Large Scale Mining of Molecular Fragments with Wildcards

Heiko Hofer[1], Christian Borgelt[2], and Michael Berthold[1]

[1] Tripos Inc., Data Analysis Research Lab,
601 Gateway Blvd., Suite 720, South San Francisco, CA 94080, USA
heiko.hofer@e-technik.tu-chemnitz.de
berthold@tripos.com

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

Abstract. The main task of drug discovery is to find novel bioactive molecules, i.e., chemical compounds that, for example, protect human cells against a virus. One way to support solving this task is to analyze a database of known and tested molecules with the aim to build a classifier that predicts whether a novel molecule will be active or inactive, so that future chemical tests can be focused on the most promising candidates. In [1] an algorithm for constructing such a classifier was proposed that uses molecular fragments to discriminate between active and inactive molecules. In this paper we present two extensions of this approach: A special treatment of rings and a method that finds fragments with wildcards based on chemical expert knowledge.

1 Introduction

The computer-aided analysis of molecular databases plays an increasingly important role in drug discovery. One of its major goals is to build classifiers that predict for a novel molecule whether it will be active or inactive, for example, w.r.t. the protection of human cells against a virus. Such a classifier can then be used to focus the expensive and time-consuming real chemical tests on the most promising candidates.

The classification model we are considering here is a set of *discriminative fragments*. Such fragments are parts of molecules that are frequent in the set of active molecules and rare in the set of inactive ones and thus discriminate between the two classes. The rationale underlying this approach is that discriminative fragments may be the key substructures that determine the activity of compounds and therefore can be used to predict the activity of molecules: If a novel molecule contains at least one discriminative fragment it is classified as active, otherwise it is classified as inactive.

Recently, several algorithms have been suggested for finding discriminative fragments efficiently. In [2] an approach was presented that finds linear fragments using a method that is based on the well-known Apriori algorithm for association rule mining [3]. However, the restriction to linear chains of atoms is limiting in many real-world applications, since substructures of interest often contain rings or branching points. The approach presented in [1] does not suffer from this restriction. It is based on a transfer of the Eclat algorithm for association rule mining [6] and thus avoids the expensive reembedding of fragments. Another approach that can find arbitrary fragments was proposed in [4]. The main differences between these approaches are the ways in which fragments of interest are generated, an issue that will become important when we study the algorithm of [1] in more detail below.

In this paper we present two extensions of the algorithm suggested in [1]. Due to a symmetry problem that makes it impossible to suppress all redundant search, this algorithm runs into problems if fragments contain a lot of rings. Our first extension solves this problem by finding rings in a preprocessing step and then treating them as units in the mining algorithm, thus making the search much more efficient. The second extension is inspired by the fact that certain atom types can sometimes be considered as equivalent from a chemical point of view. To handle such cases, we introduce wildcard atoms into the search.

As a test case for our extended algorithm we use a publicly available database of the National Cancer Institute, namely the NCI-HIV database (DTP AIDS Antiviral Screen) [5], which has also been used in [1] and [2]. This database is the result of an effort to discover new compounds capable of inhibiting the HI virus (HIV). The screen utilized a soluble formazan assay to measure protection of human CEM cells from HIV-1. Compounds able to provide at least 50% protection to CEM cells were retested. Compounds providing at least 50% protection on retest were listed as moderately active (*CM*). Compounds that reproducibly provided 100% protection were listed as confirmed active (*CA*). Compounds not meeting these criteria were listed as confirmed inactive (*CI*). The dataset consists of 41,316 compounds, of which we have used 37,171. For the remaining 4,145 no structural information was available at the time the experiments reported here were carried out. Each compound has a unique identification number (NSC number), by which we refer to it. Out of the 37,171 compounds we used, 325 belong to class *CA*, 896 to class *CM*, and the remaining 35,950 to class *CI*. We neglected the 896 moderately active compounds in our experiments. The size of this database stresses the need for a truly efficient and scalable algorithm. The method described in [1], for example, could only be applied to this data set by using small initial fragments as “seeds” for the underlying search algorithm.

2 Molecular Fragment Mining

As stated above, the goal of molecular fragment mining is to find discriminative fragments in a database of molecules, which are classified as either active or inactive. To achieve this goal, the algorithm presented in [1] represents molecules as attributed graphs and carries out a depth first search on a tree of fragments. 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

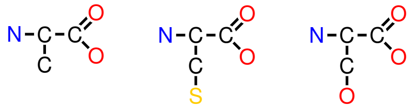


Fig. 1. The amino acids clycin, cystein and serin.

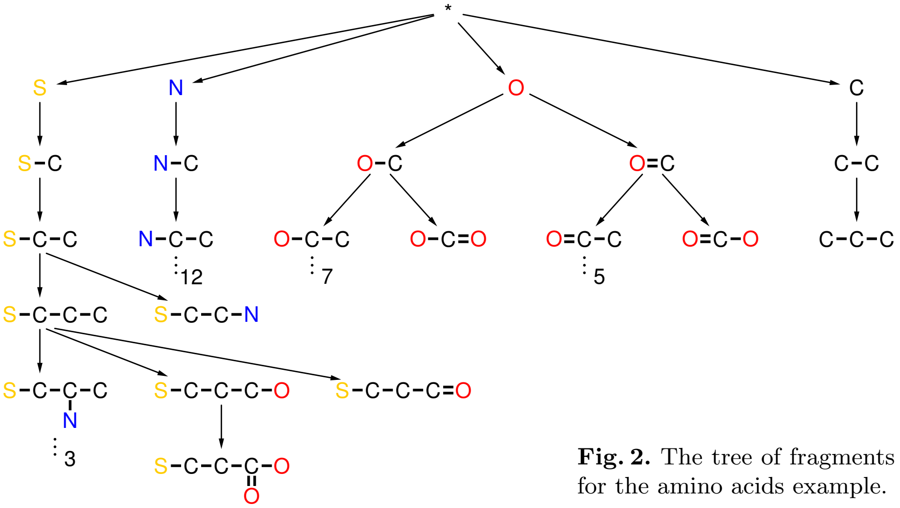


Fig. 2. The tree of fragments for the amino acids example.

the available molecules is maintained, from which the lists of embeddings of its extensions can easily be constructed. As a consequence, expensive reembeddings 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 [1] for details.)

Since the search tree we traverse in our modified algorithm differs slightly from the one used in [1], we illustrate it with 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.

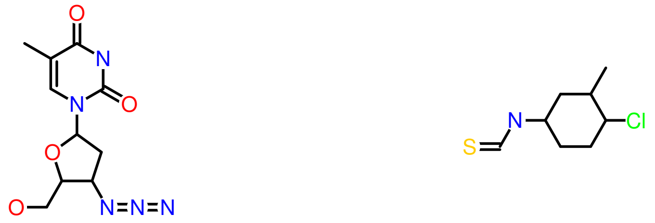
If a fragment allows for more than one extension (as is the case, for instance, for the fragments `O-C` and `S-C-C-C`), we sort them according to the local ordering rules mentioned above (see [1] for details). The main purpose of this local order is to prevent certain extensions to be generated, in order 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. 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.

Although the example molecules considered here do not contain any rings, it is clear that we have to handle them in the general case. While the original algorithm treated extensions to new atoms and extensions closing rings basically alike, only handling them through an ordering rule to avoid redundant search, our new algorithm processes them in two separate phases: Even though extensions closing rings are allowed in any step, once a ring has been closed in a fragment, the possible future extensions are restricted to closing rings as well. While this does not lead to performance improvements, it has certain advantages w.r.t. the order in which fragments are generated.

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). Furthermore, the depth of the tree may be restricted, thus limiting the size of the fragments to be generated (size based pruning).

Discriminative fragments should also be rare in the inactive molecules, defined formally by a 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



Fragment 1
freq. in CA: 15.08%
freq. in CI: 0.02%

Fragment 2
freq. in CA: 9.85%
freq. in CI: 0.00%

Fig. 3. Discriminative fragments.

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 those fragments that satisfy this threshold (in addition to the minimum support threshold in the active molecules) are reported as discriminative fragments.

In [1] 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. Among the results are, for example, the two fragments shown in Figure 3. Fragment 1 is known as AZT, a well-known nucleoside reverse transcriptase inhibitor, which was the first drug approved for anti-HIV treatment. AZT is covered by 49 of the 325 (15.08%) compounds in *CA* and by only 8 of the 35,950 (0.02%) compounds in *CI*. Fragment 2 is remarkable, because it is not covered by any compound of the 35,950 inactive compounds, but is covered by 31 molecules in *CA* (9.85%), making it a very good classification model. (Fragment 2 was not reported in [1]).

3 Rings

An unpleasant problem of the search algorithm as we presented it up to now is that the local ordering rules for suppressing redundant searches are not complete. Two extensions by identical bonds leading to identical atoms cannot be ordered based on locally available information alone. A simple example demonstrating this problem and the reasons underlying it can be found in [1]. As a consequence, a certain amount of redundant search has to be accepted in this case, because otherwise incorrect support values would be computed and it could no longer be guaranteed that all discriminative fragments are found.

Unfortunately, the situation leading to these redundant searches occurs almost always in molecules with rings, whenever the fragment extension process enters a ring. Consequently, molecules with a lot of rings, like the steroid shown in Figure 4—despite all clever search tree pruning—still lead to a prohibitively high search complexity: The original algorithm considers more than 300,000 fragments for the steroid, a large part of which are redundant.

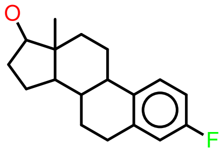


Fig. 4. Example of a steroid, which can lead to problems due to the large number of rings.

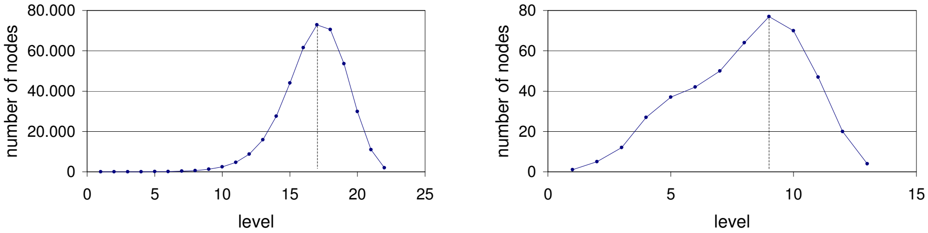


Fig. 5. Number of nodes per level for the algorithms with and without ring extensions.

To cope with this problem, we add a preprocessing step to the algorithm, in which the rings of the molecules are found and marked. One way to do this is to use the normal algorithm to find a substructure covering a given ring structure as well as a given molecule, where the search neglects the atom and bond types. In our current implementation we confined ourselves to rings with five and six atoms, because these are most interesting to chemists, but it is obvious that the approach can easily be extended to rings of other sizes.

In the search process itself, whenever an extension adds a bond to a fragment that is part of a ring, not only this bond, but the whole ring is added (all bonds and atoms of the ring are added in one step). As a consequence, the depth of the search tree is reduced (the basic algorithm needs five or six levels to construct a ring bond by bond) and many of the redundant searches of the basic algorithm are avoided, which leads to enormous speed-ups. For example, in this way the number of fragments generated for the steroid shown in Figure 4 reduces to 93.

On the NCI-HIV database our implementation of the basic algorithm (without one step ring extensions) generates 407,364 fragments in *CA* that have a support higher than 15%, for which it needs 45 minutes.[3] Using ring extensions reduces the number of fragments to 456 which are generated in 13 seconds.

A more detailed illustration of the gains resulting from one step ring extensions is provided by the diagrams shown in Figure 5. They display the number of nodes per level in the search tree for the two versions of the algorithm. As can be seen in the left diagram, the search tree of the basic algorithm has a width of approximately 80,000 fragments and a depth of 22 levels. In contrast to this, the width of the search tree using one step ring extensions is approximately 80 fragments, almost a factor of 1000 less.

[3] Our experiments were run using a Java implementation on a 1GHz Xeon Dual-Processor machine with 1GB of main memory using jre1.4.0

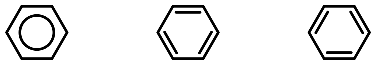
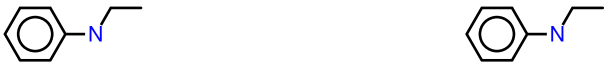


Fig. 6. Aromatic and Kekulé representations of benzene.



Fragment 3
basic algorithm
freq. in CA: 22.77%

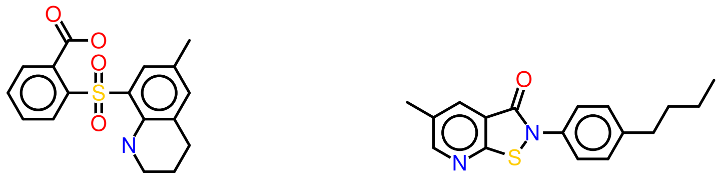
Fragment 4
with ring extensions
freq. in CA: 20.00%

Fig. 7. Frequencies in *CA* with and without ring extensions.

Furthermore, the shapes of the search trees are remarkable. While with the basic algorithm the tree width grows roughly exponentially with the depth of the tree, until a maximum is reached, with the ring extension algorithm the tree width grows only roughly linearly with the depth. As a side note, these diagrams also justify the use of a depth first search in our algorithm: The typical and extremely high width to depth ratio of the tree makes a breadth first search disadvantageous or, for the basic algorithm, infeasible.

Besides a considerable reduction of the running time, treating rings as units when extending fragments has other advantages as well. In the first place, it makes a special treatment of atoms and bonds in rings possible, which is especially important for aromatic rings. For example, benzene can be represented in different ways as shown in Figure 6. On the left, all six bonds are represented as aromatic bonds, while the other two representations—called *Kekulé structures*—use alternating single and double bonds. A chemist, however, considers all three representations as equivalent. With the basic algorithm this is not possible, because it would require treating single, double, and aromatic bonds alike, which obivously leads to undesired side effects. By treating rings as units, however, these special types of rings can be identified in the preprocessing step and then be transformed into a standard aromatic ring.

Secondly, one step ring extensions make the support values associated with a fragment more easily interpretable. To see this, consider the two fragments shown in Figure 7. The left was found with the basic algorithm, the right with the ring extension algorithm. At first glance it may be surprising that the two fragments look alike, but are associated with different support values (74 compounds, or 22.77%, for the basic algorithm and 65 compounds, or 20.00%, for the ring extension algorithm). However, the reason for this is very simple. While with the basic algorithm the N-C-C chain extending from the aromatic ring may be a real chain or may be a part of another ring, it must be a real chain with the ring extension algorithm. The reason is that with the latter it is not possible to step into a ring bond by bond, because rings are always added in one step. Therefore the ring extension algorithm does not find this fragment, for instance, in the two compounds shown in Figure 8. Although it may appear as a disadvantage at first sight, this behavior is actually highly desirable by chemists and biologists,



NSC #667948 NSC #698601

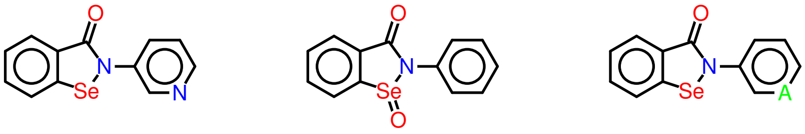
Fig. 8. Compounds that contain Fragment 3 but not Fragment 4.

because atoms in rings and atoms in chains are considered to be different and thus should not be matched by the same fragment. A fragment should either contain a ring or not, but not parts of it, especially if that part can be confused with a chain.

4 Fragments with Wildcards

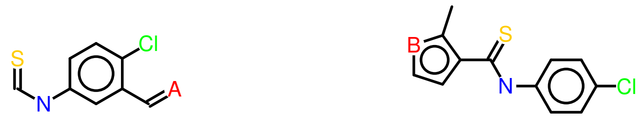
Chemists and biologists often regard fragments as equivalent, even though they are not identical. Acceptable differences are, for instance, that an aromatic ring consists only of carbon atoms in one molecule, while in another ring one carbon atom is replaced by a nitrogen atom. An example of such a situation from the NCI-HIV database is shown in Figure 9. A chemist would say that the two molecules shown on the left have basically the same structure and that the difference consists only in the oxygen atom attached to the selenium atom. However, the original algorithm will not find an appropriate fragment because of this mismatch in atom-type [1].

More generally, under certain conditions certain types of atoms can be considered as equivalent from a chemical point of view. To model this equivalence we introduce wildcard atoms into the search, which can match atoms from a user-specified range of atom types. As input we need chemical expert knowledge to define equivalence classes of atoms, possibly conditioned on whether the atom is part of a ring or not (that is, certain atom types may be considered as equivalent in a ring, but as different in a chain). In the example shown in Figure 9 we may introduce a wildcard atom into the search that can match carbon as well



NSC #639767 NSC #639772 wildcard fragment

Fig. 9. Matching molecules with a fragment with a wildcard atom.



Wildcard fragment 1

A:	O	N
freq. in CA:	5.5%	3.7%
freq. in CI:	0.0%	0.0%

Wildcard fragment 2

B:	O	S
freq. in CA:	5.5%	0.01%
freq. in CI:	0.0%	0.00%

Fig. 10. Fragments with wildcards in the NCI-HIV database.

as nitrogen atoms in rings. This enables us to find the fragment on the right, where the green A denotes the wildcard atom.

Formally, wildcard atoms can be introduced into the search process by adding to each node a branch for each group of equivalent atoms, provided that there are at least two molecules that contain different atoms from the group at the position of the wildcard atom. In the later search process, this branch has to be treated in a special way, because it can be pruned if—due to further extensions—the molecules matching the fragment all contain the same atom for the wildcard atom. In our implementation we restrict the number of wildcard atoms in a fragment to a user-defined maximum number.

We applied the wildcard atom approach to the NCI-HIV database, restricting the number of wildcard atoms to one. With this approach the two wildcard fragments shown in Figure 10 are found (among others). The left fragment has a wildcard atom indicated by a red *A* on the right, which may be replaced by either an oxygen or a nitrogen atom. The (non-wildcard) fragment that results if A is replaced with an oxygen atom has a support of 5.5% in the active compounds, whereas it does not occur in the inactive ones. The fragment with A replaced with nitrogen has a support of 3.7% in the active compounds, and also does not occur in the inactive ones. Consequently, the wildcard fragment has a support of 9.2% in the active compounds. This demonstrates one of the advantages of wildcard fragments, that is, they can be found with much higher minimum support thresholds, which can cut down the output of irrelevant fragments.

However, wildcard fragments can also make it possible to find certain interesting fragments in the first place. This is demonstrated with the wildcard fragment shown in the right in Figure 10. It contains a wildcard atom denoted by a red B in the five-membered aromatic ring on the left, which may be replaced by either an oxygen or a sulfur atom. The interesting point is that the (non-wildcard) fragment that results from replacing B with sulfur has a very low support of 0.01% in the active molecules. This (non-wildcard) fragment would not be found without wildcard atoms, because it is not possible to run the search with a sufficiently low minimum support threshold.

5 Conclusions

This paper adressed two limitations of the molecular fragment mining algorithm introduced in [1]. The first extension of the original algorithm deals with the frequent occurrence of rings and their effect on the width of the generated search tree. By preprocessing the molecular database we can identify rings before the actual search starts and then treat them as one entity during the search. This not only speeds up the computation tremendously, but also finds chemically more meaningful fragments since chemists tend to think of rings as one molecular unit. The second extension addresses the need to find fragments that behave chemically similar but are different from a graph point of view. We have proposed a mechanism that allows the user to specify equivalence classes of atoms. Fragments can then contain a certain maximum number of such *wildcard* atoms. Using this method we not only find chemically meaningful fragments, but we can also detect fragments that would otherwise fall below the support threshold.

The two extensions presented here make the original algorithm described in [1] directly applicable to much larger databases and allow the chemist to introduce expert knowledge about similarities on an atomic level, as demonstrated on the NCI-HIV dataset. As a result, the method presented here has been used very successfully, among others, at Tripos Receptor Research (Bude, UK) for the prediction of synthetic success.

References

1. 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
2. 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
3. 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
4. M. Kuramochi and G. Karypis. An Efficient Algorithm for Discovering Frequent Subgraphs. Technical Report TR 02-026, Dept. of Computer Science/Army HPC Research Center, University of Minnesota, Minneapolis, USA 2002
5. O. Weislow, R. Kiser, D. Fine, J. Bader, R. Shoemaker, and M. Boyd. New Soluble Formazan Assay for HIV-1 Cytopathic Effects: Application to High Flux Screening of Synthetic and Natural Products for AIDS Antiviral Activity. *Journal of the National Cancer Institute*, 81:577–586. Oxford University Press, Oxford, United Kingdom 1989
6. 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)*, 283–296. AAAI Press, Menlo Park, CA, USA 1997