Visualization and Grouping of Graph Patterns in Molecular Databases

Edgar H. de Graaf Walter A. Kosters
Joost N. Kok
Leiden Institute of Advanced Computer Science
Leiden University, The Netherlands

Jeroen Kazius
Leiden/Amsterdam Center for Drug Research
Leiden University, The Netherlands

Abstract

Mining subgraphs is an area of research where we have a given set of graphs, and we search for (connected) subgraphs contained in these graphs. In this paper we focus on the analysis of graph patterns where the graphs are molecules and the subgraphs are patterns. In the analysis of fragments one is interested in the molecules in which the patterns occur. This data can be very extensive and in this paper we introduce a technique of making it better available using visualization. The user does not have to browse all the occurrences in search of patterns occurring in the same molecules; instead the user can directly see which subgraphs are of interest.

1 Introduction

Mining frequent patterns is an important area of data mining where we discover substructures that occur often in (semi-)structured data. The research in this work will be in the area of frequent subgraph mining. These *subgraphs* are connected vertex- and edge-labeled graphs that are subgraphs of a given set of graphs. A subgraph is considered to be frequent if it occurs in at least *minsupp* transactions, where *minsupp* is a user-defined threshold above which patterns are considered to be frequent. The *frequent subgraph mining* algorithm will discover all these frequent subgraphs. Figure 1 shows an example graph and two of its subgraphs.

This work is motivated by bio-chemists wishing to view co-occurrences of subgraphs in a dataset of molecules (graphs):

- For a bio-chemist it is very interesting to know which fragments occur often together, for example in so-called active molecules. This is because frequent co-occurrence implies that the fragments are needed simultaneously for biological activity.

- Pharmaceutical companies provide generated libraries of molecules. A visualization of co-occurrences in molecule libraries gives a bio-chemist insight how the libraries are constructed by the company.

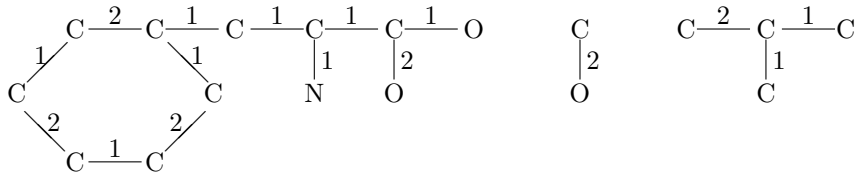

Figure 1: An example of a graph (the amino acid Phenylalanine) in the molecule data set and two of its many (connected) subgraphs, also called patterns or fragments.

The distance between patterns, the amount of co-occurrence, can be measured by calculating in how many graphs (or transactions) only one of the two patterns occurs: if this never happens then these patterns are very close to each other and if this always happens then their distance is very large.

We will define our method of building a co-occurrence model and show its usefulness. To this end, this paper makes the following contributions:
— The visualization of co-occurring graph patterns.
— We improve the clarity of the visualization by grouping.
— We will define a measure of calculating distances between patterns and show how it can be calculated (Section 2 and Section 3).
— An empirical discussion of model construction for visualizing co-occurrence (Section 5).

The mining techniques for molecules in this paper make use of a graph miner called gSpan, introduced in [17] by Yan and Han.

For the visualization a method of pushing and pulling points in accordance with a distance measure is used. The main reason to choose this particular method was because it enables us to put a limit on the number iterations and still have a result. Similar techniques were used in [1] to cluster criminal careers and in [8] for clustering association rules.

This research is related to research on clustering, in particular of molecules. Also our work is related to frequent subgraph mining and frequent pattern mining when lattices are discussed. In [18] Zaki et al. discuss different ways for searching through the lattice and they propose the Eclat algorithm.

Clustering in the area of biology is important because of the visualization that it can provide. In general our work is related to SOMs as developed by Kohonen (see [7]), in the sense that SOMs are also used to visualize data through a distance measure. A Self-Organizing Map (SOM) is a type of artificial neural network that is trained to produce a low dimensional representation of the training samples. A SOM is constructed by moving the best matching point and its neighbours (within a lattice of neurons) towards the input node. SOMs have been used in a biological context many times, for example in [5, 11]. In

some cases molecules are clustered via numeric data describing each molecule, in [16] clustering such data is investigated. Also our work is related to work done on the identification *structure activity relationships* (SARs) where one relates biological activity of molecules by analyzing their chemical structure [3, 6] in the sense that in our work the structure of a graph is used to build a model. In [2, 13, 14] a statistical analysis was done on the presence of fragment substructures in active and inactive molecules. However our work is not concerned with the discovery of SARs, but with co-occurrence of subgraphs occurring in a collection of graphs. More related is the work done by Lameijer et al. in [9]. This work is concerned with co-occurring fragments discovered with a graph splitting. Graph splitting breaks molecules at topologically interesting points. Also they use a frequency threshold to filter out some fragments after they were generated, however they do not use frequent pattern mining techniques. Furthermore they do not build a co-occurrence model or a similar visualization of co-occurrence. Figure 2 shows two co-occurring subgraphs (fragments) discovered by Lameijer et al. in their dataset of molecules.

In [4] the current setup is used to cluster data; that paper discusses an application that enables the user to further explore the results from a frequent subgraph mining algorithm, by browsing the lattice of frequent graphs.

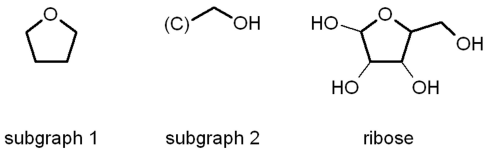

Figure 2: An example of co-occurring subgraphs from [9] with an example molecule.

The overview of the rest of the paper is as follows. In Section 2 our distance measure is introduced, in Section 3 we discuss our method of grouping, in Section 4 we introduce the visualization and finally in Section 5 we discuss our experimental results.

2 Distance Measure

As was mentioned in the introduction, we are interested to know if patterns occur in the same graphs in the dataset of graphs. Patterns in this work are *connected subgraphs*.

The distance measure will compute how often subgraphs occur in the same graphs of the dataset. In the case of our working example it will show if different patterns (subgraphs) exist in the same molecules in the database. This distance measure is known as the Jaccard metric and was primarily chosen for

its common use in Bio-informatics (see [15]). It is also easy to compute, given the appropriate supports; it doesn't make use of complicated graph comparisons, that would slow down the process. Formally we will define the distance measure in the following way (for graphs g_1 and g_2):

$$dist(g_1, g_2) = \frac{support(g_1) + support(g_2) - 2 \cdot support(g_1 \wedge g_2)}{support(g_1 \vee g_2)} \quad (1)$$

Here $support(g)$ is the number of times a (sub)graph g occurs in the set of graphs; $support(g_1 \wedge g_2)$ gives the number of graphs (or transactions) with both subgraphs g_1 and g_2 and $support(g_1 \vee g_2)$ gives the number of graphs with at least one of these subgraphs. The numerator of the *dist* measure computes the number of times the two graphs do not occur together in one graph of the dataset. We divide by $support(g_1 \vee g_2)$ to make the distance independent from the total occurrence, thereby normalizing it. We can reformulate *dist* in the following manner:

$$dist(g_1, g_2) = \frac{support(g_1) + support(g_2) - 2 \cdot support(g_1 \wedge g_2)}{support(g_1) + support(g_2) - support(g_1 \wedge g_2)} \quad (2)$$

In this way we do not need to separately compute $support(g_1 \vee g_2)$ by counting the number of times subgraphs occur in the graphs in the dataset.

The measure is appropriate for our algorithm because it exactly calculates the number of transactions in which both patterns *do not* exist, hence a small distance means much co-occurrence. This measure also normalizes the exact co-occurrence, otherwise very frequent patterns can be considered mutually more distant compared to other points with the same proportional co-occurrence.

The distance measure satisfies the usual requirements, such as the triangular inequality. Note that $0 \leq dist(g_1, g_2) \leq 1$ and $dist(g_1, g_2) = 1 \Leftrightarrow support(g_1 \wedge g_2) = 0$, so g_1 and g_2 have no common transactions in this case. If $dist(g_1, g_2) = 0$, both subgraphs occur in exactly the same transactions, but they are not necessarily equal.

3 Optimization: Restriction to Frequent Subgraphs and Grouping

In practise it is possible for the user to select a set of patterns for visualization. In this context we consider an optimization to be an automated selection of patterns such that the algorithm faster provides a model within reasonable time. The **first** optimization is to restrict the patterns to frequent patterns. Patterns (subgraphs) are considered to be frequent if they occur in at least *minsupp* graphs in the dataset. If we do not use frequent patterns we simply have too many patterns and, the frequent patterns give a comprehensive overview of the patterns. Efficient algorithms exist for finding frequent subgraphs, e.g., [17].

The **second** optimization is grouping: we group subgraphs and we will treat them as one point in our co-occurrence model. This will reduce the number of

points. Moreover, the visualization will now show more directly the structural unrelated patterns, since related patterns are grouped. This will show to a biochemist the structural unrelated patterns that suggest to be together needed for biological activity.

The formula for the distance between supergraph g_2 and subgraph g_1 originates from Equation 2, where $support(g_1 \wedge g_2) = support(g_2)$:

$$\begin{aligned} dist(g_1, g_2) &= \frac{support(g_1) + support(g_2) - 2 \cdot support(g_2)}{support(g_1) + support(g_2) - support(g_2)} \\ &= \frac{support(g_1) - support(g_2)}{support(g_1)} \end{aligned}$$

The frequent pattern mining algorithm gives rise to a so-called lattice, in which the frequent subgraphs are ordered with respect to supergraphs. All information used to compute these distances can be retrieved from the lattice information provided by the graph mining algorithm, when we focus on the subgraph-supergraph pairs. This information is needed by the graph mining algorithm to discover the frequent subgraphs and so the only extra calculating is done when *dist* does a search in this information.

Of course, many graphs have no parent-child relation and for this reason we define *lattice_dist* in the following way:

$$lattice\_dist(g_1, g_2) = \begin{cases} dist(g_1, g_2) & \text{if } g_2 \text{ is a supergraph of } g_1 \\ & \text{or } g_1 \text{ is a supergraph of } g_2 \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

Note that $lattice\_dist(g_1, g_2) < 1$ if g_1 is a subgraph of g_2 and has non-zero support, or the other way around.

We will now organize "close" patterns into groups. The algorithm forms groups hierarchically, but this can be done fast because only related subgraph are compared and also as a consequence all distances can be computed with the lattice. Now we need a distance between groups of patterns $C_1 = \{g_1, g_2, \ldots, g_n\}$ and $C_2 = \{h_1, h_2, \ldots, h_m\}$:

$$grdist(C_1, C_2) = \begin{cases} max(PG) & \text{if } PG \neq \emptyset \\ -1 & \text{otherwise} \end{cases} \quad (4)$$

$$PG = \{lattice\_dist(g, h) \mid g \in C_1, h \in C_2, lattice\_dist(g, h) \neq 1\}$$

Two clusters should not be merged if their graphs do not have a supergraph-subgraph relation, so we do not consider graphs where $lattice\_dist(g, h) = 1$. The value of *grdist* is -1 if no maximal distance exists, and clusters will not be merged in the algorithm.

The parameter *maxdist* is a user-defined threshold giving the largest distance allowed for two clusters to be joined. Note that grouping is efficient due to the fact that we can use the lattice information stemming from the frequent graph mining algorithm.

The outline of the algorithm is the following:

initialize $\mathcal{P}$ with sets of subgraphs of size 1 from the lattice
while $\mathcal{P}$ was changed or was initialized
  Select C_1 and C_2 from $\mathcal{P}$ with minimal *grdist* $(C_1, C_2) \geq 0$
  if $grdist(C_1, C_2) \leq maxdist$ **then**
    $\mathcal{P} = \mathcal{P} \cup \{C_1 \cup C_2\}$
    Remove C_1 and C_2 from $\mathcal{P}$

GROUPING

4 Visualization

We will visualize co-occurrence by positioning all groups in a 2-dimensional area. We take the Euclidean distance $eucl\_dist(C_1, C_2)$ between the 2D coordinates of the points corresponding with the two groups (of frequent subgraphs) C_1 and C_2.

The graphs in a group occur in almost all the same transactions, hence the distance between groups is assumed to be the distance between any of the points of the two groups. We choose to define the distance between groups as the distance between a smallest graph of each of the two groups (*size* gives the number of vertices): for $g_1 \in C_1$ and $g_2 \in C_2$ with $size(g_1) = min(\{size(g) \mid g \in C_1\})$ and $size(g_2) = min(\{size(g) \mid g \in C_2\})$, we let $group\_dist(C_1, C_2) = dist(g_1, g_2)$.

The coordinates (x_{C_1}, y_{C_1}) and (x_{C_2}, y_{C_2}) of the points corresponding with C_1 and C_2 are adapted by applying the following formulas:

1. $x_{C_1} \leftarrow x_{C_1} - \alpha \cdot (eucl\_dist(C_1, C_2) - group\_dist(C_1, C_2)) \cdot (x_{C_1} - x_{C_2})$
2. $y_{C_1} \leftarrow y_{C_1} - \alpha \cdot (eucl\_dist(C_1, C_2) - group\_dist(C_1, C_2)) \cdot (y_{C_1} - y_{C_2})$
3. $x_{C_2} \leftarrow x_{C_2} + \alpha \cdot (eucl\_dist(C_1, C_2) - group\_dist(C_1, C_2)) \cdot (x_{C_1} - x_{C_2})$
4. $y_{C_2} \leftarrow y_{C_2} + \alpha \cdot (eucl\_dist(C_1, C_2) - group\_dist(C_1, C_2)) \cdot (y_{C_1} - y_{C_2})$

Here α $(0 \leq \alpha \leq 1)$ is the user-defined learning rate.

Starting with random coordinates for the groups, we will build a 2D model of relative positions between groups by randomly *choosing two groups r times and applying the formulas*. This is a kind of push and pull algorithm which yields a visualization in which the distances in 2D correspond to the distances in the pattern space. Note that we always have a visualization: the longer we run the algorithm, the better the Euclidean distances correspond to the distances between groups in the pattern space.

5 Performance

The experiments are organized such that we first show that the distances are approximated correctly. Secondly we will discuss runtime in the case of different *minsupp* settings for different datasets. Finally through experiments we analyze the speed-up due to making groups first.

One dataset we use, the `4069.no_aro` *dataset*, containing 4,069 molecules; from this we extracted a lattice containing the 1,229 most frequent subgraphs. This dataset was provided by Leiden/Amsterdam Center for Drug Research (LACDR). Other datasets we use are datasets of the National Cancer Institute (NCI), and can be found in [12]. One of these datasets contains 32,557 2D structures (molecules, average size is 26.3 nodes) with cancer test data as of August 1999; we will call this dataset the `NCI.normal.99` *dataset*. The other NCI dataset contains 250,251 molecules and we will call this dataset the `NCI.large.99` *dataset*.

All experiments were performed on an Intel Pentium 4 64-bits 3.2 GHz machine with 3 GB memory. As operating system Debian Linux 64-bits was used with kernel 2.6.8-12-em64t-p4.

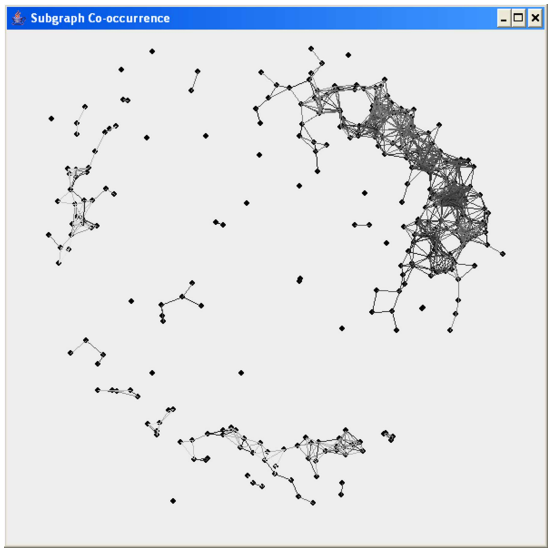

Figure 3: Clusters for graphs in the `4069.no_aro` dataset built in 24.5 seconds, connecting points at distance 0.05 or lower ($\alpha = 0.1$, $maxdist = 0.1$, $r = 1,000,000$).

Figure 3 shows how points, that represent subgraphs occurring in the same graphs (transactions) of the dataset, are close together. We draw lines between points if their Euclidean distance is ≤ 0.05. The darker these lines the lower

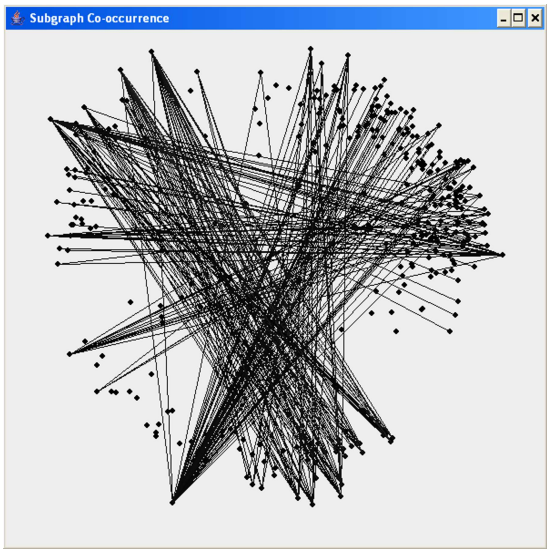

Figure 4: Clusters of graphs in the `4069.no_aro` dataset built in 24.5 seconds, connecting points at distance 0.95 or higher ($\alpha = 0.1$, $maxdist = 0.1$, $r = 1,000,000$).

their actual distance and in this way one can see gray clusters of close groups of subgraphs. Some groups are placed close but their actual distance is not close (they are light grey). This is probably caused by the fact that these groups do not occur together with some specific other groups, so being far away from these other ones.

In Figure 4 we draw lines between points with a Euclidean distance $\geq$ 0.95. The darker these lines the higher their actual distance. The figure shows their actual distance to be big also (the lines are black). Also Figure 4 shows bundles of lines going to one place. This probably is again caused by groups not occurring together with the same other groups.

The error for the cluster model for the `4069.no_aro` dataset decreases quickly, see Figure 5. After pushing or pulling 10,000 group pairs it becomes already hard to reduce the error further making a reduction of model building time possible.

In one experiment we assumed that the distances could not be stored in memory. In this experiment we first clustered 1,229 patterns without grouping, taking 81 seconds. However, grouping reduced the number of requests to the compressed occurrence data and because of this with grouping model construction was done in 48 seconds ($\alpha = 0.1$, $r = 1,000,000$, $maxdist = 0.1$, dataset is `4069.no_aro`).

Table 1 shows the runtime where *minsupp* varies. Obviously for a lower

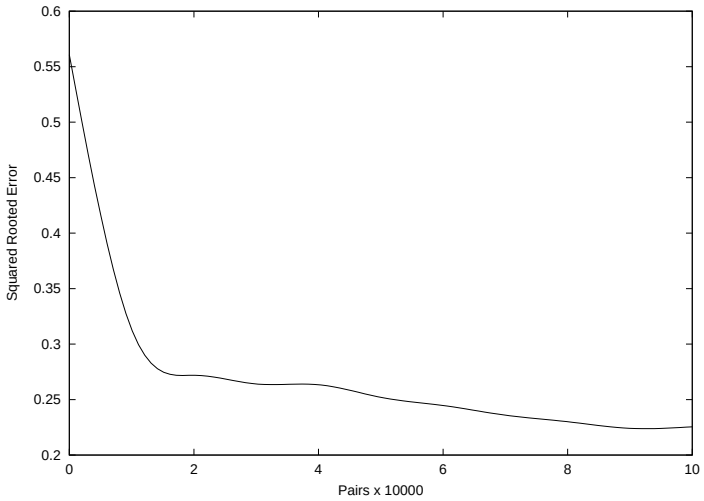

Figure 5: Root squared error for distance given by the cluster model for the `4069.no_aro` dataset ($\alpha = 0.1$).

minsupp it takes longer to build the model, but for 12,734 subgraphs a model is still built within an acceptable time frame.

Table 2 and 3 show the runtime where *minsupp* varies, it is set to a percentage of the total dataset size. Results show that the algorithm is able to handle the `NCI.normal.99` dataset of 32,557 molecules and `NCI.large.99` dataset of 250,251 molecules, even with a low *minsupp*, within a reasonable time frame.

Our final experiments were done to show how the runtime is influenced by the *maxdist* threshold and how much the preprocessing step influences runtime. Here we assume that the distances can be stored in memory. In Figure 6 the influence on runtime is shown and to each line a Bézier curve is fitted (the degree is the number of datapoints). The figure displays preprocessing to proceed more or less stable.

minsupp	average runtime (sec) $\pm$ *stdev*	number of subgraphs
200	2204.60 $\pm$ 6.36	12,734
300	335.10 $\pm$ 2.00	4,571
400	45.75 $\pm$ 0.17	2,149
500	17.95 $\pm$ 0.23	1,229

Table 1: Runtime performance in seconds for different *minsupp* settings for the `4069.no_aro` dataset ($\alpha = 0.1$, $r = 10,000$, $maxdist = 0.2$).

minsupp	average runtime (sec) ± *stdev*	number of subgraphs
5%	1495.57 ± 5.41	5,663
10%	160.82 ± 0.42	1,447
20%	17.09 ± 0.13	361
30%	4.64 ± 0.01	158

Table 2: Runtime performance in seconds for different *minsupp* settings for the `NCI.normal.99` dataset ($\alpha = 0.1$, $r = 10,000$, $maxdist = 0.2$).

minsupp	average runtime (sec) ± *stdev*	number of subgraphs
5%	2080.12 ± 9.40	2,391
7%	840.49 ± 11.67	1,313
10%	301.58 ± 3.57	648
15%	91.35 ± 0.59	332

Table 3: Runtime performance in seconds for different *minsupp* settings for the `NCI.large.99` dataset ($\alpha = 0.1$, $r = 10,000$, $maxdist = 0.2$).

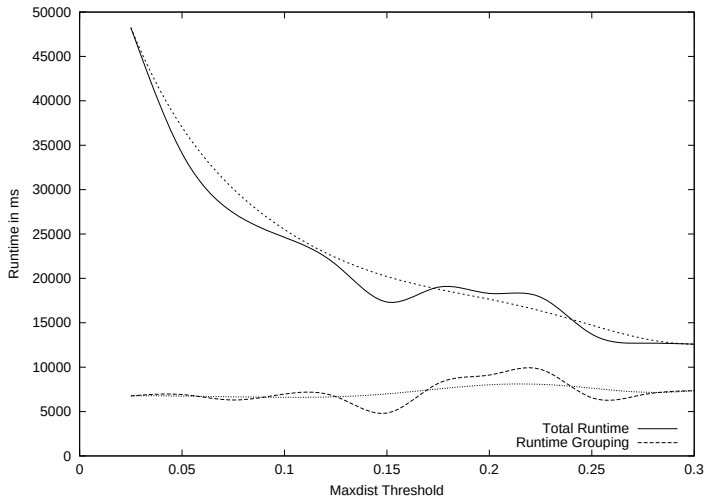

Figure 6: Average runtime for the `4069.no_aro` dataset with varying *maxdist* ($\alpha = 0.1$, nr. of patterns = 1,229, $r = 1,000,000$).

In Figure 7 results show the runtime for the `NCI.normal.99` dataset with approximatelly an equal number of patterns. The performance for grouping is nearly the same as for the `4069.no_aro` dataset. This performance depends more on the number of patterns that are grouped. The results indicate that the total runtime depends on the size of the dataset, but that runtime can be improved strongly by better selecting the *maxdist* threshold.

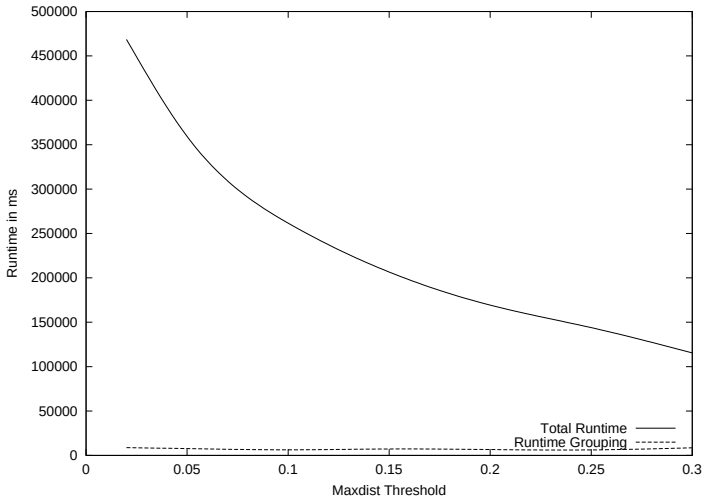

Figure 7: Average runtime for the `NCI.normal.99` dataset with varying *maxdist* ($\alpha = 0.1$, nr. of patterns = 1,447, $r = 1,000,000$).

The first analysis of results shows promising patterns, see Figure 8. The results show two frequent subgraphs (a) and (b) occurring together. This suggests that patterns (c) and (d) might also occur together, requiring further research.

Also biochemists in Leiden are actively researching the development of simple biologically active molecules consisting of fragments (subgraphs) not co-occurring frequently [10]. Modelling co-occurrence will hopefully help improve their analysis.

6 Conclusions and Future Work

Presenting data mining results to the user in an efficient way is important. In this paper we propose a visualization of a co-occurrence model for subgraphs that enables quicker exploration of occurrence data.

The forming of groups improves the visualization. The visualization enables the user to quickly select the interesting subgraphs for which the user wants to investigate the graphs in which the subgraphs occur. Additionally the model can be built faster because of the grouping of the subgraphs.

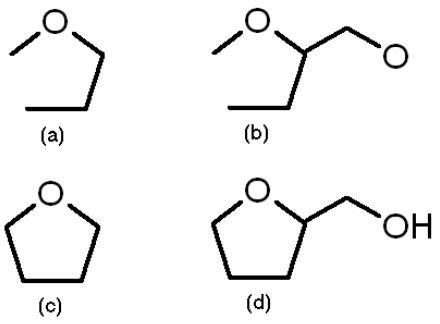

Figure 8: Two co-occurring frequent patterns (a) and (b), and two potentially interesting ones (c) and (d).

In the future we want to take a closer look at grouping where the types of vertices and edges and their corresponding weight also decide their group. Furthermore, we want to investigate how we can compress occurrence more efficiently and access it faster.

Acknowledgments This research is carried out within the Netherlands Organization for Scientific Research (NWO) MISTA Project (grant no. 612.066.304). We thank Siegfried Nijssen for his implementation of gSpan.

References

[1] Bruin, J.S. de, Cocx, T.K., Kosters, W.A., Laros, J.F.J. and Kok, J.N.: *Data Mining Approaches to Criminal Career Analysis*, in Proc. 6th IEEE International Conference on Data Mining (ICDM 2006), pp. 171–177.

[2] Gao, H., Williams, C., Labute, P. and Bajorath, J.W.: *Binary Quantitative Structure-Activity Relationship (QSAR) Analysis of Estrogen*, Journal of Chemical Information and Computer Sciences, 39 (1999), pp. 164–168.

[3] Gedeck, P. and Willett, P.: *Visual and Computational Analysis of Structure-Activity Relationships in High-Throughput Screening Data*, Current Opinion in Chemical Biology 5 (2001), pp. 389–395.

[4] Graaf, E.H. de, Kok, J.N. and Kosters, W.A.: *Improving the Exploration of Graph Mining Results with Clustering*, in Proc. 4th IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI2007), to appear.

[5] Hanke, J., Beckmann, G., Bork, P. and Reich, J.G.: *Self-Organizing Hierarchic Networks for Pattern Recognition in Protein Sequence*, Protein Science Journal 5 (1996), pp. 72–82.

[6] Izrailev, S. and Agrafiotis, D.K.: *A Method for Quantifying and Visualizing the Diversity of QSAR Models*, Journal of Molecular Graphics and Modelling 22 (2004), pp. 275–284

[7] Kohonen, T.: *Self-Organizing Maps*, Volume 30 of Springer Series in Information Science, Springer, second edition, 1997.

[8] Kosters, W.A. and Wezel, M.C. van: *Competitive Neural Networks for Customer Choice Models*, in E-Commerce and Intelligent Methods, Volume 105 of Studies in Fuzziness and Soft Computing, Physica-Verlag, Springer, 2002, pp. 41–60.

[9] Lameijer, E.W., Kok, J.N., Bäck, T. and IJzerman, A.P.: *Mining a Chemical Database for Fragment Co-Occurrence: Discovery of "Chemical Clichés"* Journal of Chemical Information and Modelling 46 (2006), pp. 553–562.

[10] Lameijer, E.W., Tromp, R.A., Spanjersberg, R.F., Brussee, J. and IJzerman, A.P.: *Designing Active Template Molecules by Combining Computational De Novo Design and Human Chemist's Expertise* Journal of Medicinal Chemistry 50 (2007), pp. 1925–1932.

[11] Mahony, S., Hendrix, D., Smith, T.J. and Golden, A.: *Self-Organizing Maps of Position Weight Matrices for Motif Discovery in Biological Sequences*, Artificial Intelligence Review Journal 24 (2005), pp. 397–413.

[12] National Cancer Institute (NCI), DTP/2D and 3D structural information, `http://cactus.nci.nih.gov/ncidb2/download.html`.

[13] Rhodes, N., Willet, P., Dunbar, J. and Humblet, C.: *Bit-String Methods for Selective Compound Acquisition*, Journal of Chemical Information and Computer Sciences 40 (2000), pp. 210–214.

[14] Roberts, G., Myatt, G.J., Johnson, W.P., Cross, K.P. and Blower Jr, P.E.: *LeadScope: Software for Exploring Large Sets of Screening Data*, Journal of Chemical Information and Computer Sciences 40 (2000), pp. 1302–1314.

[15] Willet, P., Barnad, J.M. and Downs, G.M.J.: *Chemical Similarity Searching*, Journal of Chemical Information and Computer Sciences 38 (1999), pp. 983–996.

[16] Xu, J., Zhang, Q. and Shih, C.-K.: *V-Cluster Algorithm: A New Algorithm for Clustering Molecules Based Upon Numeric Data*, Molecular Diversity 10 (2006), pp. 463–478.

[17] Yan, X. and Han, J.: *gSpan: Graph-Based Substructure Pattern Mining*, in Proc. 2002 IEEE International Conference on Data Mining (ICDM 2002), pp. 721–724.

[18] Zaki, M., Parthasarathy, S., Ogihara, M. and Li, W.: *New Algorithms for Fast Discovery of Association Rules*, in Proc. 3rd International Conference on Knowledge Discovery and Data Mining (KDD 1997), pp. 283–296.