First publ. in: 2004 IEEE International Conference on Systems, Man & Cybernetics, The Hague, Netherlands, 10 - 13 October 2004. Piscataway, NJ : IEEE Operations Center, 2004, pp. 4559-4564

Hybrid fragment mining with *MoFa* and FSG*

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

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

Abstract – *In the last few years a number of different subgraph mining algorithms have been proposed. They are often used for finding frequent fragments in molecular databases. All these algorithms behave quite well when used on small datasets of not more than a few thousand molecules. However, they all fail on larger amounts of data because they are either time consuming or have enormous memory requirements. In this paper we present a hybrid mining technique that overcomes the individual problems of the underlying algorithms and outperforms the individual methods impressively on large databases.*

Keywords: Molecules, fragments, graph mining, hybrid algorithm.

1 Introduction

A frequent problem in biochemistry concerns the discovery of common parts of molecules. This is e.g. the case in drug discovery where the biochemists try to predict the behaviour of new agents even before synthesizing them. They use large libraries with often tens to hundreds of thousands of candidates. Another example are so-called high throughout screens (HTS), the output of which are activity information of hundreds of thousands molecules. Using graph based data-mining techniques one can now try to find frequent discriminative fragments – i.e. fragments that occur frequently in one group of the molecules and which are infrequent in the others – which (hopefully) explain the reactions the compounds have shown (or have not shown).

There are two classes of approaches that are able to solve this type of problem. The first class is formed by algorithms that rely 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) that is compatible with both negative and positive examples [3]. The second and bigger class of algorithms represents molecules as graphs and then searches for frequent subgraphs in the molecule database.

All known graph based data mining algorithms rely on one of the two well-known frequent item-set mining algorithms, Apriori [1] or Eclat [11]. Examples are MoFa [2], FSG [6], gSpan [10], FFSM [5] and Gaston [9]. They all can be classified using three criteria:

- search strategy (depth-first, breadth-first, beam search)
- generation of new candidate fragments (extension and/or join of smaller fragments)
- support computation (subgraph isomorphism tests, use of embeddings lists).

In the following we consider two of these algorithms, *MoFa* and FSG because they are two well-known representatives of the two different search strategies. *MoFa* performs a depth-first search and generates new fragments by extending smaller ones while keeping embedding lists to speed up the support computation[1]. Additionally it uses some sophisticated pruning strategies. One big disadvantage of this approach is the potentially huge memory consumption because of the embeddings lists that can become extremely long (especially for small fragments). This makes it nearly impossible to find all – especially small – frequent discriminative fragments on larger databases of about 20,000 molecules and more. However, if carbon-only fragments are omitted – which are not very interesting for biochemists anyway – the boundary is much higher. It is then no problem to search on databases of 50,000 molecules with 1GB of RAM.

FSG on the other hand searches breadth-first, creates new fragments by joining smaller structures that share a common core and checks for subgraph isomorphism in the database for every new candidate. In the early stages of the search tree, when the molecules are small, FSG is fast and has only moderate memory demands. However, the bigger the fragments grow, the slower the search gets. Additionally, because of the breadth-first search and exponential growth of the the number of fragments towards the middle of the search tree, the memory consumption rises dramatically.

Since the problems of these two approaches are quite complementary, this paper presents a hybrid approach that combines these two algorithms. In section 2 we describe this hybrid strategy in more detail, in section 3 we present some experimental results and section 4 summarizes our work.

[1] An embedding of a subgraph into its supergraph is a "stored" subgraph isomorphism. The mapped nodes and edges are recorded so that extensions of the fragment can be tested very quickly for subgraph isomorphism.

2 The Hybrid approach

The idea behind the hybrid approach is to combine the two algorithms in an adaptive way. The main problem of *MoFa* is the vast amount of memory needed in the early stages of the search. Even in medium sized databases of about 20,000 structures some small fragments occur several hundred thousand times in the molecules. This leads to a memory consumption of far more than one gigabyte. But once the fragments grow bigger (typically six to seven bonds) their number of embeddings drops drastically. Figure 1 shows the number of embeddings of the most frequent fragment in the complete NCI-HIV dataset[8] in relation with its size. The HIV dataset consists of about 35,000 molecules tested against HIV. Depending on the reaction they have shown they are grouped into the three classes CA (confirmed active), CM (confirmed medium active) and CI (confirmed inactive).

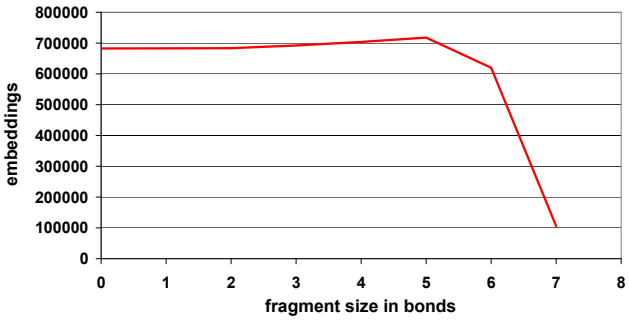


Figure 1: The number of embeddings depends heavily on the size of the fragments. This example is taken from experiments on the HIV dataset.

The reason for the drastic drop between six and seven bonds are the symmetries of the fragments. Molecular databases typically contain a large number of rings with five or six atoms. Having a subgraph that only consists of such a ring, one can find five (or six) embeddings of this ring into the molecule at one single place because of the symmetry. And often there is even more than just one ring per molecule. But as soon as in subsequent steps of the search an additional bond is added to the ring, almost all symmetries break and thus the number of subgraph isomorphisms decreases dramatically.

This effect is one of the main motivations for the hybrid approach. If *MoFa* was somehow able to pass this barrier it would be no problem to mine even very large databases. This is the point where FSG comes in. In contrast to *MoFa*, FSG does not store embeddings. Instead it tests each new fragment candidate for subgraph isomorphism (SGI) against all graphs. Therefore, it needs less memory than *MoFa* does. But as the fragments grow bigger the SGI tests become more and more expensive (SGI is proven to be in NP, see [4] for details). The central idea of our hybrid approach is to start the search process using FSG and switch to *MoFa* once the discovered fragments satisfy a suitable criterion. This is done by using each fragment from FSG as a *seed* for *MoFa*. Seeds are starting points from which *MoFa* generates new fragments by extending the seed fragment, see [2] for details.

Although the idea of a combined approach is apparent, two questions remain: What is a suitable boundary criterion and how well does *MoFa* work if a potentially large number of seeds needs to be expanded during its search? The next two sections present some answers.

2.1 A suitable boundary criterion

A naive idea is to use a fixed fragment size as a switch criterion between FSG and *MoFa*. Having figure 1 in mind this will seem to be quite natural: The number of embeddings and thus the amount of memory needed to store them drops significantly at a fragment size of 7. So the approach would be to start with FSG and let it mine until level 7 of the search tree. Then each discovered frequent fragment is passed as a seed to *MoFa*. Although this is quite easy to implement it has a huge drawback. The main problem for *MoFa* is the number of embeddings but the size of a fragment is a very weak indicator for the number of its subgraph isomorphisms. Of course this does not become clear if one only looks at the maximum number of embeddings of the most frequent fragment like in figure 1. If one takes chlorine and carbon for example, their frequencies in the HIV dataset are 8150 and over 680,000, respectively! On one hand there would be no need to let fragments containing chlorine grow to a size of seven bonds before switching over to *MoFa*. On the other hand it is hopeless to take a single carbon atom as seed, because the embedding lists will be too large.

The problem of this approach becomes even more obvious when considering the number of seeds *MoFa* will receive as starting point. On the NCI-HIV data with a minimum support of 5% in classes CA+CM FSG finds 1516 fragments of size seven. All these will be passed to *MoFa* and the result will be that a large amount of the fragment lattice is traversed multiple times. That is because seeds are freely extendable at all atoms in *MoFa*.

Table 1 shows the results of using this border criterion on the HIV dataset. The seeds have to be grown to seven bonds in FSG otherwise *MoFa* will need more than the 750MB we provided. But then it will take nearly three hours to find all frequent fragments which is magnitudes slower than the 5 minutes *MoFa* needs under the same circumstances if carbon-only fragments are left out.

Obviously we need a better indicator for the switch from FSG to *MoFa*. The number of subgraph isomorphisms a fragment has is a much better criterion as this directly addresses the problem of the number of embeddings. Using this criterion, called the *embedding threshold*, has two implications: First, fragments of different sizes are fed into *MoFa* (remember that e.g. chlorine can be used directly without extending it). Second, in FSG not only the support of each candidate has to be determined but *all* occurrences in all molecules have to be found. This is quite time consuming. However, it is usually sufficient to check against a random sample of only about 1% of all molecules. This already gives

Table 1: The number of seeds and the corresponding runtime on the HIV dataset of the FSG-*MoFa*-combination.

seed size	# seeds	runtime
0	8	*out of memory*
1	27	*out of memory*
2	63	*out of memory*
3	153	*out of memory*
4	307	*out of memory*
5	554	*out of memory*
6	929	*out of memory*
7	1,516	2:45 h
8	2,360	3:51 h
9	3,649	3:58 h
10	5,425	8:14 h

a very accurate measure for the total number of embeddings that a fragment would have. In our experiments, the prediction of this simple heuristic was only off by about 3% from the real value, which is more than sufficient for the intended purpose.

At the end of every search tree level each frequent fragment generated by FSG is evaluated using this measure. If the predicted number of embeddings is below the specified threshold the fragment is marked non-extendable and FSG prevents joins with siblings that share a common core while generating the next level. Instead this fragment is taken as a seed for *MoFa*. Subgraphs whose estimated number of embeddings lie above the threshold are joined further in FSG. Figure 2 shows part of the generated search tree. The circled structures satisfy the embedding threshold criterion and are fed into *MoFa*. The others are grown further.

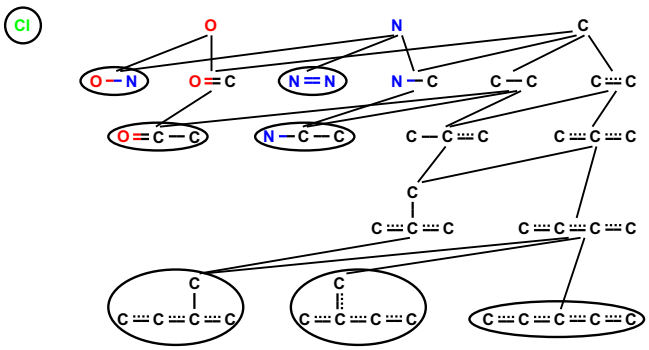


Figure 2: Creating seeds for *MoFa* with FSG

This approach heavily reduces the number of seeds that *MoFa* has to extend, as we will discuss in more detail later. Surprisingly, the number of seeds does not decrease monotonically as one would expect. Consider the partial search tree in figure 3 for an explanation. For each of the fragments the number of embeddings is shown on the right. Let us assume two different runs with embedding thresholds of 100,000 and 125,000 embeddings. For the smaller case only the circled fragment at the bottom fulfills the criterion and is fed into *MoFa*. The larger threshold causes the dashed-circled structure to be used as a second seed. And even though FSG does not join it with any other sibling in the next step, the circled fragment can also be generated making a self-join of its other parent on the left. Thus it is used a seed (which is not necessary at all), too, which results in one more seed than with the lower threshold. With an even higher threshold, the parent of both fragments already satisfies the criterion and the number of seeds would drop again.

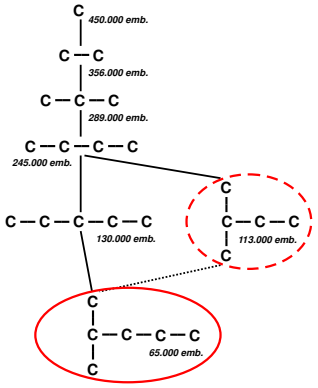


Figure 3: Even though the embedding threshold is higher more seeds are generated

In order to get correct results one has to keep in mind, that fragments generated by FSG may already satisfy the reporting criteria. Thus not only structures found by *MoFa* have to be reported. Detailed results on the performance of this technique in our hybrid approach are given in section 3.

2.2 Pruning revisited

One of *MoFa*'s important pruning mechanisms is the so-called *seed based pruning*. If the search is started with an empty seed, *MoFa* automatically chooses all frequent atoms as seeds. Once the search has been completed using one seed, in the next branch with a new atom as root, no extensions are made using the previous seed atoms. That is because all fragments containing this atom have already been discovered. In our new hybrid approach *MoFa* is seeded with bigger structures. Thus this type of search tree pruning cannot be used, slowing down the search process considerably. This effect is shown in figure 4 on the example of the complete search tree created by the hybrid algorithm on the cyclin molecule (which is shown on the bottom). Let us assume that the three circled fragments are created by FSG and are then handed over to *MoFa*. In this example the switch criterion does not depend on the number of embeddings, but is chosen arbitrarily in order to simplify the example. The dashed-circled carbon-fragment at the right is also found by FSG but not fed into *MoFa* because we assume that is has too many embeddings. Also it has no child fragment(s) so it is directly reported by FSG. The lack of seed based pruning results in many fragments having more than one parent which means that they are discovered more than once (which is undesirable of course)[2].

[2]The dashed line on the left side of the figure is only traversed if the fragment `N-C-C-O` is grown from `O-C-C`. On the path from `N-C-C` the

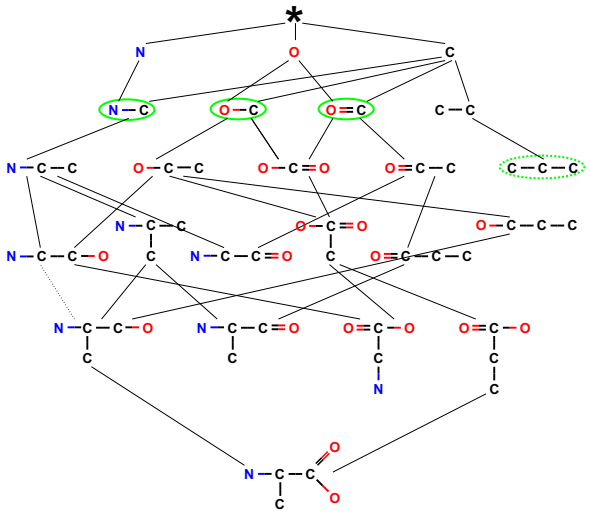


Figure 4: The search tree of the hybrid approach

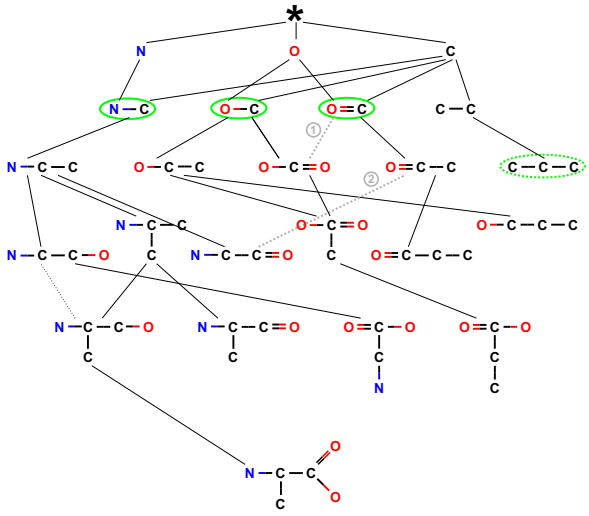


Figure 5: The check for already used seeds makes the search tree smaller than before

There is an elegant solution for this problem. Instead of preventing extensions that *add* already used seed atoms we now discard extension that *lead* to already used seeds. As every seed is freely extendable all child fragments must have been already found. So the branch can be cut off at this point. Of course this test — which requires an SGI test of all previously used seeds against the new fragment candidate — is much more expensive than the check if the newly added atom has already been used. Nevertheless this speeds up the algorithm again, as we will see in the next section. Figure 5 shows the pruned searched tree. It contains fewer paths than before which means that less fragments are found more than once. For example the edge labeled 1 is not traversed anymore, because the child fragment of `O=C`, which is `O-C=O`, contains `O-C` which has been used as seed before. The same holds for edge 2 as the larger fragment `N-C-C=O` contains `N-C` which has already been used as a seed in an earlier branch.

3 Experimental results

In order to evaluate our new hybrid mining algorithm we ran several experiments on the already mentioned NCI-HIV[8] and also NCI's H23 cancer screen [7] datasets. The latter dataset contains about 30,000 compounds that have been checked for possible activity against cancer. It consists of two groups, one having shown activity and the other having not shown any activity against cancer.

The environment was a Dual-Pentium II machine at 850MHz running Windows 2000 with 1GB RAM. We used Suns JDK 1.4.1 and allowed a maximum Java heap of 750 MB. The experiments were run using a wide range of embedding thresholds. Additionally we used the proposed seed based pruning strategy and compared it against the hybrid approach without pruning as discussed earlier. Together with the time needed to find fragments with a minimum support of 5% in the focus molecules (1,503 for NCI-HIV and 4,996 for NCI-H23) and a maximum support of 0.05% in the complement molecules (≈35,000 for NCI-HIV and ≈25,000 for NCI-H23) we also included an approximation of the total memory that was needed. To determine this we invoked the garbage collector after every search tree level and recorded the current heap size. Of course these numbers are only an approximation of the real value.

In contrast to the normal *MoFa* operation mode we do not exclude fragments that only consist of carbon atoms. As we already explained, the main reason why the standard algorithm fails on huge datasets are pure carbon fragments that can be embedded several hundreds of thousands times. With the combination of FSG and *MoFa* we are now able to find even those fragments. The results are depicted in figure 6. From an embedding threshold of 50,000 onward the runtime

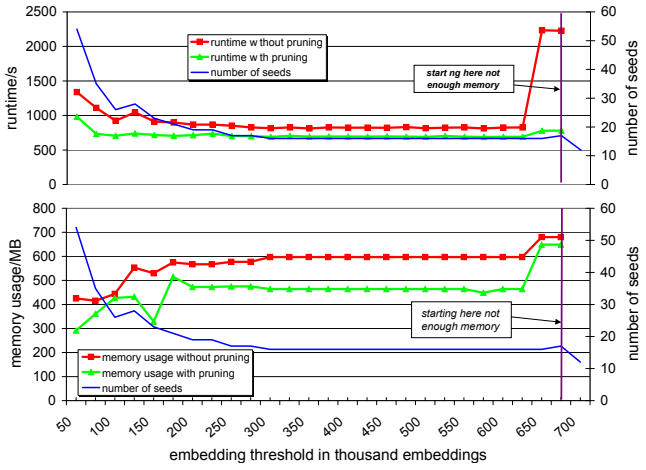


Figure 6: Results on the HIV dataset

left carbon atom is not extendable any more.

decreases initially. This is due to a drop in the number of seeds and therefore fewer fragments are found redundantly. Then a long interval of more or less constant runtime (and also memory usage) follows. At a threshold of about 625,000 allowed embeddings the runtime and memory consumption rise again. This is mainly due to frequent garbage collector calls. From a threshold of 675,000 onward the old *MoFa* problem reappears, i.e. there are too many embeddings to fit into main memory. The new seed based pruning always speeds up the computation, especially for higher embedding thresholds. The speedup varies between a factor of about 1.2 to 2.9.

The results on the NCI-H23 dataset are more interesting. We only show the numbers with the novel seed based pruning (see figure 7), because without it we were not able to mine the complete dataset without running out of memory regardless of the chosen threshold. This is clear evidence that this pruning strategy is not just helpful but may in some cases be the only way to get results. Another strange observation is the gap between 200,000 and 250,000 allowed embeddings for which even the hybrid approach ran out of memory. Let

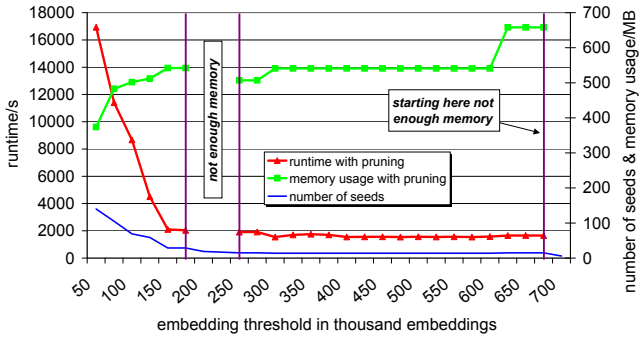


Figure 7: Results on the NCI-H23 dataset

us look closer at this phenomenon. The difference between the two thresholds 225,000 and 250,000 is that in the first case C-C-C is used as a seed whereas in the second it is C-C. Now, looking at the number of extensions of the two seeds, the reason for the gap becomes clear: C-C-C allows for 531,942 ring extensions and about 1.13 million atom-bond extensions. On the other side extending C-C results in "only" 407,479 ring extensions and 995,966 atom-bond extensions. Thus even though the number of embeddings is smaller for the the larger seed it creates more children fragments than the other one. In the end this leads to a memory consumption of more than the available 750MB.

However, the question remains: why are the numbers of extensions so dramatically different? The answer are fused rings. Figure 8 illustrates this issue. There are 18 possible ways to embed C-C into the shown "molecule" (see columns one and two of the first table[3] in figure 8), whereas there are 24 for C-C-C (see second table). The 18 embeddings of C-C create 56 different ring extensions, the 24 embeddings of C-C-C create 88 ring extensions. The different possibilities are shown in the last two columns of the two tables. The rings have to be constructed in both directions, clockwise and counterclockwise. For example starting with embedding C^1-C^2 one can close ring 1 by either creating the new fragment C^1-C^2-C^3-C^4-C^5-@C^1 or C^2-C^1-C^5-C^4-C^3-@C^2, respectively. Most of the embeddings lie in both rings so there are four different ring extensions, whereas for the remaining ones there are only extensions in one of the two rings.

To summarize, for this example the ratio between number of embeddings and their extensions – which is $\frac{56}{18} = 3.\bar{1}$ for C-C and $\frac{88}{24} = 3.\bar{6}$ for C-C-C – is smaller for the smaller core. Transferred to the complete dataset this is the main cause that even though the number of embeddings is lower for the bigger core it creates more extensions and thus needs more memory.

[3]Due to space limitations we omit the atom symbols in the two tables and only show their numbers.

4 Conclusions

We presented a new approach that addresses a general problem of all graph based data mining techniques: Either the algorithm needs huge amounts of memory or the runtime is slow. By combining two algorithms – FSG and *MoFa*, the first one featuring a small memory footprint, the second one much faster with high storage demands – we were able to mine even on huge datasets in reasonable times. To overcome the effects when different approaches (especially regarding the search strategy) are combined in this way, we needed to developed a novel pruning strategy. Future work will study whether there are combinations of other algorithms and/or strategies which perform better than the presented one.

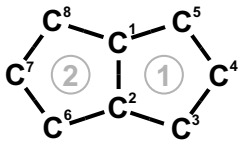
References

[1] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD Intl. Conf. on Management of Data*, P. Buneman and S. Jajodia, Eds. Washington, D.C., USA: ACM Press, 1993, pp. 207–216.

[2] C. Borgelt and M. R. Berthold, "Mining molecular fragments: Finding relevant substructures of molecules," in *Proceedings of the IEEE Intl. Conf. on Data Mining ICDM*. Piscataway, NJ, USA: IEEE Press, 2002, pp. 51–58.

[3] P. W. Finn, S. Muggleton, D. Page, and A. Srinivasan, "Pharmacophore discovery using the inductive logic programming system PROGOL," *Machine Learning*, vol. 30, no. 2-3, pp. 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.



seed C^1-C^2			
embeddings C^1-C^2 $\rightarrow C^x$-C^y		**ring extensions**	
1-2 2-1	2	each with ring 1 and 2 clockwise and counter-clockwise	$2 \cdot 4 = 8$
2-3, 1-5, 2-6, 1-8 3-2, 5-1, 6-2, 8-1	8	each with ring 1 and 2 clockwise and counter-clockwise	$8 \cdot 4 = 32$
3-4, 4-3, 4-5, 5-4	4	ring 1 counter- and clockwise	$4 \cdot 2 = 8$
6-7, 7-6, 7-8, 8-7	4	ring 2 counter- and clockwise	$4 \cdot 2 = 8$
	18		56

seed C^1-C^2-C^3			
embeddings C^1-C^2-C^3 $\rightarrow C^x$-C^y-C^z		**ring extensions**	
1-2-3, 1-2-6, 2-1-8, 2-1-5 3-2-1, 6-2-1, 8-1-2, 5-1-2	8	each with ring 1 and 2 clockwise and counterclockwise	$8 \cdot 4 = 32$
2-3-4, 1-5-4, 2-6-7, 1-8-7 4-3-2, 4-5-1, 7-6-2, 7-8-1	8	each with ring 1 and 2 clockwise and counterclockwise	$8 \cdot 4 = 32$
3-4-5, 5-4-3	2	ring 1 counter- and clockwise	$2 \cdot 2 = 4$
6-7-8, 8-7-6	2	ring 2 counter- and clockwise	$2 \cdot 2 = 4$
8-1-5, 6-2-3 5-1-8, 3-2-3	4	each with ring 1 and 2 clockwise and counterclockwise	$4 \cdot 4 = 16$
	24		88

Figure 8: In fused rings C-C-C creates more extensions than C-C

[5] J. Huan, W. Wang, and J. Prins, "Efficient mining of frequent subgraphs in the presence of isomorphism," in *Proceedings of the 3rd IEEE Intl. Conf. on Data Mining ICDM*. Piscataway, NJ, USA: IEEE Press, 2003, pp. 549–552.

[6] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," in *Proceedings of the IEEE Intl. Conf. on Data Mining ICDM*. Piscataway, NJ, USA: IEEE Press, 2001, pp. 313–320.

[7] "Dtp human tumor cell line screen," http://dtp.nci nih.gov/docs/cancer/cancer_data.html.

[8] "HIV antiviral screen," http://dtp.nci nih.gov/docs/-aids/aids_data html.

[9] S. Nijssen and J. N. Kok, "A quickstart in frequent structure mining can make a difference," LIACS, Leiden University, Leiden, The Netherlands, Tech. Rep., April 2004.

[10] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proceedings of the IEEE Intl. Conf. on Data Mining ICDM*. Piscataway, NJ, USA: IEEE Press, 2002, pp. 51–58.

[11] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules," in *3rd Intl. Conf. on Knowledge Discovery and Data Mining*, D. Heckerman, H. Mannila, D. Pregibon, R. Uthurusamy, and M. Park, Eds. AAAI Press, 1997, pp. 283–296.