Combining Ring Extensions and Canonical Form Pruning

Christian Borgelt

European Center for Soft Computing
c/ Gonzalo Gutiérrez Quirós s/n, 33600 Mieres, Spain
`christian.borgelt@softcomputing.es`

Abstract. A common problem in frequent graph mining is the size of the output, which can easily exceed the size of the database to analyze. In the application area of molecular fragment mining a promising approach to tackle this problem is to treat certain substructures as a unit. Among such structures, rings are most prominent, and by requiring that either a ring is present as a whole in a fragment, or not at all, the size of the output can be reduced considerably. In this paper I present two ways to combine such ring mining with canonical form pruning.

1 Introduction

In recent years frequent graph mining has become an area of very active research. Given a database of (attributed) graphs, one tries to find all subgraphs that appear with a user-specified minimum frequency. Some algorithms for this task rely on principles from inductive logic programming [5]. However, the vast majority transfers techniques from frequent item set mining. Examples are MolFea [9], FSG [10], MoSS/MoFa [1], gSpan [12], CloseGraph [13], FFSM [7], and Gaston [11]. A related, but slightly different approach is used in Subdue [4].

The basic idea is to grow subgraphs into the graphs of the database, adding an edge and maybe a node in each step, counting the number of graphs containing each grown subgraph, and eliminating infrequent subgraphs. Unfortunately, with this method the same subgraph can be constructed in several ways, adding its nodes and edges in different orders. The predominant method to avoid the ensuing redundant search is to define a canonical form of a graph that uniquely identifies it up to automorphisms: together with a specific way of growing the subgraphs it enables us to determine whether a given subgraph can be pruned from the search tree (see, for example, [3] for a family of such canonical forms).

Another common problem in frequent graph mining is the size of the output, which can easily exceed the size of the database to analyze. To tackle this problem, it can be useful to restrict the output to subgraphs with certain meaningful properties. In the application area of molecular fragment mining, for example, treating rings as units—that is, requiring that either a ring is present as a whole, or not at all—not only reduces the size of the output, but also improves its interpretability and speeds up the mining process considerably [6].

However, [6] employed a repository of reported fragments in order to suppress duplicate fragments. The canonical form pruning approach is clearly preferable, in particular, because it makes the method substantially simpler to parallelize. In this paper I present two ways in which ring mining—that is, using extensions by full rings instead of extensions by individual ring edges—can be combined with canonical form pruning: a filter approach and a reordering approach.

2 A Filter Approach to Ring Mining

Ring Preprocessing. Ring mining requires preprocessing the rings in the graphs (here usually: molecules) to analyze: given a user-specified range of ring sizes, all rings within this size range are marked in the graphs [6]. Technically, there are two parts: a marker in the edge attribute, fundamentally distinguishing ring edges from non-ring edges, and a set of flags identifying the different rings an edge is contained in. (Note that an edge can be part of several rings.)

Filtering Open Rings. If we require the output to have only complete rings, we have to identify and remove fragments with ring edges that do not belong to any complete ring. Since ring edges have been marked in the preprocessing, we know which edges are ring edges. Using the same procedure as for the preprocessing, we mark rings in the fragment, but only set the ring flags, while the edge type marker is kept. Afterwards we can find edges that belong to incomplete rings by simply checking whether a ring edge (as identified by its type marker) did not receive any ring flags. If there exists such an edge, the fragment is discarded.

Filtering Unclosable Rings. Canonical form pruning allows to restrict the possible extensions of a fragment [3]. For the canonical forms of gSpan [12] and MoSS/MoFa [1] these are so-called *rightmost extensions* and *maximum source extensions*, respectively. Their effect is that due to previous extensions certain nodes in the graph become unextendable, that is, no new edge may be attached to them. This can easily be exploited to prune the search. Obviously, a necessary (though not sufficient) condition for all rings being closed is that every node has either zero or at least two incident ring edges. If there is a node with only one incident ring edge, this edge must be part of an incomplete ring. Hence we can filter fragments in the search by checking whether any non-extendable node has only one incident ring edge. If this is the case, the fragment can be discarded.

Merging Ring Extensions. The two filtering methods described above work on individual edges and hence they cannot always detect if an extension by a ring edge only leads to fragments with complete rings that are infrequent. Consider, for example, a database with two molecules: one is a ring with 5 carbon atoms, the other a ring with 6 carbon atoms (all bonds are single). If we want to find fragments that are part of both molecules, we need not consider any extension by a ring bond, since there is no common fragment with a complete ring. However, any single ring bond is contained in both fragments and thus it is, in itself, a frequent fragment. Therefore, by working on individual bonds, we only recognize after constructing the full 5-ring that there is no common fragment.

To avoid such redundant search, we add all edges of a ring that an extension edge is contained in, thus distinguishing extensions that start with the same single edge, but lead into rings of different size or different composition. Then we determine the support and prune infrequent extensions, and finally we trim and merge ring extensions that share the same initial edge. Note that the resulting situation differs considerably from direct extensions by individual edges. In the first place, all extensions by ring edges, which are frequent as ring edges, but become infrequent when completed into rings (and thus cannot produce any output), have been removed. In addition, for the remaining edges all embeddings (that is, occurrences of the fragment in the graph database), which lead to infrequent fragments once rings are completed, have been eliminated.

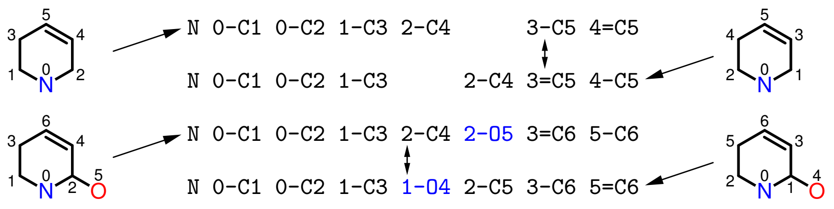
3 A Reordering Approach to Ring Mining

Even with merging ring extensions the search is still based on an edge-by-edge scheme and thus fragments grow fairly slowly. It would be better if we could add complete rings in one step, thus adding several edges to the fragment. Unfortunately such an approach interferes with canonical form pruning, as is discussed below. However, the problems can be solved, although up to now I only achieved a solution for the breadth-first search canonical form of MoSS/MoFa [3].

Ring Preprocessing. Although for the filtering approach it is sufficient to mark rings in the user-specified size range, the reordering approach also needs *pseudo-rings*, as I call them, to be marked. These are rings of smaller size than the user specified, which consist only of edges that are part of rings within the user-specified size range. As an example consider the molecule shown on the left of Figure 3: if only rings with 5 and 6 bonds are marked, this molecule contains a pseudo-ring of size 3 comprising the atoms labeled 1, 3, and 4.

Ring Extensions and Canonical Forms. If we add complete rings in one step and want to use canonical form pruning, we have to order the new edges in such a way that the result is (as far as possible) in canonical form. This can easily be achieved with a recursive procedure similar to the canonical form test. However, this can have the effect that—due to asymmetries in the ring—we commit to a numbering of the nodes that prevents us from finding certain fragments. The reason is that in order to be in canonical form an (asymmetric) ring may need its nodes to be numbered differently depending on whether it is considered purely (i.e. only ring edges are present) or whether it has branches attached to it.

As an example consider the molecules shown in Figure 1 and assume the orderings `N` $\prec$ `O` $\prec$ `C` and `=` $\prec$ `-`. The nodes of the pure ring can be numbered in two ways in a breadth-first manner starting at the nitrogen atom (as shown in the top part of Figure 1). Of course, they lead to different code words, which are shown in the middle. Since single bonds precede double bonds, the upper code word is smaller and thus the left fragment is in canonical form. If, however, we consider the ring with an oxygen atom attached (as shown in the lower part of Figure 1), the alternative node numbering yields the canonical form. The reason



```
N 0-C1 0-C2 1-C3 2-C4        3-C5 4=C5
                               ↕
N 0-C1 0-C2 1-C3        2-C4 3=C5 4-C5

N 0-C1 0-C2 1-C3 2-C4 2-O5 3=C6 5-C6
                   ↕
N 0-C1 0-C2 1-C3 1-O4 2-C5 3-C6 5=C6
```

Fig. 1. Attaching a branch to a ring can change the ordering for the canonical form.

is that the attached bond to the oxygen atom has to be inserted into rather than appended to the code word. As a consequence we cannot simply commit to the numbering of the nodes as it is prescribed by the canonical form of a ring, since this numbering may change when branches or other rings are added.

A second, even nastier problem is posed by connected or even nested rings: if two ring extensions follow each other, their edges may have to be "spliced" to construct a proper code word. In doing so, one has to take care that edges can be reordered in a sufficiently flexible way, so that no fragments get lost due to the canonical form pruning. On the other hand, one also has to make sure that no duplicates result, which cannot be detected with a canonical form test. This is fairly difficult to achieve, in particular if the fragment contains branching points of equivalent ring edges, that is, ring edges that all start at the same node and have the same edge attribute and the same destination node attribute.

Canonical Form Pruning for Ring Extensions. A strategy to overcome the above problems consists in keeping (and thus also extending) fragments that are not in canonical form, but that could become canonical once branches are added. Which non-canonical fragments to keep is determined as follows: adding all edges of a ring can be seen as adding its initial edge as in an edge-by-edge procedure, and some additional edges, the positions of which are not yet fixed. Hence we split the code word into two parts: a fixed part, which would also be built by an edge-by-edge procedure, and a volatile part comprising the additional bonds. The fixed part is the prefix of the code word up to (and including) the last edge added in an edge-by-edge manner, and the volatile part is the suffix of the code word after this edge. If the current code word deviates from the canonical code word in the fixed part, the fragment is pruned, otherwise it is kept.

As an example consider the search tree shown in Figure 2. The search starts at the nitrogen atom and constructs the ring with both possible numberings of the nodes. The form in the left branch is canonic (indicated by a solid box), so it is kept. In the right branch only the first ring bond (from the nitrogen atom to the right carbon atom) is fixed, every other bond is volatile. Since the code word for this fragment deviates from the canonical one only at the 5th bond (see Figure 1), we may not discard it. However, we mark it as non-canonical (indicated by a dashed box), so that it is not reported.

On the next level, adding the bond to the oxygen atom in the two possible places yields, for each branch, one canonical (solid box) and one non-canonical

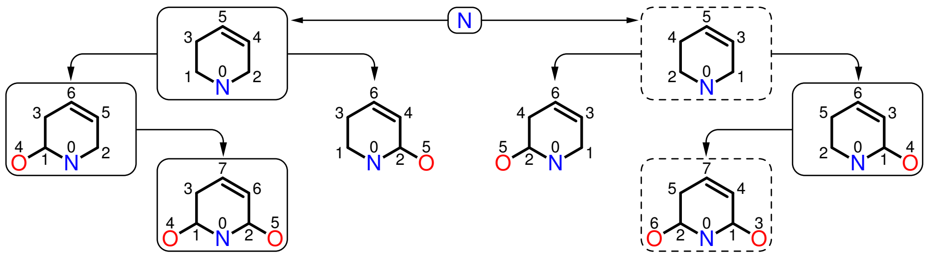


Fig. 2. Search tree for an almost symmetric ring with two identical branches.

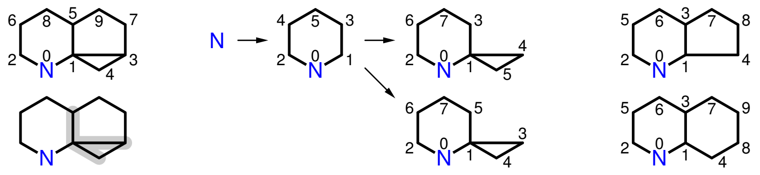


Fig. 3. Splicing equivalent bonds of ring extensions to achieve canonical form.

fragment (no box). The non-canonical fragments both differ in the fixed part, which now consists of the first three bonds[1], and hence they are both pruned. Extending the canonical fragments adds the other bond to an oxygen atom, thus making the fragment symmetric again (with the exception of the double bond). Hence the left fragment is in canonical form. The fragment in the right branch, however, is not canonical. Nevertheless it has to be kept: the first four bonds are fixed, but it deviates from the canonical code word only in the 7th bond.

In order to handle connected and nested rings, we have to consider how the edges of a ring extension may be "spliced" with edges already in the fragment. The problem here are branching points, where several equivalent edges start (i.e. edges with the same attribute and same attribute of the destination node). As an example consider the molecule on the left of Figure 3, the relevant equivalent edges of which are highlighted. Part of an example search for this fragment (shown in the middle) starts from the nitrogen atom, adds a 6-bond ring and then a 3-bond ring. If we only allowed, for example, shifts of new bonds to the left up to equivalent bonds, we can only reach the upper form. However, the canonical form is the lower one. On the other hand, we may not reorder equivalent edges freely, as this would interfere with keeping certain non-canonical fragments: the fragments in the first level of the search tree in Figure 2 differ only in the order of the two equivalent bonds starting at the nitrogen atom. If adding another ring to this atom could change this order, duplicate fragments could result.

[1] Note how adding the bond to the oxygen atom turned one of the volatile ring bonds into a fixed one: everything preceding the bond characterizing an extension—whether it is a single non-ring bond or the first bond of a ring extension—becomes fixed.

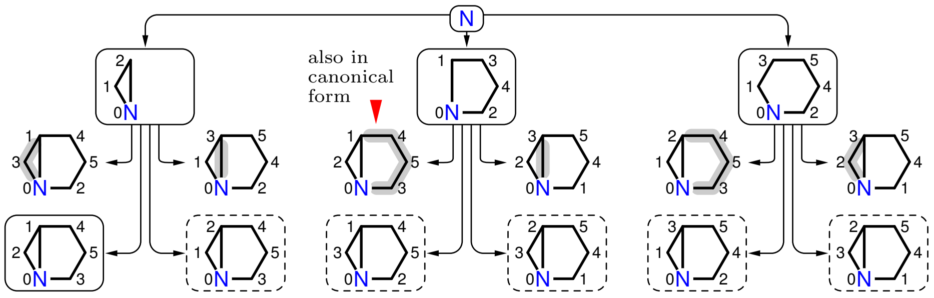


Fig. 4. Search tree for nested rings of different size (with ring order pruning).

As a consequence, the splicing rule for equivalent edges must be as follows: the order of the equivalent edges already in the fragment must be maintained, and the order of the equivalent new edges must be maintained (to avoid "flipping" rings). In addition, no new edge must be shifted into the fixed part. However, any sequence satisfying these conditions is acceptable and must be considered. In other words: the two sequences of equivalent edges—one from the existing fragment and one from the added ring—may be merged in a zipper-like manner, selecting the next edge from either list, but maintaining the order in each list.

This also explains why we need pseudo-rings: without them it would be impossible in some cases to achieve canonical form. If, in the example, we could only add the 5-ring and the 6-ring, but not the 3-ring (see the right of Figure 3), the upward bond from the atom numbered 1 would always precede at least one of the other two bonds that are equivalent to it. However, in the canonical form (shown at the top left of Figure 3) the upward bond succeeds both other bonds.

Unfortunately, the same fragment can now be reached in the same form in different ways. To see this, consider the search tree shown in Figure 4: the same fragment results in six different forms, each of which (including the canonical form) occurs twice. Hence we need an additional test to augment canonical form pruning. The idea underlying this test is that the canonical form not only fixes an order of the bonds, but also an order of the rings: order the rings by the first edge in the canonical form that is contained in them. Break ties by considering the second, third etc. edges in the canonical form the rings contain.

This ring order is exploited as follows: we mark all edges in the fixed part of the fragment as well as all new edges. Then we traverse the unmarked ring edges. For each of these edges we eliminate all rings it is contained in (by removing ring flags, which are set in the fragment in the same way as in the ring preprocessing). If by this procedure all ring flags of a marked edge (fixed edge or new edge) are removed, this marked edge is part of a ring preceding the newly added one (w.r.t. the ring order defined above). Therefore it had to be added before the current ring. If, however, no marked edge loses all its ring flags, there exists a ring that succeeds the currently added one. This ring should rather be added later and thus the fragment is discarded, even if it is in canonical form.

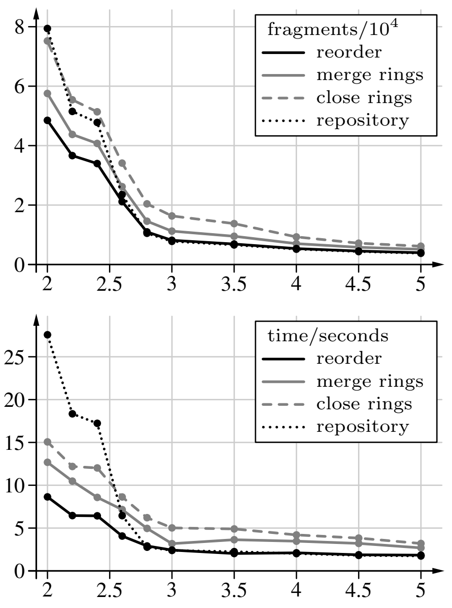


Fig. 5. Experimental results on the IC93 data. The horizontal axis shows the minimal support in percent. The curves show the number of generated fragments (top left), the number of generated embeddings (top right), and the execution time in seconds (bottom left) for the four different strategies. Note that the dotted line (repository) is often covered by the solid line.

As an example consider again Figure 4. The canonical form (fragment at bottom left) shows that the 6-bond ring succeeds both the 3-bond ring and the 5-bond ring, as it does not contain the first bond. The second bond shows that the 3-bond ring precedes the 5-bond ring. For the second fragment in canonical form (see arrow) the test, as outlined above, fails, using any of the highlighted edges. However, for the leftmost fragment in the bottom row the ring flag removal always leaves a marked edge flagless and thus it is accepted.

4 Experiments

I incorporated the described methods into the MoSS program.[2] As a test dataset I used the well-known subset of the *Index Chemicus* 1993 [8]. The results are shown in Figure 5: each curve corresponds to one of the approaches, which were executed with (full) perfect extension pruning [2]. The black dotted line refers to the repository-based approach, the solid black line to the reordering approach. The filter approach I used in two flavors: with ring extensions and consecutive merging (solid grey line) and using only open and unclosable ring filtering (dashed grey line). These results show that the somewhat complicated treatment of ring extensions in order to combine them with canonical form pruning clearly pays off, as the reordering method hugely outperforms the filter approaches.

[2] MoSS is available for download under the Gnu Lesser (Library) Public License at `http://fuzzy.cs.uni-magdeburg.de/∼borgelt/moss.html`.
The test system was a Lenovo Thinkpad X60s (Intel Core Duo@1.67GHz, 1GB) with S.u.S.E. Linux 10.0, IBM's Jikes compiler 1.22, and Sun's Java 2 1.5.0_07.

5 Conclusions

In this paper I developed two methods to combine canonical form pruning with ring extensions, which are very useful for molecular fragment mining. The first method is based on a filter approach, which exploits the requirement for closed rings to restrict the output and prune the search. This approach is simple and straightforward, but not competitive in terms of running time or memory usage as the experiments show. Therefore it is indeed worthwhile to invest into the more complicated reordering method, which enables us to do full ring extensions, thus considerably reducing the height of the search tree and its number of nodes.

References

1. C. Borgelt and M.R. Berthold. Mining Molecular Fragments: Finding Relevant Substructures of Molecules. *Proc. IEEE Int. Conf. on Data Mining*, 51–58. IEEE Press, Piscataway, NJ, USA 2002
2. C. Borgelt, T. Meinl, and M.R. Berthold. Advanced Pruning Strategies to Speed Up Mining Closed Molecular Fragments. *Proc. IEEE Conf. on Systems, Man and Cybernetics*, CD-ROM. IEEE Press, Piscataway, NJ, USA 2004
3. C. Borgelt. On Canonical Forms for Frequent Graph Mining. *Proc. 3rd Int. Workshop on Mining Graphs, Trees and Sequences*, 1–12. ECML/PKDD 2005 Organization Committee, Porto, Portugal 2005
4. D.J. Cook and L.B. Holder. Graph-Based Data Mining. IEEE Trans. on Intelligent Systems 15(2):32–41. IEEE Press, Piscataway, NJ, USA 2000
5. 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
6. H. Hofer, C. Borgelt, and M.R. Berthold. Large Scale Mining of Molecular Fragments with Wildcards. *Intelligent Data Analysis* 8:495–504. IOS Press, Amsterdam, Netherlands 2004
7. 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*, 549–552. IEEE Press, Piscataway, NJ, USA 2003
8. *Index Chemicus — Subset from 1993*. Institute of Scientific Information, Inc. (ISI). Thomson Scientific, Philadelphia, PA, USA 1993
   `http://www.thomsonscientific.com/products/indexchemicus/`
9. 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*, 136–143. ACM Press, New York, NY, USA 2001
10. M. Kuramochi and G. Karypis. Frequent Subgraph Discovery. *Proc. 1st IEEE Int. Conf. on Data Mining*, 313–320. IEEE Press, Piscataway, NJ, USA 2001
11. S. Nijssen and J.N. Kok. A Quickstart in Frequent Structure Mining Can Make a Difference. *Proc. 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 647–652. ACM Press, New York, NY, USA 2004
12. X. Yan and J. Han. gSpan: Graph-Based Substructure Pattern Mining. *Proc. 2nd IEEE Int. Conf. on Data Mining*, 721–724. IEEE Press, Piscataway, NJ, USA 2002
13. X. Yan and J. Han. Closegraph: Mining Closed Frequent Graph Patterns. *Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 286–295. ACM Press, New York, NY, USA 2003