GraphClust: a Method for Clustering Database of Graphs

D. Reforgiato Recupero[1], Rodrigo Gutierrez[2], Dennis Shasha[3]

[1]Dipartimento di Matematica e Informatica

Università degli Studi di Catania

e-mail: diegoref@dmi.unict.it

[2]Biology Department

New York University

Departamento de Genetica Molecular y Microbiologia

P. Universidad Catolica

Santiago, Chile.

e-mail: rg98@nyu.edu

[3]Computer Science Department

New York University

e-mail: shasha@cs.nyu.edu

Abstract

Any application that represents data as sets of graphs may benefit from the discovery of relationships among those graphs. To do this in an unsupervised fashion requires the ability to find graphs that are similar to one another. That is the purpose of GraphClust. The GraphClust algorithm proceeds in three phases, often building on

other tools:

1) it finds highly connected substructures in each graph;

2) it uses those substructures to represent each graph as a feature vector; and

3) it clusters these feature vectors using a standard distance measure. We validate the cluster quality by using the silhouette method. In addition to clustering graphs, GraphClust uses SVD decomposition to find frequently co-occurring connected substructures. The main novelty of GraphClust compared to previous methods is that it is application-independent and scalable to many large graphs.

Index terms - Text clustering, Document vectors, Graph clustering, Graph substructure matching

1 Background

In the last few years, developing algorithms for clustering data represented by graphs has been recognized as a problem in the pattern recognition community [Bunke, 2003]. Nevertheless, graph clustering is still an open problem for two reasons. First, many interesting exact graph matching problems, e.g. subgraph isomorphism, and maximum common subgraph, are NP-complete, so graph clustering algorithms built on top of graph matching can take exponential time. Second, the proper distance metric between graphs is a matter of debate.

In chemistry, the clustering of substances is of central interest as well described in [Brint and Willett, 1987]. The reason is that compounds that are structurally similar to each other and whose nodes have similar properties are likely to exhibit similar properties [Johnson and Maggioara, 2000]. For this reason, an appropriate description of molecule structures and an adequate clustering algorithm are both essential to a good clustering. Such techniques have been applied to the prediction of the physiochemical and biological properties of chemical compounds [Brown and Martin, 1996, Brown and Martin, 1997], the selection of diverse representative compounds and reagent sets for the design of combinatorial libraries [Herpin et al., 2000], compound acquisition selection [Shemetsukis et al., 1995, Menard et al., 1998] and the compilation of screening sets [Engels et al., 2000]. Many methods, often based on fingerprint descrip-

tions, have appeared in the literature [Forgy, 1965]. The hierarchical Ward [Ward, 1963], the nonhierarchical Jarvis-Patrick [Jarvis and Patrick, 1973] and the k-means relocation methods [Downs and Barnard, 2002] are the most popular. Ward's method outperforms the other two in chemical databases [Brint and Willett, 1987, Engels et al., 2000, Forgy, 1965, Wild and Blankley, 2000]. In [Ott et al., 2004] the authors proposed a sequential superparamagnetic clustering approach that correctly clusters datasets composed of structures from seven chemically distinct compound classes.

There is an extensive literature on subgraph searching [Wang et al., 1999, Suciu, 1998]. Most of these methods are designed for specific applications. For example, Daylight [James et al., 2000] is a searching system for molecular databases using fingerprints consisting of bit vectors, where each position is associated to a small path. It outputs all the molecules that contain at least one occurrence of the query.

Other methods for XML databases have been proposed (see for example [xpa, 1999, McHugh et al., 1997, Sheng et al., 1999, Galanis et al., 2001, Shanmuga. et al., 1999]).

In [Messmer and Bunke, 1996] the authors proposed a method which indexes the graphs in a database and computes a graph isomorphism. Both indexing and matching are based on all possible permutations of the adjacency matrices of the graphs.

GraphGrep [Giugno and Shasha, 2002] is a tool to find all the occurrences of a subgraph in a pre-known database of graphs. Its filtering process starts by computing for each graph and for each node all the paths starting at that node up to a certain length. A hash-table associates all the paths with their corresponding indexes.

Frequent subgraph mining has been studied extensively in the data mining community. Mining patterns from graph databases in different domains such as cheminformatics and bioinformatics is challenging since operations within graphs have high time complexity.

Subdue is a system that captures essential structure information from graphs. The Subdue substructure discovery system, [SUB, 1988], discovers repetitive subgraphs in a labelled graph representation by using the minimum description length principle. Subdue has found application to several domains, such as molecular biology, image analysis and computer-aided design.

FFSM, [Huan et al., 2003], is a novel frequent subgraph mining algorithm that employs a vertical search scheme within an algebraic graph framework in order to

reduce the number of redundant candidates. In some experiments, it outperforms gSpan, [Yan and Han, 2002], where the introduction of a lexicographic order among graphs allows the use of a depth first search strategy to mine frequent connected subgraph efficiently.

In order to decrease the size of the output which could be exponential, Spin, [Huan et al., 2004], is a new algorithm that finds only maximal frequent subgraphs: it first retrieves all frequent trees from a general graph database and then reconstructs all maximal subgraphs from the mined trees.

In [Kuramochi and Karypis, 2004] a method for finding frequent connected subgraphs that have a sufficient number of edge-disjoint embeddings in a large sparse graph is presented.

Text retrieval has focussed on the need to locate textual information efficiently (see [Steinbach et al., 2000, Cutting et al., 1992, Berry et al., 2003, Kowalski, 1997]). A classical text searching method [Cutting et al., 1992] involves modelling a text collection as a document-term matrix, and evaluating a document's relevance to a query using a linear algebraic dot product. In a term-document matrix A, $A[i, j]$ gives the number of occurrences of term j in document i. Queries are normally represented as a bit vector over the same set of terms. The similarity between document vectors (the rows of document-term matrices) can be found by their inner product. This corresponds to determining the number of term matches (weighted by frequency) in the respective documents. Singular Value Decomposition (SVD) has been shown to work well for text retrieval over the last fifteen years [Deerwester et al., 1990, Hull, 1994]. The efficiency motivation is clear: large document-by-term matrices have a significant amount of redundant data. Surprisingly, removing this information allows a more precise as well as efficient search. Singular Value Decomposition achieves rank reduction by breaking the matrix A in the product of 3 matrices T, S, D^T which can be truncated to r dimensions to find the most significant "dimensions" in the data. Latent Semantic Indexing (LSI), [Husbands et al., 2001] uses the truncated Singular Value Decomposition (SVD) of the term-document matrix to improve search.

By analogy, *Spectral* methods try to represent the most interesting properties of the input graphs using vectors, thus reducing the graph clustering problem to a problem in a vector space [Luo et al., 2002, Luo et al., 2003, Kosinov and Caelli, 2002]. This

allows a new *spectral* method, closely related to latent semantic indexing, to be used.

This paper presents GraphClust, a simple algorithm for clustering labelled graphs. The algorithm synthesizes several ideas from the previous work in each of its three phases. In the first phase, it finds interesting substructures using Subdue (for sparse graphs) or GraphGrep (for dense graphs). In the second phase, it maps the graphs to feature vectors where the frequency of substructure j in the graph i is stored at $A[i, j]$. After this, the rows of the matrix A are clustered using a standard k-means or a fast approximation algorithm called Antipole. In an optional phase, GraphClust uses latent semantic indexing to pull out interesting common substructures.

In contrast to previous methods ([Johnson and Maggioara, 2000, Ott et al., 2004, Brown and Martin, 1996, Brown and Martin, 1997, Schreiber and Schwobbermeyer, 2005]), GraphClust works with any kind of labelled graph. In contrast to FFSM, gSpan and Spin, GraphClust can use the AllPairShortestPath procedure to find important substructures. Avoiding the isomorphism computation makes GraphClust able to be run on large datasets as shown in 5. Subdue is slower than AllPairShortestPath, but is fast enough for moderate sized applications thanks to techniques developed in [Ketkar et al., 2005] so we offer that as another option.

To summarize our strategy, we synthesize simple algorithms with sophisticated previous techniques to develop a scalable algorithm that can give insights real applications as we show by example in 6. As of this writing, more than 70 researchers worldwide have downloaded our software for applications ranging from sociology to XML clustering.

2 Design

GraphClust assumes that each node of the database graphs has a unique identification number and a label. Edges are unlabelled (for purpose of this paper).

GraphClust deals with either directed or undirected graphs. The substructures can be discovered in two ways:

- by using the AllPairShortestPath algorithm as in [Giugno and Shasha, 2002]: for each graph of the dataset and for each vertex v, all shortest paths whose lengths are at most a parameter l_p are generated from v. Each path is represented by the

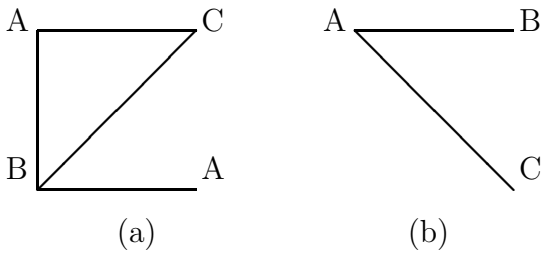

Figure 1: Dataset of two graphs.

Graph (a)		Graph (b)	
Initial Node	Substructures generated	Initial Node	Substructures generated
top-left A	{A,AC,AB,ABA}	A	{A,AB,AC}
B	{B,BC,BA,BA}	B	{B,BA,BAC}
bottom-right A	{A,AB,ABC,ABA}	C	{C,CA,CAB}
C	{C,CA,CB,CBA}		

Table 1: Patterns generated from the dataset of Fig. 1 using AllPairShortestPath with $l_p = 3$.

	C	CA	CB	CBA	A	AB	ABA	B	BAC
graph (a)	1	2	2	2	2	4	2	1	0
graph (b)	1	2	0	0	1	2	0	1	2

Table 2: Matrix A generated from the patterns of Table 1.

sequence of node labels in that path. The set of such sequences represents the substructure around vertex v.

- by using the Subdue substructure discovery system ([SUB, 1988]); in this case, for each graph g of the dataset, Subdue finds common or approximately common substructures of g.

A matrix having a number of columns equal to the number of found substructures and a number of rows equal to the number of the graphs in the dataset is created. Each entry $A[i, j]$ represents the number of times in which the substructure j is contained in the graph i.

If AllPairShortestPath is run, it finds for each vertex all the label sequences corresponding to all paths whose lengths are at most l_p and therefore it creates more columns in the matrix A than Subdue does. Experimentally we have observed that $l_p = 4$ is a good trade-off between time and precision. However, if too many substructures are found when using either the AllPairShortestPath or Subdue, GraphClust considers only the max_sub most frequent, where max_sub is a parameter.

Once the matrix A is built, we cluster its rows. We offer two clustering schemes: the k-means algorithm in which the user chooses the number of clusters k to create; Antipole clustering [Cantone et al., 2004] in which the user chooses a "tightness" measure (an integer value in the range 1 to 4) where the higher the measure the smaller the cluster radius and hence the larger the number of generated clusters. Antipole clustering [Cantone et al., 2004] is much faster than k-means. The metric distance used in both clustering algorithms just described can be either Euclidean distance or inner product distance. Euclidean distance is appealing for applications having a natural geometry. Inner product is better for non-spatial applications such as text-similarity.

In table 2 a matrix obtained from the patterns of table 1 generated by applying the AllPairShortestPath algorithm with $l_p = 3$ to the dataset in Fig. 1 is shown.

Another operation we perform on matrix A is to find closely related substructures. Using singular value decomposition (SVD), the substructure-graph matrix A^T is broken apart into the product of 3 matrices T, S and D^T. Table 3 shows a SVD of the matrix A in table 2. These matrices are truncated to r dimensions with r chosen by the user. Dimensionality reduction reduces the noise present in the substructure-substructure

matrix revealing a robust relationships between the substructures. The substructure-substructure correlation matrix X_r is then computed by multiplying $T_r \times S_r \times (T_r \times S_r)^T$. Table 4 shows the substructure-substructure correlation matrix X_r computed from the matrices T_r, S_r of table 3 reduced for $r = 1$.

$$\begin{pmatrix} -0.20 & -0.17 \\ -0.40 & -0.33 \\ -0.26 & 0.35 \\ -0.26 & 0.35 \\ -0.33 & 0.01 \\ -0.66 & 0.02 \\ -0.26 & 0.35 \\ -0.20 & -0.17 \\ -0.13 & -0.68 \end{pmatrix} \times \begin{pmatrix} 6.8 & 0 \\ 0 & 2.61 \end{pmatrix} \times \begin{pmatrix} -0.89 & 0.46 \\ -0.46 & -0.89 \end{pmatrix}$$

$$T \qquad \times \qquad S \qquad \times \qquad D^T$$

Table 3: Matrices T, S, D^T generated by the Singular Value Decomposition of A^T of Table 2.

$$\begin{pmatrix} -0.20 \\ -0.40 \\ -0.26 \\ -0.26 \\ -0.33 \\ -0.66 \\ -0.26 \\ -0.20 \\ -0.13 \end{pmatrix} \times \begin{pmatrix} 6.8 & 0 \end{pmatrix} \times \left(\left(\begin{pmatrix} -0.20 \\ -0.40 \\ -0.26 \\ -0.26 \\ -0.33 \\ -0.66 \\ -0.26 \\ -0.20 \\ -0.13 \end{pmatrix} \times \begin{pmatrix} 6.8 & 0 \end{pmatrix} \right)^T \right) =$$

	C	CA	CB	CBA	A	AB	ABA	B	BAC
C	2	4	2	2	3	6	2	2	2
CA	4	8	4	4	6	12	4	4	4
CB	2	4	4	4	4	8	4	2	0
CBA	2	4	4	4	4	8	4	2	0
A	3	6	4	4	5	10	4	3	2
AB	6	12	8	8	10	20	8	6	4
ABA	2	4	4	4	4	8	4	2	0
B	2	4	2	2	3	6	2	2	2
BAC	2	4	0	0	2	4	0	2	4

Table 4: Reduced correlation substructure-substructure matrix $X_r = T_r \times S_r \times (T_r \times S_r)^T$ for $r = 1$.

3 Algorithms

It turns out that GraphClust consists of 16 different algorithms broken down along the four binary dimensions described in the section 2. In this section, the algorithms used by GraphClust in the three steps of its main procedure will be discussed in more detail

than in the introduction in order that the reader can understand the tradeoffs involved in their application.

Subdue [SUB, 1988] discovers interesting and repetitive subgraphs in a labelled graph representation using the minimum description length principle; Subdue discovers substructures that compress the original data and represent structural concepts in the data. By replacing previously-discovered substructures in the data, multiple passes of Subdue produce a hierarchical description of the structural regularities in the data. Subdue uses a computationally-bounded inexact graph match that identifies similar, but not identical, instances of a substructure and finds an approximate measure of closeness of two substructures. In addition to the minimum description length principle, other background knowledge can be used by Subdue to guide the search towards more appropriate substructures.

Once the substructures and the matrix A have been created, the clustering is performed by the k-means or Antipole clustering method [Cantone et al., 2004]. In our implementation of k-means, at the first iteration (that is for $t = 1$), the initial k centroids $q_1^1, q_2^1, \ldots, q_k^1$ are computed by using the Gonzalez [Gonzalez, 1985] algorithm to find mutually distant centroids; then, the remaining objects are assigned to a class according to the relation $x_l \in C_j^1$ iff $d(x_l, q_j^1) \leq d(x_l, q_i^1),\ 1 \leq j,\ i \leq k,\ i \neq j$. After each iteration t, new centroids are computed in such a way that the performance index, $\gamma_i = \sum_{x \in C_i^t} |x - q_i^t|^2, i = 1, 2, 3, \ldots, k$, is minimized. This achieves the condition $q_i^{t+1} = \frac{1}{n_i^t} \sum_{\forall x \in C_i^t} x$. If $q_i^{t+1} = q_i^t$, the process finishes, otherwise, the objects are grouped again.

The Antipole clustering [Cantone et al., 2004] algorithm of bounded radius is performed by a top-down procedure starting from a given finite set of points S which checks if a given splitting condition is satisfied. This condition asks for two points whose distance is greater than the radius. If there are no two such points, then splitting is not performed and the given subset is a cluster on which an approximate centroid is then found. Otherwise, a suitable pair of points (A, B) of S called Antipole is generated and the set is partitioned by assigning each point of the splitting subset to the closest endpoint of the Antipole (A, B). As seen in [Cantone et al., 2004] the randomized algorithms used by Antipole clustering makes its construction much faster than k-means's.

4 Complexity

Here is a description of the worst case complexity for the three steps of GraphClust. We will consider two different settings of GraphClust: GraphClust$_{DENSE}$ which uses AllPairShortestPath for substructure construction and the Antipole method for clustering; GraphClust$_{SPARSE}$ which uses Subdue for substructure constructions and k-means algorithm for clustering. The former is preferable in the presence of large dense graphs whereas the latter is preferable in finding the substructures and clustering of sparse graphs.

4.1 GraphClust$_{DENSE}$

Let $|D|$ be the number of graphs in a database D. The complexity of the first step of GraphClust$_{DENSE}$ is $\mathcal{O}(\sum_i^{|D|}({n_i}^3))$, where n_i is the number of nodes of the graph i. The second step has complexity $\mathcal{O}(|D| \cdot |pat| \cdot \sum_i^{|D|}(n_i \cdot m_i^{l_p}))$, where $|pat|$ is the total number of patterns generated and m_i is the number of patterns starting from n_i. Finally, the Antipole clustering has a worst-case complexity of $\frac{\tau(\tau-1)}{2} \cdot |D| + o(|D|)$ in the input size $|D|$, where τ is the bounded radius (see [Cantone et al., 2004] for further details). GraphClust$_{DENSE}$ performs well on large datasets. In Fig. 2 we demonstrate the scalability of GraphClust$_{DENSE}$ on NCI databases up to 50000 molecules. We also report the number of substructures created for each dataset. The graphs in these databases have an average number of 20 nodes; several graphs have up to 270 nodes.

4.2 GraphClust$_{SPARSE}$

The complexity of finding substructures by using Subdue is $\mathcal{O}(\sum_i^{|D|}(\sum_{j=1}^{nsubs}\ (ninst_j \cdot gm_j)))$, where $ninst_i$ is the maximum possible number of non-overlapping instances for substructure j and gm_j is the user-defined maximum number of partial mappings that are considered during a graph match between substructure definition j and a potential instance of the substructure. Details of the Subdue complexity analysis can be found in [Rajappap, 2003]. The complexity of the second step is $\mathcal{O}\ (|D| \cdot |pat| \cdot \sum_i^{|D|}(\sum_{j=1}^{nsubs}(ninst_j \cdot gm_j)))$. Finally the k-means clustering takes time $\mathcal{O}(t \cdot k \cdot |D|)$, with k the number of clusters and t is the number of iterations. Normally, $k, t << |D|$.

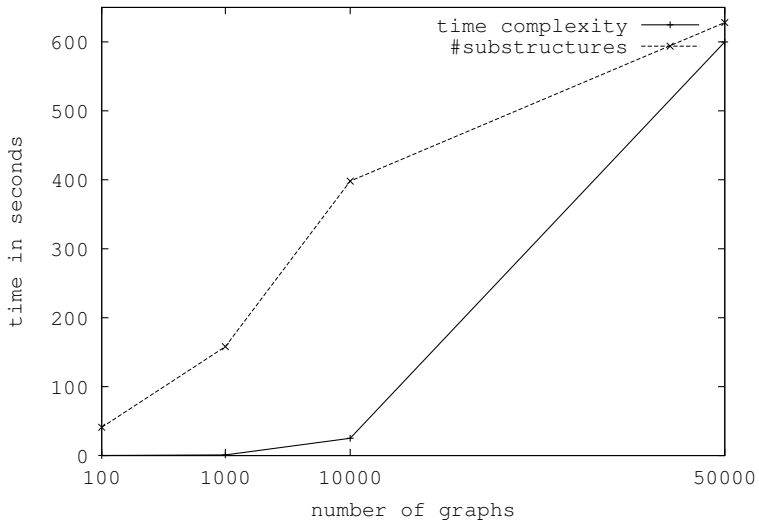

Figure 2: Time vs. number of substructures for the NCI database.

The SVD process complexity applies to both GraphClust$_{DENSE}$ and GraphClust$_{SPARSE}$: $\mathcal{O}\left(|pat|^2 \cdot |D| + |pat| \cdot |D|^2\right)$ so it would not be practical for extremely large datasets. To make the process effective also with large datasets, the SVD considers only the max_sub substructures most interesting substructures (i.e., the most repetitive in the database) where max_sub is a parameter of the algorithm.

5 Performance Studies

We start with a systematic evaluation of the quality of the clusters. The Silhouette method [Bolshakova and Azuaje, 2003] is used as an intrinsic measure of clustering quality. However, we want to study how well the final clustering performed by GraphClust captures the clusters of graphs whose properties are known *a-priori*. For this, we have used an artificial graph generator [Ferro et al., 2005] to create a database containing five different categories of undirected graphs. The five categories include randomly graphs, regular 2D-meshes, regular 3D-meshes, irregular 2D-meshes, irregular 3D-meshes. For each category we have generated 1000 graphs with 30 nodes and 1000 graphs with 80 nodes. The number of edges vary from 50 to 200. Each group of 1000 graphs differs in 7% of the edges. Thus, our artificial dataset contains 10000 graphs. As is known a priori, an optimal clustering creates 10 clusters, each one containing one structural group of graphs. Table 5 depict the global silhouette values GSu for our ob-

tained clustering based on the Antipole Tree data structure (a) and k-means algorithm (b). The number of clusters c varies from 5 to 15. For both clustering algorithms, the substructures have been discovered by using the AllPairShortestPath algorithm fixing the constant $l_p = 4$. In both cases $c = 10$ is suggested as the best clustering configuration for the examined data set and this is also the optimal number of clusters known for when the data set has been created.

c	GSu
5	0.509
6	0.562
7	0.631
8	0.672
9	0.703
10	0.712
11	0.665
12	0.693
13	0.607
14	0.518
15	0.489

(a)

c	GSu
5	0.585
6	0.690
7	0.653
8	0.792
9	0.821
10	0.886
11	0.830
12	0.714
13	0.678
14	0.643
15	0.660

(b)

Table 5: Global Silhouette values for clustering obtained by using GraphClust with Antipole Tree data structure (a) and k-means algorithm (b).

Now, we have asked whether this clustering, not just the number of clusters but the clustering itself, agrees with the *a-priori* classification of the data set. Recall that in the optimal clustering each cluster contains a single structural group and that within that group, the graphs differ by 7% of their edges.

Table 6 shows the similarities in percent between the best clustering obtained in table 5 (that is for $c = 10$) and the optimal clustering. A value $x\%$ for the cluster C_i obtained with GraphClust means that C_i is equal to $x\%$ of the optimal cluster C_i. For most clusters, the correct graphs are found. K-means is slightly better than Antipole in this measure of recall.

To measure the precision of the clustering obtained, a pair of graphs g_1 and g_2 are considered to be consistent in the two clusterings if they are in the same cluster in both cases or in different clusters in both cases. Otherwise they are inconsistent. In table 7, for $c = 10$, the output clustering generated by GraphClust has been compared with the optimal clustering. The value *consistent_value* shows the number of graph pairs that are correctly marked as consistent divided by the total number of graph pairs for the

output obtained from GraphClust.

Clustering Algorithm	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
Antipole Tree	100%	100%	100%	100%	100%	100%	95.14%	66.0%	50%	28.9%
K-means	100%	100%	100%	100%	100%	100%	100%	54.4%	50%	45.6%

Table 6: Similarities in percent between the best clustering found ($c = 10$) in Table 5 and the optimal clustering.

Clustering Algorithm	Number of consistent pairs	Total number of pairs	$consistency\_value$
Antipole Tree	48704861	49995000	0.9741
K-means	48746936	49995000	0.9750

Table 7: Robustness between the best clustering found in Table 5 and the optimal clustering.

6 Inferring Functionality from Gene Coregulation – case study

In this section we use GraphClust to analyze two biological datasets corresponding to mRNA measurements obtained from nitrate treatments of the *Arabidopsis thaliana* plant, the most important model species for plant biologists. Nitrogen-related experiments are significant because they can help improve the use of nitrogen, thus reducing the need for fertilizer and reducing pollution.

Recent studies using microarray technology have demonstrated that the expression of many genes is affected by nitrate [Wang et al., 2000, Wang et al., 2001, Wang et al., 2003]. In [Wang et al., 2004], mRNA (messenger RNA) levels were determined using microarrays in the presence and absence of nitrate in wild-type and in mutant plants that lack the two genes coding for the nitrate reductase enzymes (NR-null). These NR-null plants can not assimilate (i.e. effectively break down) nitrate. Therefore, any change in gene expression observed after nitrate treatments may be due to the action of (intact) nitrate as a signal rather than as a result of the assimilation of nitrate.

To study which genes respond to nitrate as a signal and which groups of genes exhibit similar such responses, we analyzed the expression of all genes in the Arabidopsis genome across a wide variety of nitrate treatments. We selected those genes that both respond to nitrate and whose gene expressions are highly positively correlated with one

another. We have done this for both the NR-null mutant and wild-type, yielding two graphs. We will then used GraphClust to compare the resulting correlation graphs.

To facilitate the biological interpretation of the results at the functional level, we have replaced each gene with the Gene Ontology (GO) terms (there may be several) that characterize it. Thus the node associated with each gene is labeled with that gene's GO terms.

An edge in each graph identifies two genes that have a similar (correlation of 0.5 or better and $p - value \leq 0.01$) response to nitrate. Thus, the pair indicates genes that are sensitive to nitrate and may be linked together (as implied by correlation).

Because the two resulting graphs are large and dense (13867 nodes and 999658 edges for the wild-types and 13428 nodes and 1022692 edges for (NR)-null double mutant), we have used AllPairShortestPath to find graphical motifs (parameter $l_p = 4$).

We found 912 paths of length 3 that were 100-fold more frequent in one genotype as compared to the other. We used the Cytoscape software [Shannon et al., 2003] to visualize the patterns that were 10-fold or more frequent in wild-type as compared to mutant or *vice-versa*. We reasoned that a high relative density of paths associated with a function F in wild-type indicates that genes that perform function F use assimilated forms of nitrate in a closely coordinated fashion. Fig. 3 shows the connected components generated for the functional processes ontology.

The graph suggests a change in the molecular physiology of mutant plants. In wild-type, nitrate treatments cause the co-regulation of genes involved in "regulation of transcription" in addition to processes such as "carbon utilization", "vitamin B12 biosynthesis", and "mRNA splicing". These associations are lacking or much less frequent in the mutant. The central role particularly of "regulation of transcription" suggests that nitrate is necessary for the coordinated expression of transcription factors and their targets, a vital function in plant physiology. Hence, the near absence of "regulation of transcription" co-expression in the mutant suggests that the differences observed between mutant and wild-type are at least partly due to a change in the transcriptional regulatory processes of the plant. Perturbing transcriptional regulation would then influence the expression of genes involved in other processes such as "carbon utilization".

This analysis provided concrete biological hypothesis that can be useful to biologist

for follow-up experimental work:

(1) Nitrate assimilation is critical to coordinated transcriptional responses in normal plants. A follow-up experiment could use a more precise quantitative and temporal analysis of the response to nitrate of the critical transcription factor pairs identified by this analysis.

(2) Altering transcription, a key regulatory factor in plants, would then influence unrelated processes unrelated to nitrate such as "carbon utilization". A follow-up experiment could start with wild type and genetically mutate some of the discovered transcription factors.

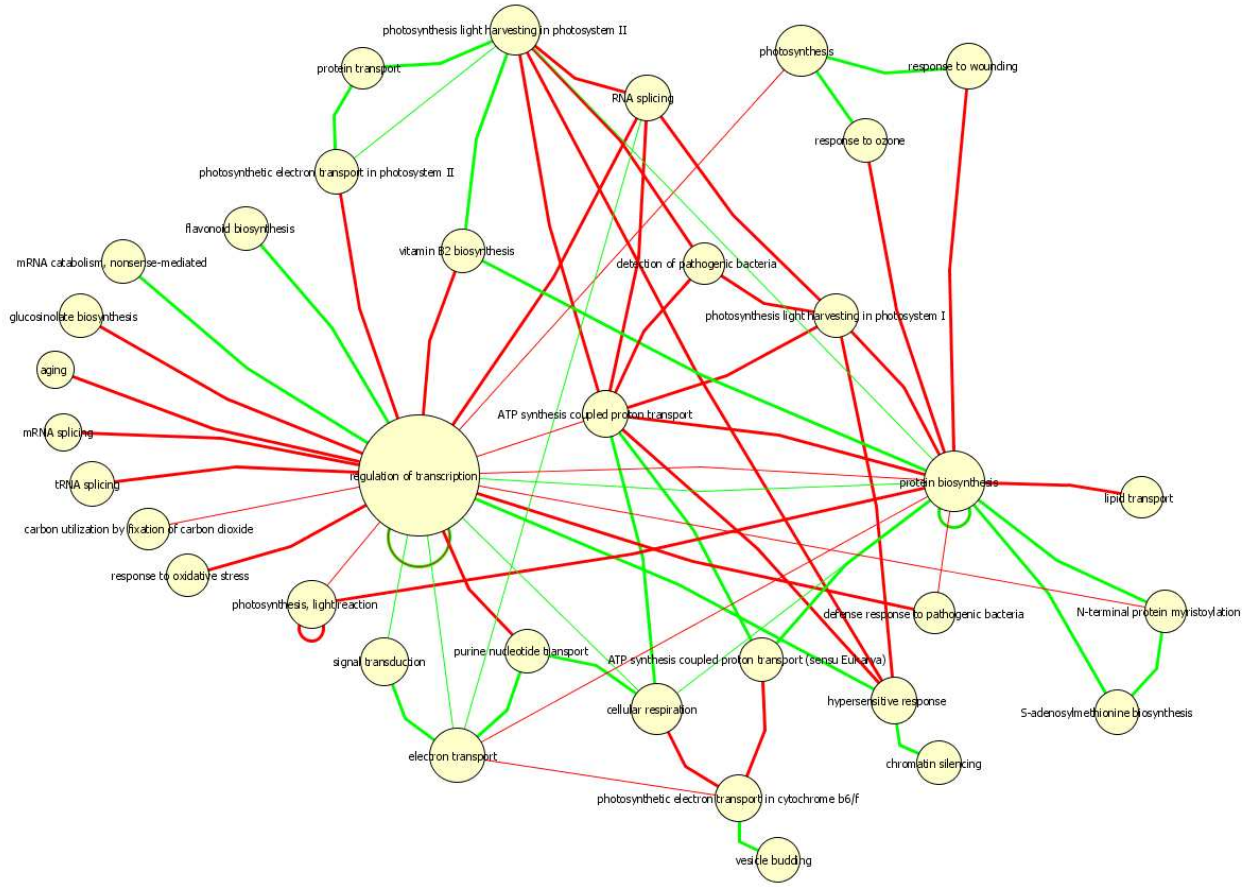

Figure 3: Connected component generated for the functional processes ontology.

7 Summary

Building on previous work, we have proposed a new general method for clustering labelled graphs that comprises three steps (i) identifying interesting substructures, (ii)

clustering the graphs by their substructures and (iii) finding frequently co-occurring substructures pairs.

GraphClust performs efficiently and gives high quality results for both artificial and real data sets across a wide variety of domains.

We show, as an example, suggestive biological results that have been obtained by using GraphClust substructure inference to compare correlations among gene expression values in wild-type and mutant experiments.

For all the experiments we used a Mobile Intel Pentium Processor 2.30GHz and 512MB of RAM equipped with Debian 1.3. The source code has been written in standard ANSI C and is freely available at www.cs.nyu.edu/shasha/papers/graphclust/. As mentioned earlier, over 70 researchers have downloaded it as ofthis writing.

Wall clock times range from minutes to hours in the applications we have tried. For the biological dataset cited, the GraphClust substructure module run in about 1 hour. For the NCI database it took under ten minutes, but the average number of nodes in graphs there is much smaller. With such times for even very large graphs (roughly one million edges in the biological case study), this is a practical tool for a large class of natural and computer science research.

Acknowledgement

This work is based upon work supported by the U.S. National Science Foundation under grants IIS-0414763, DBI-0445666, N2010 IOB-0519985, N2010 DBI-0519984, DBI-0421604, and MCB-0209754 as well as Proyecto Andes (C14060/62). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation This support is greatly appreciated.

References

[SUB, 1988] (1988). http://cygnus.uta.edu/subdue. The SUBDUE Knowledge Discovery System.

[xpa, 1999] (1999). http://www.w3.org/TR/xpath. XML Path Language (XPath).

[Berry et al., 2003] Berry, M., Drmac, Z., and Jessup, E. (2003). Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362.

[Bolshakova and Azuaje, 2003] Bolshakova, N. and Azuaje, F. (2003). Improving expression data mining through cluster validation. *Information Technology Applications in Biomedicine, 2003*, pages 19–22.

[Brint and Willett, 1987] Brint, A. and Willett, P. (1987). Algorithms for the identification of threedimensional maximal common substructures. *J. Chem. Inf. Comput. Sci.*, 27:152–158.

[Brown and Martin, 1996] Brown, R. and Martin, Y. (1996). Use of structure activity data to compare structure-based clustering methods and descriptors for use in compound selection. *J. Chem. Inf. Comput. Sci.*, 36:572–584.

[Brown and Martin, 1997] Brown, R. and Martin, Y. (1997). The information content of 2d and 3d structural descriptors relevant to ligand - receptor binding. *J. Chem. Inf. Comput. Sci.*, 37:1–9.

[Bunke, 2003] Bunke, H. (2003). Graph-based tools for data mining and machine learning. *Proceedings of Machine Learning and Data Mining in Pattern Recognition*, pages 7–19.

[Cantone et al., 2004] Cantone, D., Ferro, A., Pulvirenti, A., Reforgiato, D., and Shasha, D. (2004). Antipole tree indexing to support range search and k-nearest-neighbor search in metric spaces. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(4):535–550.

[Cutting et al., 1992] Cutting, D., Karger, D., Pedersen, J., and Tukey, J. (1992). Scatter / gather: A cluster-based approach to browsing large document collections. *Proc. ACM SIGIR 92*, pages 318–329.

[Deerwester et al., 1990] Deerwester, S., Dumais, S., Landauer, T., Furnas, G., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407.

[Downs and Barnard, 2002] Downs, G. and Barnard, J. (2002). Clustering methods and their uses in computational chemistry. *Reviews in Computational Chemistry, Lipkovitz K., Boyd D.B., Eds; VCH Publishers: New York*, 18:1–40.

[Engels et al., 2000] Engels, M., Thielmans, T., Verbinden, D., Tollenaere, J., and Verbeeck, R. (2000). Cerberus: A system supporting the sequential screening process. *J. Chem. Inf. Comput. Sci.*, 40:241–245.

[Ferro et al., 2005] Ferro, A., Giugno, R., Pulvirenti, A., Recupero, D. R., and Shasha, D. (2005). Blastgen, a graphs generator for graph matching benchmarking. *Preprint*.

[Forgy, 1965] Forgy, E. (1965). Cluster analysis of multivariate data: Efficiency vs interpretability of classifications. *Biometrics*, 21:767–780.

[Galanis et al., 2001] Galanis, L., Viglas, E., DeWitt, D., Naughton, J., and Maier, D. (2001). Following the paths of xml data: An algebraic framework for xml query evaluation. *Submitted*.

[Giugno and Shasha, 2002] Giugno, R. and Shasha, D. (2002). Graphgrep, a fast and universal method for querying graphs. *Proceeding of the IEEE International Conference in Pattern recognition (ICPR)*.

[Gonzalez, 1985] Gonzalez, T. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306.

[Herpin et al., 2000] Herpin, T., Kirk, K. V., Salvino, J., and Yu, S. (2000). Synthesis of a 10000 member 1,5 benzodiazepine-2-one library by the direct sorting method. *J. Chem. Inf. Comput. Sci.*, 2:513–521.

[Huan et al., 2003] Huan, J., Wang, W., and Prins, J. (2003). Efficient mining of frequent subgraph in the presence of isomorphism. *IEEE International Conference of Data Mining (ICDM)*, pages 549–552.

[Huan et al., 2004] Huan, J., Wang, W., Prins, J., and Yang, J. (2004). Spin: Mining maximal frequent subgraphs from graph databases. *KDD*, pages 581–586.

[Hull, 1994] Hull, D. (1994). Improving text retrieval for the routing problem using latent semantic indexing. *In Proceedings of the 17th ACM/SIGIR Conference*, pages 282–290.

[Husbands et al., 2001] Husbands, P., Simon, H., and Ding, C. (2001). On the use of singular value decomposition for text retrieval. *Proc. of SIAM Comp. Info. Retrieval Workshop*, pages 145–156.

[James et al., 2000] James, C., Weininger, D., and Delany, J. (2000). Daylight theory manual-daylight 4.71. *Daylight Chemical Information Systems, www.daylight.com*.

[Jarvis and Patrick, 1973] Jarvis, R. and Patrick, E. (1973). Clustering using a similarity measure based on shared near neighbors. *IEEE Trans. Comput.*, C-22:1025–1034.

[Johnson and Maggioara, 2000] Johnson, M. and Maggioara, G. (2000). Concepts and applications of similarity. *Wiley- New York*.

[Ketkar et al., 2005] Ketkar, N., Holder, L., Cook, D., Shah, R., and Coble, J. (2005). Subdue: Compression-based frequent pattern discovery in graph data. *ACM KDD Workshop on Open-Source Data Mining*.

[Kosinov and Caelli, 2002] Kosinov, S. and Caelli, T. (2002). Inexact multisubgraph matching using graph eigenspace and clustering models. *Proceedings of joint Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition*.

[Kowalski, 1997] Kowalski, G. (1997). *Information retrieval systems: Theory and implementation*. Boston: Kluwer Academic Publishers.

[Kuramochi and Karypis, 2004] Kuramochi, M. and Karypis, G. (2004). Finding frequent patterns in a large sparse graph. *In SIAM International Conference on Data Mining*.

[Luo et al., 2002] Luo, B., Wilson, R., and Hancock, E. (2002). Spectral feature vectors for graph clustering. *Proceedings of joint Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition*, 2396:83–93.

[Luo et al., 2003] Luo, B., Wilson, R., and Hancock, E. (2003). Spectral clustering of graphs. *Proceedings of 4th IAPR-TC15 Graph based Representations in Pattern Recognition*, pages 190–201.

[McHugh et al., 1997] McHugh, J., Abiteboul, S., Goldman, R., Quass, D., and Widom, J. (1997). Lore: A database management system for semistructured data. *SIGMOD Record*, 26:54–66.

[Menard et al., 1998] Menard, P., Lewis, R., and Mason, J. (1998). Rational design and compound selection: Cascaded clustering. *J. Chem. Inf. Comput. Sci.*, 38:497–505.

[Messmer and Bunke, 1996] Messmer, B. and Bunke, H. (1996). Subgraph isomorphism detection in polynomial time on preprocessed model graphs. *Proceedings of ACCV*, pages 373–382.

[Ott et al., 2004] Ott, T., Kern, A., Schuffenhauer, A., Popov, M., Acklin, P., Jacoby, E., and Stoop, R. (2004). Sequential superparamagnetic clustering for unbiased classification of high-dimensional chemical data. *J. Chem. Inf. Comput. Sci.*, 44:1358–64.

[Rajappap, 2003] Rajappap, S. (2003). Interactive biasing in graph-based data mining. *Master Thesis in Computer Science and Engineering.*

[Schreiber and Schwobbermeyer, 2005] Schreiber, F. and Schwobbermeyer, H. (2005). Frequency concepts and pattern detection for the analysis of motifs in networks. *Transactions on Computational Systems Biology, 3(LNBI 3737)*, pages 89–104.

[Shanmuga. et al., 1999] Shanmuga., J., Gang., H., Tufte, K., Zhang, C., Witt, D. D., and Naughton, J. (1999). Relational databases for querying xml documents: Limitations and opportunities. *VLDB Journal.*

[Shannon et al., 2003] Shannon, P., Markiel, A., Ozier, O., Baliga, N., Wang, J., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003). Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Res*, 13(11):2498–2504.

[Shemetsukis et al., 1995] Shemetsukis, N., Dunbar, J., Dunbar, B., Moreland, D., and Humblet, C. (1995). Enhancing the diversity of a corporate database using chemical database clustering and analysis. *J. Comput.-Aided Mol. Des.*, 9:407–416.

[Sheng et al., 1999] Sheng, L., Ozsoyoglu, Z., and Ozsoyoglu, G. (1999). A graph query language and its query processing. *International Conference on Data Engineering (ICDE)*, pages 572–581.

[Steinbach et al., 2000] Steinbach, M., Karypis, G., and Kumar, V. (2000). A comparison of document clustering techniques. *Proc. Text Mining Workshop, KDD 2000*, pages 1–11.

[Suciu, 1998] Suciu, D. (1998). An overview of semistructured data. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 29.

[Wang et al., 1999] Wang, J., Shapiro, B., and Shasha, D. (1999). Pattern discovery in biomolecular data. *New York Oxford, oxford university press edition.*

[Wang et al., 2000] Wang, R., LaBrie, S., and Crawford, N. (2000). Genomic analysis of a nutrient response in arabidopsis reveals diverse expression patterns and novel metabolic and potential regulatory genes that are induced by nitrate. *Plant Cell*, 12:1491–1510.

[Wang et al., 2003] Wang, R., Okamoto, M., Xing, X., and Crawford, N. (2003). Microarray analysis of the nitrate response in arabidopsis roots and shoots reveals over one thousand rapidly responding genes and new linkages to glucose, trehalose-6-p, iron and sulfate metabolism. *Plant Physiol*, 132:556–567.

[Wang et al., 2004] Wang, R., Tischner, R., Gutierrez, R., Hoffman, M., Xing, X., Chen, M., Coruzzi, G., and Crawford, N. (2004). Genomic analysis of the nitrate response using a nitrate reductase-null mutant of arabidopsis. *Plant Phisiol*, 136:2512–2522.

[Wang et al., 2001] Wang, Y., Garvin, G., and Kochian, L. (2001). Nitrate-induced genes in tomato roots: array analysis reveals novel genes that may play a role in nitrogen nutrition. *Plant Physiol*, 127:345–359.

[Ward, 1963] Ward, J. (1963). Hierarchical grouping to optimize an objective function. *J. Am. Stat. Assoc.*, 58:236–244.

[Wild and Blankley, 2000] Wild, D. and Blankley, C. (2000). Comparison of 2d fingerprint types and hierarchy level selection methods for structural grouping using ward's clustering. *J. Chem. Inf. Comput. Sci.*, 40:155–162.

[Yan and Han, 2002] Yan, X. and Han, J. (2002). Gspan: Graph-based substructure pattern mining. *IEEE International Conference of Data Mining (ICDM)*, pages 721–724.