Graph Based Molecular Data Mining - An Overview*

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

Abstract – *In the last years quite a lot of algorithms concerning frequent graph pattern mining have been published. In this paper an overview on the different methods for graph data mining is given, starting with the greedy searches proposed in the middle of the ninties. The ILP-based approaches are taken into account as well as ideas influenced by basket analyses proposed lately. A remaining question is how the different approaches can be tailored to meet the needs for mining molecules. In this area special problems occur as molecules are not just "normal arbitrary graphs". There are structures that are typical and frequent as rings and chains, some node types resp. atoms occur more often than others. It is an unsolved question how chemically isomorphic mining can be handled.*

Keywords: Molecules, fragments, graph mining.

1 Introduction

The amount of available data is increasing very fast. With this data the desire for data mining is also growing. More and larger databases have to be searched to find interesting (and frequent) elements and connections between them. Most often the data of interest is very complex. It is common to model complex data with the help of graphs consisting of nodes and edges that are often labeled to store additional information. Having a graph database, it is interesting to find common graphs in it, connections between different graphs and graphs that are subgraphs of a certain number of entries.

This graph-based data mining has become more and more popular in the last few years. It has a broad range of applications. Examples are the analysis of XML documents, citation networks, CAD circuits, weblogs, and web searches [5]. A currently very popular area where graph bases data mining is applied is in drug discovery and compound synthesis. When analyzing molecules, it is common to represent their two-dimensional structure with atoms as labeled nodes and bonds as labeled edges. Within molecular databases, it is interesting to find patterns — called fragments in this context — that appear at least in a certain percentage of graphs. Another problem is to find fragments that are frequent in one part of the database but infrequent in the other. This way this substructure is separating the database into active and inactive molecules [2]. In chemo- or bioinformatics it is not only mining molecules that is interesting. Similar problems occur for protein databases. Here graph data mining can be used to find structural patterns in the primary, secondary and tertiary structure of protein categories [4].

Despite the need for graph data mining, the first published algorithm in this area appeared in the mid-1990s. *Subdue* [4] is the oldest algorithm but is still used in various applications. Being the first, the number of extensions available for *Subdue* is enormous. The algorithm is combined with background knowledge, inexact graph matching and there is also a parallelized variant available. Supervised and unsupervised mining is possible. It took a few more years before more and faster approaches appeared. In [8] graph databases are mined for simple paths, for a lot of other applications only trees are of interest [21]. Also Inductive Logic Programming [7] was applied in this area. At the beginning of the new millenium finally more and more and everytime faster approaches for general mining of graph databases were developed [2, 13, 24, 17]. The latest development, a system named *Gaston* [20], combines mining for paths, trees and graphs leading to a fast and efficient algorithm.

The rest of the paper is structured as follows: In section 2 we want to present the problem of graph based data mining. Section 3 explains the basic technqiues that can be used to mine in graph databases, section 4 briefly discusses some extensions to the basic algorithms for molecular databases. An outlook of future trends and open problems is given in 5. Finally the paper is concluded in 6.

2 Defining the problem

Theoretically mining in graph databases can be modeled as the search in the lattice of all possible subgraphs. In figure 1 a small example is shown based on a small molecule named *cyclin* (shown at the bottom of the figure). The six nodes are labeled corresponding to their atom types (`O`, `N` and `C`), the edges are labeled with their corresponding bond types (single and double bonds). All possible subgraphs of this small graph are listed in this figure. At the top, the empty graph modeled with `*` is shown. In the next row all possible fragments containing just one atom (or zeros bonds) are

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

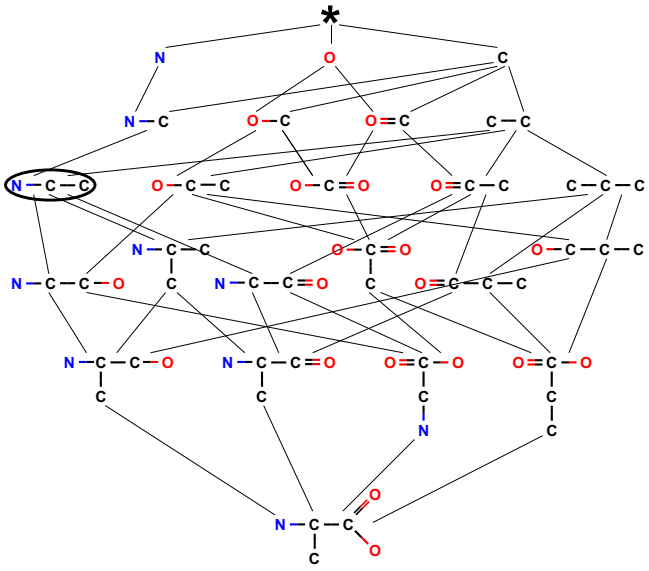

Figure 1: The lattice of all subgraphs in cyclin

listed. The second row contains subgraphs with one bond. The ”parent-child“ relation between the subgraphs (indicated by the lines) is the subgraph property. The empty graph can be embedded in every graph containing one node. The graph containing just one node labeled N can be embedded in a one edge graph containing nodes N and C. Please note, that in figure 1 no fragment with one bond is given containing atom types N and O. As there is no such subgraph in our running example, the lattice does not contain a graph like this. Only graphs that are real subgraphs are listed in the lattice. In the third row, fragments with two bonds are shown and so on. At the bottom of the figure, the complete molecule with five bonds is given. Each subgraph given in the lattice above can be embedded in this graph. All graph mining algorithms have in common, that they search this subgraph lattice. They are interested in finding a subgraph (or several subgraphs) that can be embedded as often as possible in the graph to be mined. In figure 1 the circled graph can be embedded twice in the running example.

When mining real life graph databases, the situation is of course much more complex. Not only one but a lot of graphs are analyzed leading to a very large lattice. Searching this lattice can be done depth or breadth first. When searching depth first in figure 1, the first discovered subgraph will be N followed by N-C, N-C-C and so forth. So first all subgraphs containing N, in the next branch all containing O are found. If the lattice is traversed breadth first, all subgraphs in one level of the lattice, i.e. structures that have the same number of edges, are searched before the next level is started. The main disadvantage of breadth first search is the larger memory consumption because in the middle of the lattice a large amount of subgraphs has to be stored. With depth first search only structures which amount is proportional to the size of the biggest graph in the database have to be recorded during the search.

Building this lattice of frequent subgraphs involves two main steps: *Candidate Generation*, where new subgraphs are created out of smaller ones, and *Support Computation* where the frequency or support of the new subgraphs in the database is determined. Both steps are highly complex and thus various algorithms and techniques have been developed to find frequent subgraphs in finite time with reasonable resource consumptions.

3 Basic techniques

We will now have a more detailed look on the two main steps of the search mentioned above, candidate generation and support computation. There are two popular ways of creating new subgraphs: merging smaller subgraphs that share a common core [17, 14] or extending subgraphs edge by edge [2, 24].

The merge process can be explained by looking at the subgraph lattice shown in figure 1. The circled subgraph has two parents, N-C and C-C. Both share the same core which is C. So the new fragment N-C-C is created by taking the core and adding the two additional edge-node pairs, one from each parent. There are two problems with this approach: First the common core needs to be detected somehow, which can be very expensive. Second a huge amount of subgraphs generated in this way may not even exist in the database. Merging e.g. N-C and O-C in the example will lead to N-C-O which does not occur in the database. To overcome this problem various modifications of the merge process have been developed [11, 23].

Extending fragments has the advantage that no cores have to be detected. New edge-node pairs (or sometimes only edges, if cycles are closed) are just added to an existing subgraph. Also here non-existing candidates can be generated, but there is an easy way to combine the generation of new subgraphs with the support computation, so that only existing candidates are created. As shown later, during the support computation the candidate subgraph has to be embedded into all graphs of the database (which is essentially a subgraph isomorphism test). Once an embedding has been found the surrounding of the subgraph is known and in the following extension step only edge-node pairs (or single edges) are added that really exist in the database's graphs. The small drawback is, that now *all* subgraph isomorphisms have to be computed and not just one as is normally required for the support computation. Nevertheless this technique is currently the fastest subgraph mining algorithms all rely on extending subgraphs.

Computing the support of new candidates can also be done in two different ways. For the first algorithm proposed *Subdue*, the MDL heuristic is used. It is based on the idea that the best substructure in a graph data base is the one that minimizes the underlying graph database most. Another already mentioned simple approach is to test subgraph isomorphism against all graphs in the database. This is a computationally expensive task because the subgraph isomorphism problem is NP-complete. However there is a small improvement for

this strategy, as it suffices to check for subgraph isomorphism only in the graphs where the parent graph(s) occur. Unfortunately this requires to keep a list of the graphs in which a subgraph occurs which can be quite memory consuming if the database is large.

The other way of calculating the support is the use of so-called embeddings lists. An embedding can be thought of as a stored subgraph isomorphism i.e. a map from the nodes and edges in the subgraph to the corresponding nodes and edges in the graph. Now, if the support of a new greater subgraph has to be determined, the position in the graph where it can occur is already known and only the new nodes and edges have to be checked. This reduces the time to find the isomorphism but comes with the drawback of enormous memory requirements as all embeddings of a subgraph in the database have to be stored which can be millions for small subgraphs on even medium-sized databases of about 25,000 items. Using embedding lists the actual support for a structure can be determined by counting the number of different graphs that are referred to by the embeddings. This can be done in linear time.

In general it is not possible to traverse the complete lattice because the number of subgraphs is too large to be handled efficiently. Also most of them can be reached by following many different paths in the lattice (see figure 1). A mechanisms is needed to prune the search tree that is built during the discovery process. Whereas *Subdue* uses a beam-search, the pruning became more sophisticated during the years. First of all it is obvious, that a supergraph of a graph that is infrequent must be infrequent, too. It cannot occur in more graphs than its parents in the lattice. This property is also known as the *antimonocity constraint* and was first used in the Apriori algorithm [1]. Once the search reaches a point where a graph does not occur in enough items of the database any more, this branch can be pruned. This leads to a drastic reduction of the number of subgraphs to be searched. To speed up the search even further various authors have proposed additional pruning strategies that rely on canonical representations of the subgraphs or local orders on the nodes and edges. This information can be used to prune the search tree even further while still finding all frequent fragments.

4 Mining Molecules

Nearly every algorithm mentioned befor was applied to molecular databases. Some of them have even been developped with molecular databases in kind [2], [12], [6], [16], [8], [10]. Especially when using Inductive Logic Programming molecules are a common application area [15], [7].

Based on the basic algorithms discussed in the previous section quite a few extenions have been developed. We want to present three of them and discuss them briefly. The first extension deals with the search for structures that are similar but not equal. This is again of interest in biochemistry where there is a huge amount of molecules that behave the same in chemical reactions but have slight variations in their structure. A ”crisp“ search for frequent subgraph will most certainly not find them. And even *if* some or all structures of this chemical equivalent group are found they are in general not visible as such a group at first. To solve this issue in [9] an approach to find fragments with wildcards was presented. There fragments containing carbon rings in which some of the atoms have been replaced by e.g. nitrogen are found as groups of equivalent fragments. Another similar technique was described in [18] where fragments with carbon chains of varying lengths are found. An example is shown in figure 2. The three fragments indicated by the shadowed regions are not identical but equivalent in the sense, that the only difference is the length of the carbon chain.

The next interesting enhancement is the search for so-called *closed subgraphs*. A closed subgraph is a subgraph where no supergraph of it has the same frequency in the database. This concept useful for two reasons. First, again looking at biochemistry, the user is often only interested in those closed structures and not in smaller ones that have the same support values. In general a biochemist wants to see the biggest fragment with certain properties. Second using this property the search can be sped up. Extensions of two different algorithms have been presented in [25] and [19, 3].

A third extension tries to cope with the problem the huge memory requirements most algorithms have. In [22] a modification of the gSpan-algorithm is presented that swaps out data from memory to disk. In order to overcome the slow response times of hard disks a novel indexing scheme has been developed that avoids most of the random accesses to the graphs in the database. With the use of theses indexes the authors not only achieve impressive results on ver large databases that do not fit into memory but even improved the performance of the underlying algorithm when only in-memory data is processed.

5 Future trends

Although the current algorithms already perform quite well, there are still some open topics in graph-based data mining. Still memory and runtime are a challenge for most of the algorithms. Having real world graph databases containing millions of different graphs, various new algorithms and extensions of the existing ones are necessary. A promising research direction are parallel and distributed algorithms. Distributing the graphs and their subgraph lattice onto different machines can help in processing even larger databases than with current algorithms. It is an open question how to realize the distribution without searching different branches of the lattice several times. Searching only in one part of the smaller database might also lead to the rejection of frequent subgraphs as they may be infrequent in this part but frequent in the whole database. If on another machine the support for this subgraph is high enough to be reported the total number of subgraphs is not correct.

As already mentioned the need for inexact graph matching might become more and more important. The available solutions are only very specialized extensions of existing algorithms. The use of graph transformation rules might be

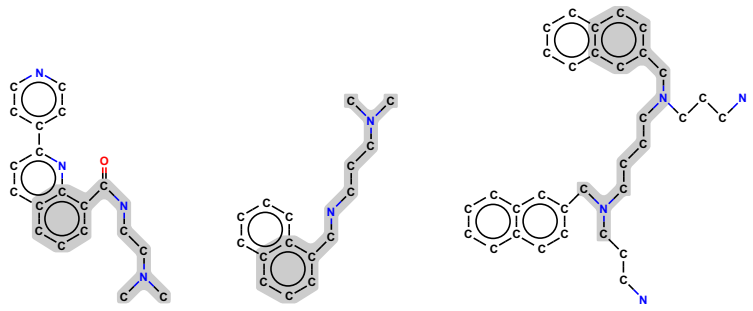

Figure 2: Three equivalent fragments that only differ in the length of the carbon chain

helpful in developing more general approches.

Finally visualization of the search and the results is difficult. A semi-automatic search can be helpful. A human expert decides whether the search in a subpart of the lattice is useful or if the search is more promising in another direction. To achive this goal a visualization component is necessary that allows browsing in the graph database showing the embeddings of subgraphs.

6 Conclusion

Graph Data Mining is a currently very active research field. At the main data mining conferences of the ACM or the IEEE every year various new approaches appear. The application areas of graph data mining are widespread ranging from biology and chemistry to internet applications. Wherever graphs are used to model data, data mining in graph data bases is useful.

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] C. Borgelt, T. Meinl, and M. R. Berthold, "Advanced Pruning Strategies to Speed Up Mining Closed Molecular Fragments," October 2004, to appear in the Proceedings of SMC 2004, Den Haag, The Netherlands.

[4] D. J. Cook and L. B. Holder, "Graph-based data mining," *IEEE Intelligent Systems*, vol. 15, no. 2, pp. 32–41, 2000.

[5] D. J. Cook, N. Manocha, and L. B. Holder, "Using a graph-based data mining system to perform web search," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 17, no. 5, pp. 705–720, 2003.

[6] M. Deshpande, M. Kuramochi, and G. Karypis, "Automated approaches for classifying structures," in *Proceedings of the second Workshop on Data MIning in Bioinformatics*, 200.

[7] P. Finn, S. Muggleton, D. Page, and A. Srinivasan, "Pharmacophore discovery using the inductive logic programming system Progol," *Machine Learning*, vol. 30, pp. 241–271, 1998.

[8] C. Helma, S. Kramer, and L. de Raedt, "The molecular feature miner molfea," in *Proceedings of the Beilstein-Institut Workshop*, M. G. Hicks and C. Kettner, Eds., May 2002.

[9] H. Hofer, C. Borgelt, and M. R. Berthold, "Large scale mining of molecular fragments with wildcards," in *Advances in Intelligent Data Analysis V*, ser. Lecture Notes in Computer Science (LNCS). Springer Verlag, 2003, no. 2810, pp. 380–389.

[10] J. Huan, W. Wang, D. Bandyopadhyay, J. Snoeyink, J. Prins, and A. Tropsha, "Mining protein family specific residue packing patterns from protein structure graphs," in *Proceedings of the Intl Conf on Research in Computational Molecular Biology (RECOMB)*, 2004.

[11] 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.

[12] A. Inokuchi, T. Washio, T. Okada, and H. Motoda, "Applying the apriori-based graph mining method tp mutagenesis data analysis," *Journal of Computer Aided Chemistry*, vol. 2, pp. 87–92, 2001.

[13] A. Inokuchi, T. Washio, and H. Motoda, "Complete mining of frequent patterns from graphs: Mining graph data," *Machine Learning*, vol. 50, no. 3, pp. 321–354, 2003.

[14] A. Inokuchi, T. Washio, K. Nishimura, and H. Motoda, "A fast algorithm for mining frequent connected subgraphs," IBM Research, Tokyo Research Laboratory, Tech. Rep., 2002.

[15] R. King, A. Srinivasan, and L. Dehaspe, "Warmr: A data mining tool for chemical data," *Journal of Computer-Aided Molecular Design*, vol. 15, pp. 173–181, 2001.

[16] S. Kramer, L. D. Raedt, and C. Helma, "Molecular feature mining in HIV data," in *Proceedings of the seventh ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. ACM Press, 2001, pp. 136–143.

[17] 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.

[18] T. Meinl, "Erweiterte Fragmentsuche in Molekül datenbanken," diploma thesis, Computer Science Department 2, University of Erlangen-Nuremberg, August 2004.

[19] T. Meinl, C. Borgelt, and M. R. Berthold, "Discriminatice Closed Fragment Mining and Perfect Extensions in MoFa," August 2004, to appear in the Proceedings of STAIRS 2004, Valencia, Spain.

[20] 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.

[21] U. Rückert and S. Kramer, "Frequent free tree discovery in graph data," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM Press, 2004, pp. 564–570.

[22] C. Wang, W. Wang, J. Pei, and Y. Z. amd Baile Shi, "Scalable Mining of Large Disk-based Graph Databases," August 2004, to appear in the Proceedings of KDD 2004, Seattle, Washington, USA.

[23] T. Washio and H. Motoda, "State of the art of graph-based data mining," *SIGKDD Explor. Newsl.*, vol. 5, no. 1, pp. 59–68, 2003.

[24] 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.

[25] ——, "Closegraph: Mining closed frequent graph patterns," in *Proceedings of the ninth ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. ACM Press, August 2003, pp. 286–295.