fragments

The notion of a *closed fragment* is derived from the corresponding notion of a *closed item set*, which is defined as an item set no superset of which has the same support, i.e., is contained in the same number of transactions. Analogously, a *closed fragment* is a fragment no superstructure of which has the same support, i.e., is contained in the same number of molecules. As an example consider the three example molecules shown in Figure 3 and the corresponding (unpruned) MoFa search tree (starting from sulfur as a seed) shown in Figure 4: the closed fragments (for a minimum support of two molecules, i.e., 66%) are circled.

As for item sets, restricting the search for molecular fragments to closed fragments does not lose any information: all frequent fragments can be constructed from the closed ones by simply forming all substructures of closed fragments that are not closed fragments themselves and assigning to them as their support the maximum of the support values of those closed fragments of which they are substructures. Consequently, restricting the search to closed fragments is a very convenient way to reduce the size of the output. In addition, chemists are usually not interested in all frequent fragments, but only in the closed ones, presumably because they contain all relevant information without redundancy.

Note that in [2] we already restricted the search to closed fragments, even though we did not use the term *closed fragment* in that paper. Recently [12] studied the advantages of a restriction to closed fragments in more detail. The main advantage of restricting the search to closed fragments, certain pruning techniques become applicable, which can speed up the search considerably.

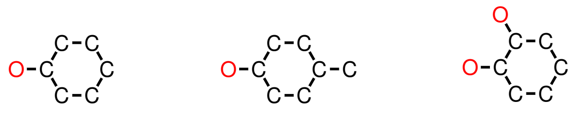

Figure 5: Three phenols: phenol, p-cresol, and catechol.

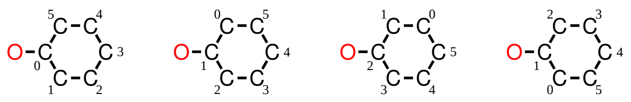

Figure 6: Equivalent extended embeddings.

3.2 Equivalent Sibling Pruning

In order to suppress all redundant search, one would have to check whether any two subtrees of the search tree have the same fragment as their root (or a fragment which only differs in the restrictions placed on its possible extensions), so that only one (namely the least restricted one) is actually searched. Such a general check, however, is costly, because one would have to store all fragments that have been considered in the search so far. In addition, one would need an efficient way of checking these stored fragments for an equivalent one. Finally, it is difficult to find out whether a subtree to be searched is the least restricted one appearing in the whole search tree, even though the depth first traversal applied in MoFa always considers the least restricted sibling node first.

However, what can be checked fairly easily is whether *two sibling nodes* in the search tree correspond to fragments that are equivalent (represent the same substructure) and differ only in the restrictions placed on their possible extensions. That such a situation can actually occur can be seen from the example molecules shown in Figure 5. Suppose we start the search by embedding a benzene ring into these molecules. Since such a ring exhibits a high symmetry (which is a necessary prerequisite for equivalent siblings), it can be embedded into each of the molecules in twelve different ways (the ring can be rotated to six positions and it can be traversed in two directions). Consider now the extensions of this benzene ring: each of the twelve embeddings is extended by a bond and an oxygen atom. This leads to twelve new fragments, all of which are equivalent and differ only in the label of the ring atom that was extended (see Figure 6 for examples). Obviously it suffices in this situation to consider the fragment in which the ring atom with the smallest number was extended. All other fragments must lead to redundant search, because they allow for a subset of the possible extensions, but no additional ones.

Since the MoFa algorithm considers sibling nodes w.r.t. a local order that is based on the extension information of the parent fragment (i.e. bond and atom type, number of the atom the bond is incident to) and processes the least restricted extensions first, it is fairly easy to implement the described pruning approach: for each node its preceding sibling nodes are checked for equivalence and if an equivalent sibling is found, the current node is skipped.

The equivalence test is carried out as follows: for the fragment corresponding to the current node an arbitrary embedding into an arbitrary molecule is selected. In the corresponding molecule all bonds and atoms belonging to this embedding are marked and all other bonds and atoms are unmarked. Then all embeddings of the fragment corresponding to a sibling node that is to be checked for equivalence are traversed. For each of these embeddings it is checked whether it refers only to marked atoms and bonds, that is, whether it essentially represents the same substructure. If such an embedding can be found, the two fragments are equivalent and consequently the current node (the extensions of which are more severely restricted than those of its already processed sibling) can be skipped.

Note that it suffices to check one molecule, because if there are identical embeddings into one molecule there must be identical embeddings into all molecules referred to by the fragment. As a consequence the check is comparatively cheap and thus does not degrade performance much even if the pruning cannot be carried out.

3.3 Perfect Extension Pruning

Perfect extension pruning is based on the observation that sometimes there is a fairly large common fragment in *all* currently considered molecules. As long as the search has not grown a fragment to this maximal common one, it is not necessary to branch in the search tree.

From the definition of a closed fragment it is clear that if there is a common substructure in *all* currently considered molecules of which the current fragment is only a part, then any extension that does not grow the current fragment towards the maximal common one can be postponed until this maximal common fragment has been reached. The reason is, obviously, that the maximal common fragment is part of all closed fragments that can be found in the currently considered set of molecules. Consequently, it suffices to follow one path in the search tree that leads to this maximal common fragment and to start branching only from there.

As an example consider again the set of molecules shown in Figure 3. If the search is seeded with a single sulfur atom, considering extensions by a single bond starting at the sulfur atom and leading to an oxygen atom can be postponed until the structure `S-C-N` common to all molecules has been grown (provided that the extensions of this maximal common fragment are not restricted in any way — see below).

Technically, the search tree pruning is based on the notion of a *perfect extension*. An extension of a fragment, consisting of a bond and possibly an atom (if the bond does not close a ring), is called *perfect* if all of its embeddings can be extended in exactly the same way by this bond and atom. (Note that there may be multiple ways of extending an embedding by this bond and atom. In this case all embeddings must be extendable in the same number of ways.) If there is

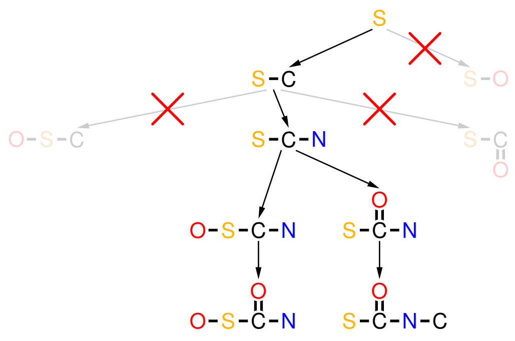

Figure 7: Search tree with perfect extension pruning.

a perfect extension of a fragment, all closed fragments can, in principle, be found by searching only the corresponding branch.

There is, however, a minor complication, because perfect extension pruning interferes with the normal structural pruning done in MoFa. Normal structural pruning prevents extensions of a fragment by bonds that start from atoms with smaller numbers than the one extended in the preceding step. However, a perfect extension should not lead to such a restriction, because otherwise some search results are lost.

As an example consider again the search tree shown in Figure 4. If we simply confined the search to the subtree rooted at the fragment `S-C-N`, we would lose the fragment `O-S-C-N` in the left branch. The reason is that the extension of `S-C` to `S-C-N`, due to the normal structural pruning rules, prevents an extension of the sulfur atom in this subtree, because an atom with a higher number, namely the carbon atom, has been extended in the preceding step.

However, a perfect extension should not restrict possible future extensions of the fragment in this way. Therefore the extension information of a fragment obtained by a perfect extension are set to those of its parent fragment, bypassing the normal structural pruning rules. In the considered example, this allows us to extend the fragment `S-C-N` by bonds starting from the sulfur atom and thus we get the search tree shown in Figure 7, in which the fragment `O-S-C-N` is found.

When it comes to implementing perfect extension pruning, one should bear in mind that checking whether an extension is perfect by testing the extensions of all embeddings is costly. Therefore we precede the actual test by cheap tests in order to, if possible, *fail early* or *succeed early*. The ideas underlying these tests are fairly simple: an extension leading to a fragment cannot be perfect (1) if the number of molecules referred to by the fragment differs from those referred to by its parent and (2) if the number of embeddings of the fragment is not an integer multiple of the number of embeddings of its parent. On the other hand, if these tests do not indicate that the extension is not perfect, we can immediately conclude that it is perfect (1) if the fragment refers only to one molecule or (2) the number of molecules referred to equals the number of embeddings (i.e., there is exactly one embed-

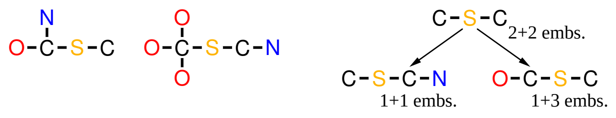

Figure 8: Examples of non-perfect extensions.

Table 1: Effects of pruning on the steroids data set.

pruning	# nodes	# frags.	# embs.
neither	58134	58134	87593
equivalent sibling	26768	27565	42926
perfect extension	4260	6766	16494
both	1561	3035	7839

ding per molecule). Only if these test do not yield a decision we actually carry out the costly test whether each parent embedding leads to the same number of extended embeddings.

Note that it is actually necessary to count the embeddings per molecule. Checking only whether the total number of embeddings into all molecules coincides with (or is an integer multiple of) the parent embeddings does not suffice as can be seen from the example shown in Figure 8. Even though the total number of embeddings is the same in the right branch, the extension is not perfect. The left extension is not perfect, because the number of extended embeddings, even though the same for each parent embedding, is reduced from the number of extensions of its parents. This indicates that some symmetry has been destroyed by the extension, which therefore cannot be perfect.

4 Experiments

In order to demonstrate the effects of the two pruning approaches, we carried out experiments on two data sets. The first is a very small data set consisting of 17 steroid molecules.[1] The effects of the two pruning methods are shown in Table 1 (all experiments were carried out with a minimum support of one molecule). As can be seen, both pruning methods have considerable effects, with those of perfect extension pruning being much more pronounced. However, this is presumable due to the very low minimum support, which makes perfect extension pruning highly effective. At higher support values equivalent sibling pruning seems to be more effective (see below).

In a second test we ran the program on the well-known HIV data set available from the National Cancer Institute [8]. This library contains about 44,000 molecules tested for their activity against the HI-virus, which are grouped into three classes: about 400 belong to class CA (confirmed active), about 1000 to CM (confirmed medium active) and the rest belongs to CI (confirmed inactive). In the experiments we report here, however, we neglected these class assignments

[1]See Section 6 below for a download URL.

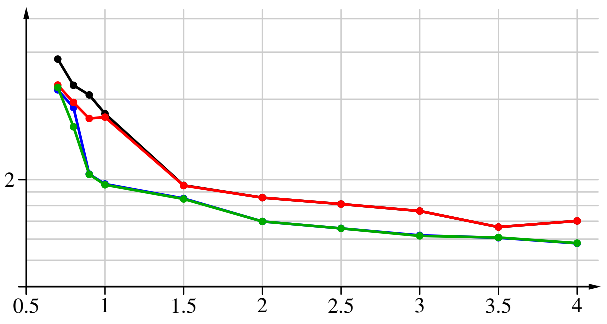

Figure 9: Execution times on the NCI's HIV data.

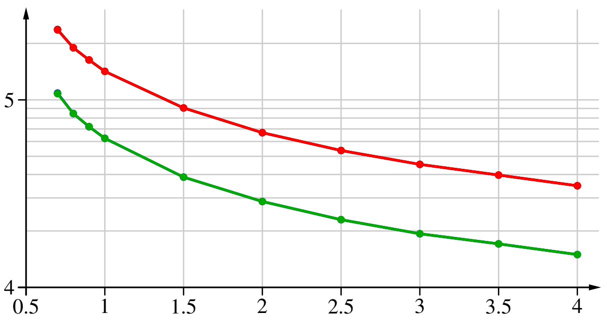

Figure 10: Numbers of nodes on the NCI's HIV data.

and tried to find common substructures of all molecules at minimum support thresholds ranging from 0.7 to 4%. We used an AMD XP 2000+ system with 756 MB RAM running S.u.S.E. Linux 9.1 and Java 1.4.2.

The results are shown in Figures 9 and 10. The former states the decimal logarithm of the execution time in seconds, the latter the decimal logarithm of the number of visited nodes in the search tree. Each diagram contains four graphs: The black line (which in Figure 9 is almost, in Figure 10 completely hidden under the red line) corresponds to no pruning. The blue line (which in Figure 9 is almost, in Figure 10 completely hidden under the green line) corresponds to equivalent sibling pruning. Finally, the red line corresponds to perfect extension pruning and the green line to both pruning methods combined.

As can be seen from Figure 9, equivalent sibling pruning yields the highest gains for higher support values, while perfect extension pruning is more effective at lower support values. It is interesting to node that the effects of perfect extension pruning are not achieved by considerably reducing the number of search tree nodes as a comparison with Figure 10 shows.

5 Conclusions

In this paper we presented two advanced pruning strategies for the MoFa molecular fragment mining algorithm: *equivalent sibling pruning* and *perfect extension pruning*. Especially the second leads to a considerable acceleration of the search for closed fragments, as the experiments demonstrate. Its drawback, namely that it is only applicable if closed fragments suffice is usually negligible, since chemists are rarely interested in non-closed fragments. The experiments show that both methods are useful and lead to considerable speed-ups.

6 Software

An implementation of the described molecular fragment mining algorithm can be retrieved free of charge at

http://fuzzy.cs.uni-magdeburg.de/~borgelt/moss.html
http://www.inf.uni-konstanz.de/bioml/projects/mofa/

Note that this implementation does not support wildcard atoms (as described in [4]). The version supporting such an option is property of Tripos, Inc., USA.

References

[1] R. Agrawal, T. Imielienski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. *Proc. Conf. on Management of Data,* 207–216. ACM Press, New York, NY, USA 1993

[2] C. Borgelt and M.R. Berthold. Mining Molecular Fragments: Finding Relevant Substructures of Molecules. *Proc. IEEE Int. Conf. on Data Mining (ICDM 2002, Maebashi, Japan),* 51–58. IEEE Press, Piscataway, NJ, USA 2002

[3] P.W. Finn, S. Muggleton, D. Page, and A. Srinivasan. Pharmacore Discovery Using the Inductive Logic Programming System PROGOL. *Machine Learning,* 30(2-3):241–270. Kluwer, Amsterdam, Netherlands 1998

[4] H. Hofer, C. Borgelt, and M.R. Berthold. Large Scale Mining of Molecular Fragments with Wildcards. *Proc. 5th Int. Symposium on Intelligent Data Analysis (IDA 2003, Berlin, Germany),* LNCS 2810:380–389. Springer-Verlag, Heidelberg, Germany 2003

[5] J. Huan, W. Wang, and J. Prins. Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism. *Proc. 3rd IEEE Int. Conf. on Data Mining (ICDM 2003, Melbourne, FL),* 549–552. IEEE Press, Piscataway, NJ, USA 2003

[6] S. Kramer, L. de Raedt, and C. Helma. Molecular Feature Mining in HIV Data. *Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2001, San Francisco, CA),* 136–143. ACM Press, New York, NY, USA 2001

[7] M. Kuramochi and G. Karypis. Frequent Subgraph Discovery. *Proc. 1st IEEE Int. Conf. on Data Mining (ICDM 2001, San Jose, CA),* 313–320. IEEE Press, Piscataway, NJ, USA 2001

[8] HIV Antiviral Screen. National Cancer Institute. http://dtp.nci.nih.gov/docs/aids/aids_data.html

[9] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering Frequent Closed Itemsets for Association Rules. *Proc. 7th Int. Conf. on Database Theory (ICDT'99, Jerusalem, Israel),* LNCS 1540:398–416. Springer-Verlag, Heidelberg, Germany 1999

[10] T. Washio and H. Motoda. State of the Art of Graph-Based Data Mining. *SIGKDD Explorations Newsletter* 5(1):59–68. ACM Press, New York, NY, USA 2003

[11] X. Yan and J. Han. gSpan: Graph-Based Substructure Pattern Mining. *Proc. 2nd IEEE Int. Conf. on Data Mining (ICDM 2003, Maebashi, Japan),* 721–724. IEEE Press, Piscataway, NJ, USA 2002

[12] X. Yan and J. Han. Closegraph: Mining Closed Frequent Graph Patterns. *Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2003, Washington, DC),* 286–295. ACM Press, New York, NY, USA 2003

[13] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97, Newport Beach, CA),* 283–296. AAAI Press, Menlo Park, CA, USA 1997