Discovery of Significantly Enriched Subgraphs Associated with Selected Vertices in a Single Graph

Pieter Meysman[1,2], Yvan Saeys[3,4], Ehsan Sabaghian[5,6], Wout Bittremieux[1,2], Yves Van de Peer[5,6], Bart Goethals[1], Kris Laukens[1,2]
[1]Advanced Database Research and Modeling (ADReM), University of Antwerp, Belgium
[2]Biomedical informatics research center Antwerpen (biomina), Belgium
[3]VIB Inflammation Research Center, Belgium
[4]Department of Respiratory Medicine, Ghent University, Belgium
[5]Department of Plant Biotechnology and Bioinformatics, University of Gent, Belgium
[6]Department of Plant Systems Biology, VIB, Belgium
Correspondance e-mail: pieter.meysman@uantwerpen.be

ABSTRACT

In this paper, we present a subgroup discovery algorithm to find subgraphs in a graph that are associated with a given set of vertices. The association between a subgraph pattern and a set of vertices is defined by its significant enrichment based on a Bonferroni-corrected hypergeometric probability value. This interestingness measure requires a dedicated pruning procedure to limit the number of subgraph matches that must be calculated. The presented mining algorithm to find associated subgraph patterns in large graphs is therefore designed to efficiently traverse the search space. We demonstrate the operation of this algorithm by applying it on two biological graph data sets and show that we can find associated subgraphs for a biologically relevant set of vertices and that the found subgraphs themselves are biologically interesting.

Keywords

Subgraph mining, Single graph, Subgroup discovery, Transcriptional Regulatory Networks

1. INTRODUCTION

Graph data has become an important resource in various research fields, from social networks in social sciences to transcription regulatory networks in life sciences. There is thus great interest in developing data mining techniques to extract new and useful knowledge from graph data. A common problem is the discovery of frequent subgraphs in a graph data set. This problem can be defined as the identification of those graph patterns that occur more frequently in a given graph data set than a given support threshold. Many solutions exist to address this specific problem, such as GASTON and gSPAN [25, 18]. However often the interesting subgraphs are not simply those that occur most frequent, but those that are most associated with a specific set of vertices. For example, in a biological graph, specific local subgraphs of interest might be those that can be associated with a disease. An example can be found in figure 1. In this example, we can imagine that the circular vertices are regulatory proteins of one type (such as kinases), the square vertices are regulatory proteins of another type (such as transcription factors), and the edges are regulatory interactions from one protein to another. Four of these proteins, namely the blue circular vertices, are known to be involved in disease A, and that this association might be due to specific regulatory activity (represented by the edges). The potential association of other proteins with this disease are not known, except that most are believed to be irrelevant for disease A. The question is then to find the subgraph motifs that are specific for the disease-associated proteins. In other words, finding those subgraphs that occur more often with the blue vertices than the white vertices. For this problem, identification of all frequent graphs is unsuitable as there is no guarantee that these are in any way associated with any given set of vertices, i.e. our disease-associated proteins. For example, a regulatory interaction between a disease-associated kinase and a transcription factor is frequent in the graph in figure 1, but the same is true for interactions with kinases irrelevant for disease A. However the specific pattern where the transcription factor self-regulates only occurs with the disease-associated kinases, which is represented by the subgraph at the bottom of figure 1. Simply finding the frequent subgraphs is therefore not enough in this case. Instead, this problem requires a test to check if these subgraphs are associated with a subset of the vertices. In addition, it requires a dedicated algorithm to prune the search space. The identification of associated subgraphs can then be used to uncover insights into the characteristics of these selected vertices, for example the regulatory interactions that these proteins are involved in, and they could then be used as input for a classification approach, for example to identify other disease-associated proteins. Finding the patterns associated with a subset of the database has been termed subgroup discovery. Here we present an approach to uncover the subgraphs significantly associated with a given set of vertices. Furthermore we demonstrate its potential by applying it to two real biological graph data sets. As far as

BIOKDD'15 August 10, 2015. Sydney, Australia

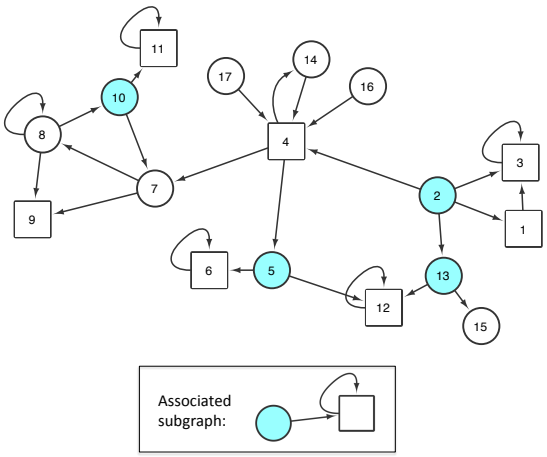

Figure 1: Example graph data set with the vertex shape denotes the label (square or circle) and the selected vertices are colored in blue. The bottom of the figure includes the introduced subgraph associated with the blue vertices.

we are aware, this is the first subgroup discovery approach for subgraph mining in the single graph setting.

2. RELATED WORK

As mentioned in the introduction, there are several subgraph mining algorithms available for a variety of problems. A subgraph can be considered as a pattern or motif of vertices and edges that occurs frequently in a single or multiple graphs. The subgraphs of interest are then typically those that occur more times than a predefined support value. Several subgraph mining algorithms make use of the Apriori principle to prune the search space of frequent subgraphs [7, 12]. Thus if the frequency of a subgraph is found to be below the support threshold, any other subgraph that contains this subgraph will never be able to pass the support threshold and thus need not be considered. Two types of approaches for traversing the search space can be distinguished here; a breadth-first approach where the frequency of all subgraphs of a given size will be enumerated before extending them, and a depth-first approach where the subgraphs are extended along a branch until they no longer exceed the support threshold. The advantage of breadth-first is that it is more efficient in pruning the candidate tree. However depth-first has a much higher memory efficiency as it only keeps the subgraphs within the same lattice in memory. Further one can make a distinction based on the graph data set structure. Some subgraph miners search frequent subgraphs across different graphs so that the frequency is defined as the number of graphs that contain the subgraph, while subgraph miners that search a single graph will define the frequency as the number of occurrences of the subgraph within that graph [8, 17]. Examples for the single-graph setting as is relevant for this paper, include SEuS, SUBDUE and FSG [2, 5, 13]. In each case, the most intensive step of the algorithm is to check the frequency of a subgraph within the larger graph. Indeed subgraph matching is known to be NP complete [3]. Therefore the number of subgraphs to be tested must be kept to a minimum to create an efficient algorithm. This includes pruning the search space but also removal of subgraphs that are identical to each other and thus redundant. A common technique to avoid testing the same subgraph multiple times is to create a canonical representation of each subgraph [25, 13]. Each subgraph is represented by a unique string, vector of edges or an adjacency matrix. The canonical representation is therefore a unique code for this subgraph irrespective of the ordering of the vertices or edges in the graph. Redundant subgraphs can then easily be pruned as they will have the same canonical representation.

The significance of a subgraph in a graph data set can be defined in many different ways. One common definition is that subgraphs are significant if they have a higher frequency than expected from random networks, which have been generated based on a specific network model. Tools to find such subgraphs will often subsample the graphs to speed up the algorithm, and examples include Mfinder, Pajek and FANMOD [9, 1, 24]. While these tools do calculate an enrichment of subgraphs, these are not in any way associated with a specific set of vertices as we propose. There is therefore still no guarantee that even though these subgraphs are more common in the network than randomly expected that they have any relation to a set of selected vertices. As far as we are aware, the only subgraph miner for subgroup discovery is iSubgraph [19]. However this algorithm attempts to find subgroups across graphs, not vertices, based on occurring subgraphs. It is therefore only applicable to a setting with multiple graphs and cannot specifically target any relevant selection of vertices.

There are conceptual similarities between the problems that can be addressed with the presented associated subgraph mining algorithm and pathway or gene ontology enrichment analysis, which is a prevalent procedure within life science research. In the latter case this involves a search for classes or annotations, such as gene ontology terms or pathway assignments, that are significantly associated with a set of biological entities, such as genes or proteins, compared to a given background, typically the entire genome of an organism [11, 14]. As we demonstrate in our case studies, the significant subgraph mining algorithm is able to find the associated subgraphs for a set of biological entities within a biological graph. Thus instead of finding associated annotations, the algorithm finds associated subgraphs which can include relevant biological annotation labels. The presented algorithm can therefore be seen as an extension of pathway or gene ontology enrichment analysis towards the network context. The underlying statistics used in our algorithm are therefore similar to some of those that have been used within pathway analysis [21]. The presented algorithm and accompanying implementation has therefore the potential to be a valuable addition to the toolbox for life science research.

3. DEFINITIONS AND NOTATION

We will start by defining the different terms that will be used in the presented algorithm.

The graph data set subjected to mining is defined as $D = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ the edges, with a vertex labeling function $\ell : V \to L$ where L is the set of all possible labels that can be assigned to vertices. For the purposes of our test cases, we will assume that graph D is directed. Note that the presented algorithm will also work on an undirected graph by assuming that each edge

$e_i = (v_i, v_j) = (v_j, v_i)$. Also the graph D need not be fully connected for the operation of the algorithm.

The set of *selected* vertices is defined as $N \subseteq V$ within the graph, which contains the vertices for which we wish to find associated subgraphs.

A subgraph pattern can be defined as $P = (V_P, E_P)$, where V_P is the set of vertices of P and $E_P \subseteq V_P \times V_P$ the edges of P, with a labeling function $\ell_P : V_P \to L$. We explicitly define a single vertex $v_{PR} \in V_P$ as the *source vertex* of the subgraph pattern P. Note that this source vertex is set when a single edge candidate subgraph is generated and it remains the same vertex when extending the graph. There is no explicit constraint on which vertices may be a source vertex. Theoretically for a single subgraph structure there is a candidate subgraph for each vertex as a source vertex, excluding any symmetrical positions. An instance of a pattern P is defined as the subgraph $G \subseteq D$ if P and G are isomorphic with respect to the edges and vertices labels of each. Thus there exists a mapping function μ which uniquely maps the edges and vertices of G onto P with conservation of the labels, so that $\forall v \in V_P, \ell_P(v) = \ell(\mu(v))$. The corresponding vertex $v_{GR} \in V$, so that $\mu(v_{GR}) = v_{PR}$, is defined as the source vertex of the subgraph instance G. The collection of all source vertices in D for the pattern P, i.e. the collection of all v_{GR}, is denoted by R_G.

Each subgraph pattern P and its instances will be described by a canonical string S, so that isomorphic patterns have the same canonical string. Definition of the canonical string encoding can be found in section 4.3.

3.1 Support measure

The total frequency, $Freq_T$, of a given subgraph pattern is defined as equal to $|R_G|$. Thus this corresponds to the total number of vertices, irrespective if they are part of N or not, that are a source vertex for a subgraph instance of pattern P.

The subgroup frequency, $Freq_S$, of a subgraph pattern is defined as equal to $|R_G \cap N|$. Thus this corresponds to the number of selected vertices in N that are a valid source vertex for the subgraph instance.

Defining the support based on the number of source vertices matching the given subgraph pattern avoids the problem common to single graph subgraph mining where the support does not fit the Apriori criterion due to the occurrence of symmetrical patterns [23]. A vertex can never become a source vertex for a subgraph if it is not a source vertex for any subgraphs that it contains.

3.2 Interestingness measure

If more selected vertices N are source vertices for pattern P than expected when compared to the non-selected vertices $V \setminus N$, we will define the pattern P as associated with, i.e. *enriched* in, the vertices N. This measure must account for the number of selected vertices N and the number of vertices in the graph V that could possibly serve as a source vertex for the subgraph pattern. This is equivalent to the problem in statistics of random sampling without replacement. This concept is best represented by blindly selecting marbles out of jar that contains a fixed amount of red and blue marbles without putting them back. If we draw a fixed number of marbles at random, we can model the probability distribution of drawn red and blue marbles with the hypergeometric distribution. In this case, the jar with marbles are all the vertices in the graph and the two colors represent being either a selected vertex or not. The *drawing* of the vertices is then provided by the source vertices of the subgraph R_G. Thus the probability of observing a $Freq_S = X$ is given by:

$$P(Freq_S = X) = \frac{\binom{|N|}{X}\binom{|V|-|N|}{Freq_T - X}}{\binom{|V|}{Freq_T}} \quad (1)$$

The probability of observing value of $Freq_S$ or higher is then given by the upper cumulative probability:

$$P(Freq_S \geq X) = \sum_{i=X}^{\min(|N|,|R_G|)} P(Freq_S = i) \quad (2)$$

The value $P(Freq_S \geq X)$ thus gives the probability that we find X or more selected vertices as source vertices under the assumption of a uniform distribution of source vertices. This value is commonly known as the *P-value* for the statistical test of enrichment and this is the measure that we will use to define significant subgraphs. The lower the P-value is, the less likely that one would find that number of selected vertices N or more as source vertices R_G if the tested subgraph was independent from these vertices. Therefore if the P-value of a subgraph is sufficiently low, we can claim significant enrichment of the source vertices in the selected vertices and therefore this subgraph patterns can be seen as associated with these selected vertices. The cut-off for the significant P-value is defined as a threshold P_{MAX} so that all subgraph patterns for which $P(Freq_S \geq X) \leq P_{MAX}$ holds true are regarded as interesting and returned. The P-value corresponds to the probability of an associated subgraph being a false positive value and thus can be configured to make the test more stringent. Typical values for the P-value are between 0.1 and 0.01, with respectively a one-in-ten or a one-in-a-hundred chance of a false positive. However as there will be a test performed for each subgraph, we need to correct the P-value for multiple tests. For this reason, we introduce a Bonferroni correction, a well known multiple testing correction, by dividing P_{MAX} by the number of subgraphs tested prior to checking the significance of patterns. This is the most strict multiple testing correction that can be applied, which will limit the search space and the number of significant patterns to exclude a large fraction of the false positives. Other less conservative multiple corrections can also be used in this context, such as the Benjamini-Hochberg correction. Our choice for the Bonferroni correction is to demonstrate that the algorithm is able to find true positive significantly associated subgraphs across several datasets in the most stringent of circumstances.

4. SIGNIFICANT SUBGRAPH ALGORITHM

In this section, we will introduce an algorithm to find significantly associated subgraphs with a set of vertices within a graph. Three important sections of the algorithm are first described before providing an overview of the main algorithm structure.

4.1 Candidate generation

Candidate significant subgraphs are generated according to a depth-first approach. The initial subgraph consists of a single edge and one or two vertices, each of these edges

and vertices is assigned a label from L. This step is repeated twice, once assigning the vertices with the incoming edge as the source vertex, and once for the vertex with the outgoing edge. If $Freq_S$ passes the pruning threshold, the subgraph is extended with a single edge. Several possibilities for edge addition are explored. Firstly, an edge can be added between any two vertices already present in the subgraph. This edge can go from a vertex to itself. In such a case, we speak of a 'self-edge'. Secondly, a single vertex can be added to the subgraph with an outgoing edge to one of the vertices already in the subgraph or an incoming edge from one of the vertices in the subgraph. All possibilities are explored, including different labels for the additional vertex. We will refer to the subgraph pattern that was extended as the 'parent pattern' and the new pattern as the 'child pattern'. As long as the pruning threshold is exceeded, the subgraph pattern is extended until it no longer does so or if it reaches a predefined cutoff on the maximum number of edges, E_{MAX}. After the search space has been entirely exhausted for the initial subgraph, a new combination of vertex and edge labels for a single edge is selected until every possible combination of labels with sufficient frequency have been attempted (see section 4.2).

4.2 Candidate pruning

Candidate significant subgraphs can not be pruned based on enrichment P-value as this metric does not satisfy the Apriori condition. Child patterns can easily achieve a lower P-value than their parents. However the P-value for each subgraph is primarily dependent of $Freq_S$ and $Freq_T$, all other variables remain constant for each subgraph. Intuitively it can be realized that a subgraph must exceed a certain value with $Freq_S$ to achieve $P(Freq_S \geq X) \leq P_{MAX}$. For example, a subgraph where $Freq_S = 0$, i.e. where none of the selected vertices are source vertices, can never be significant. It can be shown that this minimal value of $Freq_S$ always exists and is a value $Freq_{minS} > 0$ so that:

$$P(Freq_{minS}) = \sum_{i=Freq_{minS}}^{|N|} \frac{\binom{|N|}{i}\binom{|V|-|N|}{Freq_{minS}-i}}{\binom{|V|}{Freq_{minS}}} \quad (3)$$

$$P(Freq_{minS}) \leq P_{MAX} \quad (4)$$

$$P(Freq_{minS} - 1) > P_{MAX} \quad (5)$$

Therefore the candidate subgraph tree can be pruned based on $Freq_S > Freq_{minS}$. This is a straightforward frequency count of the number of selected vertices N that are a source vertex of the subgraph pattern P, which satisfies the Apriori condition.

4.3 Subgraph canonical representation

Each subgraph is converted into a canonical representation so that isomorphic subgraphs have the same representation [12]. In this representation, each of the vertices that make up the subgraph is given a unique numeric identifier. Each edge in the representation is then made of two vertex identifiers and two vertex labels if the vertices are labeled. This procedure is based on the canonical representations used by several other graph mining algorithms, which has been shown to avoid redundant subgraph matching [25, 13].

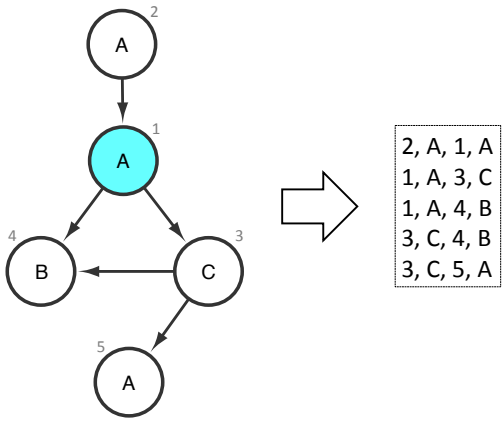

Figure 2: Example subgraph (left) and its canonical representation (right). First each vertex in the subgraph is assigned an identifier (numbers in grey) based on several criteria, such as distance to the source vertex. The source vertex is always assigned the identifier 1. The edges are then sorted based on the identifiers.

The canonical representation of the subgraph patterns has been modified to account for the presence of a source vertex, which is always explicitly set to an id of 1. Thus for child subgraphs, the source vertex will never change, which is critical for the support measure. Each vertex is then numbered based on a sorting criterion, after which each edge is sorted based on the involved vertex identifiers. Non-source vertices are sorted based on, in order, proximity to the source vertex, presence of a self-loop, number of outgoing edges, number of incoming edges and the vertex label. Edges can then be sorted based on lower vertex identifiers. An example of the canonical representation can be found in figure 2.

4.4 Main algorithm

Algorithm 1 ASSOCIATED_SUBGRAPH_MINER. An algorithm for identifying all subgraph patterns (stored in the set $\mathbb{P}$) significantly associated with the selected vertices N in the graph data set D.

Require: graph D, selected vertices N, all possible labels L, maximum number of edges E_{MAX}, maximum p-value threshold P_{MAX}.

```
ℙ = ∅, 𝕊 = ∅, 𝔼 = ∅
Calculate Freq_minS using equation 3
for each label l_1 of L do
    𝔼 ← 𝔼 ∪ (1, l_1, 1, l_1)
    for each label l_2 of L do
        𝔼 ← 𝔼 ∪ (1, l_1, 2, l_2)
        𝔼 ← 𝔼 ∪ (2, l_1, 1, l_2)
    end for
end for
for each single edge pattern P of 𝔼 do
    Subgraph_Search(D, P, N, E_MAX, P_MAX, Freq_minS)
end for
```

In this section we will describe the structure of the main

algorithm supporting the subgroup subgraph mining. The initialization procedure is detailed as algorithm 1. In the first step $Freq_{minS}$ is calculated based on the Bonferroni-corrected P_{MAX} to satisfy equation 4 and 5. Each potential initial single edge subgraph is then generated in line 3-9 of algorithm 1 as detailed in section 4.1 and stored in $\mathbb{E}$. Each of these single edge subgraphs is then subjected in turn to algorithm 2. Here line 1-9 evaluates the current pattern and calculates the P-value for significance. In line 11-17, the algorithm will recursively explore each possible extension of the subgraph as long as it exceeds the support threshold and is not larger than the maximum size. Each new subgraph is transformed into a canonical representation using the procedure detailed in section 4.3, to prune redundant subgraphs by comparison to the stored set $\mathbb{S}$. The end result of the entire algorithm will be the set $\mathbb{P}$ containing all significantly associated subgraph patterns for the given selected vertices N.

Algorithm 2 SUBGRAPH_SEARCH. A recursive procedure to enumerate the significance of the subgraph P and all its child subgraph patterns. The canonical representation of every tested subgraph is stored in the set $\mathbb{S}$ and every significant subgraph is stored in $\mathbb{P}$.

Require: graph D, subgraph pattern P, selected vertices N, maximum number of edges E_{MAX}, maximum p-value threshold P_{MAX} and the minimum support $Freq_{minS}$ for the selected vertices.

1: Calculate $Freq_S$ for pattern P and vertices N
2: **if** $Freq_S < Freq_{minS}$ **then**
3: **return**
4: **end if**
5: Calculate $Freq_T$ for pattern P in graph D
6: Calculate $P(Freq_S \geq X)$ using equation 2
7: **if** $P(Freq_S \geq X) < P_{MAX}$ **then**
8: $\mathbb{P} \leftarrow \mathbb{P} \cup P$
9: **end if**
10: **if** $size(P) < E_{MAX}$ **then**
11: **for** each child pattern C of P **do**
12: $S \leftarrow$ Canonical(C)
13: **if** $S \notin \mathbb{S}$ **then**
14: $\mathbb{S} \leftarrow \mathbb{S} \cup S$
15: Subgraph_Search($D, S, N, E_{MAX}, P_{MAX}, Freq_{minS}$)
16: **end if**
17: **end for**
18: **end if**

5. EXAMPLE DATA SET

To demonstrate the operation of the algorithm and its use, we applied it to the example data set described in the introduction. This data set consists of a graph with 17 vertices, of which 4 are selected for discovering of associated subgraphs. Each vertex carries one of two possible labels, represented by a circle or a square, as can be seen in figure 1. The structure of the graph was designed so that there is exactly one subgraph with one or two edges significantly associated with the selected vertices. The subgraph mining algorithm was run on this example graph with $P_{MAX} = 0.05$ and $E_{MAX} = 2$. The algorithm correctly identified $Freq_{minS} = 4$, i.e. that all of the selected vertices at minimum have to be source vertices for a subgraph for it to pass P_{MAX} following Bonferroni correction. The only single edge subgraph that passed this criterion is one where a circular vertex has a direct edge to a square vertex: $Freq_S = 4, Freq_T = 9$. This single edge subgraph is still found in 5 other non-selected vertices and scores a P-value of 0.053. It therefore did not pass the significance cut-off P_{MAX}. The extension of this subgraph with a self-edge on the square vertex also passed $Freq_{minS} = 4$. However none of the non-selected vertices were source vertices for this subgraph pattern and therefore $Freq_S = 4, Freq_T = 4$. This subgraph is then found to be significant at a P-value of 4.2×10^{-4}.

6. EXPERIMENTAL APPLICATIONS

We demonstrate the use of this algorithm on two real biological data sets to find unknown subgraph patterns that are associated with a set of selected vertices. In the first case, we demonstrate the algorithm on a single connected graph with a large fraction of selected vertices (around 1 in 7). In the second case we apply the algorithm to a large disconnected graph with a very small number of selected vertices. In both cases, these are directed graphs as we wish to demonstrate the ability of the algorithm to uncover relevant graph topology. Further both cases involve transcription regulation networks, where each vertex represents a gene and each edge represents a regulatory interaction. Note that these regulatory interactions are always directed and can include self-edges.

6.1 Duplicated genes in the yeast transcription regulatory network

The first case is a graph representing the transcription regulatory network from baker's yeast as reported by the YEASTRACT database [22]. This graph contains 6 402 genes as vertices, which share 48 080 edges. The selected vertices in this case are genes identified as being retained in duplicate from a recent whole genome duplication [10]. Throughout biological evolution the entire genome (and all of the genes it contains) is on occasion duplicated, so that every gene appears in two copies. Typically the species does not retain both copies but will gradually lose them over the course of many generations. However some genes are known to be retained in duplicate even after several millions of years. There are currently many theories on why such genes are retained, but here we will investigate if there are specific subgraph patterns that can be associated with such genes. The most recent whole genome duplication of yeast occurred around 100 million years ago and 900 duplicate genes remain [10]. The selected set of vertices therefore consisted of 900 gene vertices and E_{MAX} was set to 3, which resulted in $Freq_{minS} = 3$. The labels of the graph were derived from the gene ontology assignments available in Uniprot-GO for yeast [4]. To allow for meaningful patterns, we reduced the total gene ontology annotation to 15 terms by traversing the hierarchical tree of these terms [20], and in case multiple annotations were available for one gene, we choose the term at the lowest level (e.g. nucleotide metabolism is a subset of nitrogen metabolism). Each vertex that did not have a corresponding gene ontology annotation, was assigned a label of 'no annotation'. In a brute force approach without any pruning, we would need to evaluate the enrichment of more than 20 million different subgraphs (the number of possible edge configurations times the number of possible label assignments). However pruning for subgraphs that had at

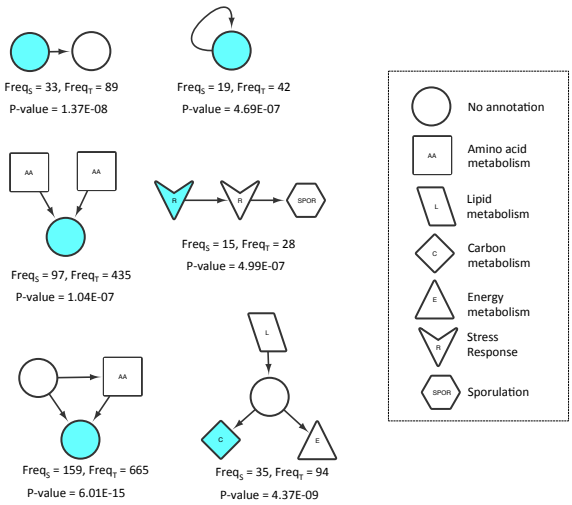

Figure 3: Selected subgraphs with the P-value for enrichment with duplicated genes in the yeast transcription regulatory network. Duplicated genes were set as the selected vertices. Labels are provided by the vertex shape and the subgraph source vertices are marked in blue.

least three selected vertices (duplicated genes) as source vertices reduced the search space to 72 742 subgraphs, of which 5 575 were found to be significantly enriched. Several examples of enriched subgraphs can be found in figure 3. Analysis of the single edge subgraph patterns reveals that 12 were found to be enriched with duplicated genes as source vertices. In all 12 significant single-edge subgraphs, the source vertex had a label of 'no annotation' and had an outgoing edge or a self-edge. This signifies that these duplicated genes are more likely to take on a regulatory role in the network but are also more likely to have no assigned label, i.e. are unannotated by any of the selected gene ontology terms. However in the larger enriched subgraphs (2 or 3 edges), the source vertex was often assigned a label other than 'no-annotation'. These larger subgraphs illustrate nicely that simply because a subgraph is not found to be significant, it does not mean that its child patterns will not either. Further the labels of these larger subgraphs often display well known biological relationships, such as the induction of sporulation under nutritional stress and the relationship between lipid metabolism and carbon/energy metabolism. In summary, these results match current biological hypotheses that these duplicated genes often have an important regulatory role in the transcription networks, in particular with respect to general metabolic processes and stress responses [10, 16].

6.2 Orthologous genes in prokaryotic transcription regulation networks

The second case is a data set featuring the combined transcription regulatory network from seven bacterial organisms with 28 609 vertices and 62 675 edges. While this is a single graph data set, it contains seven independently connected graphs. These graphs were obtained by applying the GENIE3 prediction algorithm [6] on the Colombos expression database [15]. The selected vertices were defined as the 10 homologs of the PhoR transcription factor, i.e. the 10 copies of this gene that exist in these seven networks. This transcription factor is present in all seven species and is involved in the regulation of the phosphate homeostasis, which is of critical importance for both the energy levels of the organism and the creation of new molecular compounds, such as DNA and RNA, in living cells. Application of the proposed subgraph mining algorithm in this manner allows us to track the PhoR vertex across different graphs, representing different species, and characterize the subgraphs that occur consistently throughout. However, this problem cannot be addressed with a multiple graph algorithm, despite spanning several graph datasets. This is because the interestingness, and in turn the support, of the subgraph pattern needs to be defined for the selected vertices and not on the graph level, as some graph data sets have more than one PhoR vertex. As we wish to find the subgraphs that can be consistently found with PhoR, we explicitly set $Freq_{minS} = 10$ so that all PhoR copies must be source vertices of the subgraph to be considered. This value is higher than the one generated through the procedure described in section 4.2, but has no further impact on the operation of the algorithm. The significance measure, namely the hypergeometric P-value, does not change as even with $Freq_S = 10$ it is possible to find subgraphs which are not enriched. No labels were included in this graph data set so that the subgraphs are focused on the graph structure. At $E_{MAX} = 5$, a total of 257 subgraph configurations were checked for enrichment with the PhoR homologs, of which 169 passed the Bonferroni adjusted P-value cut-off. As can be seen in figure 4, only one single edge subgraph was found to be significant, namely the subgraph with an outgoing edge from the source vertex. However larger significant subgraphs do include incoming edges into the source vertex, again demonstrating the notion that a child subgraph can be significantly associated even if the parent subgraph is not. Further as $Freq_S$ is fixed to 10, the P-value is now entirely dependent on $Freq_T$, so that lower values will always correspond to a lower P-value. Other example associated subgraphs include the commonly found feed-forward/back loop in regulatory networks, and subgraphs that include several outgoing edges from the source vertex. These results are in line with the current knowledge on PhoR, namely that it is a transcription factor with many outgoing edges and that it and its targets are frequently under the control of the same global regulators. However the presented algorithm is able to reveal the specific subgraph patterns that are associated with PhoR across many different species, giving an unprecedented level of detail into the evolutionary conservation of its regulatory graph patterns.

7. CONCLUSIONS

In this paper, we have presented a novel type of subgraph mining algorithm for subgroup discovery that can be used to extract specific subgraph patterns related to a set of selected vertices in a single graph. This approach has been shown to be useful on two biological data sets for uncovering novel patterns that are biologically relevant. However the presented algorithm is generic and not limited to a biological setting. It can easily be used, without adaptation, to discover patterns associated with fraudulent behavior in financial graphs, or patterns associated with deviant behavior in social networks. In summary, any instance where the subgraphs of interest are those that can be associated with a specific set of pre-selected vertices.

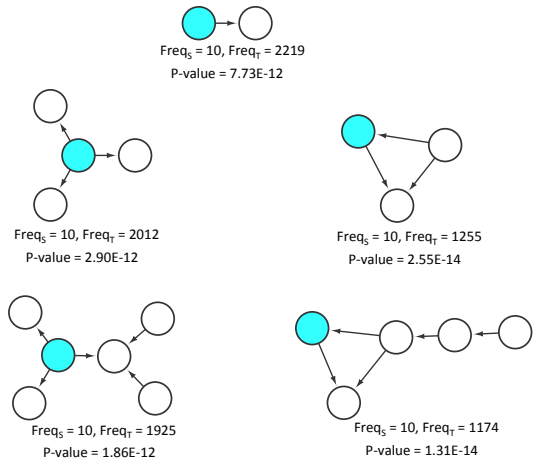

Figure 4: Selected subgraphs with the P-value for enrichment with the 10 PhoR homologs in the prokaryotic transcription regulation networks. PhoR homologs were set as the selected vertices. The subgraph source vertices are marked in blue.

An implementation of the significant subgraph miner for a directed graph with vertex labels can be found on the ADReM website (*http://adrem.ua.ac.be/sigsubgraph*) and includes the featured toy example data set.

8. FUTURE WORK

Several refinements and extensions of the presented algorithm are still possible. Firstly, the time to traverse the search space and evaluate each subgraph can likely be sped up in a number of ways. For example, a breadth-first approach could be used where first the subgraphs of a given size are explored and then combined to efficiently prune the search space for uninteresting subgraphs. However many of the speed improvements will do so in exchange for memory usage and this trade-off will likely be case dependent.

Secondly several extension towards more generic frameworks remain possible. A straight-forward extension of the algorithm would be the introduction of a background set of vertices, as is common with gene ontology enrichment analysis. In this case, the occurrence of the subgraphs of interest in the selected nodes with no longer be compared against the occurrence in all nodes of the graph, but a limited subset. For example, in the dataset presented in section 6.2, the background vertex set could be limited to only transcription factors. In this manner the algorithm would identify those subgraph patterns specific to PhoR with regards to all other transcription factors and not just all other genes. This can be accomplished by modifying the definition of $Freq_T$ so that it only counts those subgraph instances with source vertices from the select background set. However for the P-value to remain statistically valid, the background set must include the original selected vertices. A second extension can be made towards the integration of multi-labeled vertices, as would be of great interest when dealing with complex annotations such as gene ontology or pathways. This can be addressed by modifying the labeling function ℓ so that it can map to multiple labels for a single vertex (or edge). However this will be case-dependent as several decisions must be made with regards to dealing with unlabeled vertices and multi-labeled vertices in a subgraph pattern. A third possible extension could be the inclusion of edge labels, or explicitly accounting for the number of times that a given vertex is a source node for a subgraph. The latter case might be useful in situations where one is not only interested if a vertex is associated with a subgraph, but how many times this subgraph occurs together with this node as well.

Finally, dedicated variants algorithm can designed to solve specific problems. For example, the subgraph search space could be limited to subgraphs of a given type and this in turn could lead to different and more efficient pruning conditions.

9. ACKNOWLEDGMENTS

The authors wish to thank Stefan Naulaerts, Cheng Zhou, Tayena Hendrickx and Aida Mrzic for the helpful discussions and insights. The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Hercules Foundation and the Flemish Government - department EWI. This work was supported by the Fund for Scientific Research - Flanders (FWO-Vlaanderen) project "Evolving graph patterns". YVdP acknowledges support from Ghent University [Multidisciplinary Research Partnership "Bioinformatics: from nucleotides to network"] and from the European Union Seventh Framework Programme [FP7/2007-2013] under ERC Advanced Grant Agreement no. 322739 - DOUBLE-UP.

10. REFERENCES

[1] V. Batagelj and A. Mrvar. *Pajek: analysis and visualization of large networks*. 2004.

[2] D. J. Cook and L. B. Holder. Substructure Discovery Using Minimum Description Length and Background Knowledge. *Journal of Artificial Intelligence Research*, pages 231–255, 1994.

[3] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing - STOC '71*, pages 151–158, New York, New York, USA, May 1971. ACM Press.

[4] E. C. Dimmer, R. P. Huntley, Y. Alam-Faruque, T. Sawford, C. O'Donovan, M. J. Martin, B. Bely, P. Browne, W. Mun Chan, R. Eberhardt, M. Gardner, K. Laiho, D. Legge, M. Magrane, K. Pichler, D. Poggioli, H. Sehra, A. Auchincloss, K. Axelsen, M.-C. Blatter, E. Boutet, S. Braconi-Quintaje, L. Breuza, A. Bridge, E. Coudert, A. Estreicher, L. Famiglietti, S. Ferro-Rojas, M. Feuermann, A. Gos, N. Gruaz-Gumowski, U. Hinz, C. Hulo, J. James, S. Jimenez, F. Jungo, G. Keller, P. Lemercier, D. Lieberherr, P. Masson, M. Moinat, I. Pedruzzi, S. Poux, C. Rivoire, B. Roechert, M. Schneider, A. Stutz, S. Sundaram, M. Tognolli, L. Bougueleret, G. Argoud-Puy, I. Cusin, P. Duek-Roggli, I. Xenarios, and R. Apweiler. The UniProt-GO Annotation database in 2011. *Nucleic acids research*, 40(D1):D565–570, Nov. 2011.

[5] S. Ghazizadeh and S. S. Chawathe. SEuS: Structure Extraction Using Summaries. In *Proceedings of the 5th*

International Conference on Discovery Science, pages 71–85. Springer-Verlag, Nov. 2002.

[6] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts. Inferring regulatory networks from expression data using tree-based methods. *PloS one*, 5(9):e12776, Jan. 2010.

[7] A. Inokuchi, T. Washio, and H. Motoda. An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data. *Princ Data Min Knowl Discov*, 1910:13–23, 2000.

[8] C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(01):75–105, Nov. 2012.

[9] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics (Oxford, England)*, 20(11):1746–58, July 2004.

[10] M. Kellis, B. W. Birren, and E. S. Lander. Proof and evolutionary analysis of ancient genome duplication in the yeast Saccharomyces cerevisiae. *Nature*, 428(6983):617–24, Apr. 2004.

[11] P. Khatri, M. Sirota, and A. J. Butte. Ten years of pathway analysis: current approaches and outstanding challenges. *PLoS computational biology*, 8(2):e1002375, Jan. 2012.

[12] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 313–320. IEEE Comput. Soc, 2001.

[13] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1038–1051, Sept. 2004.

[14] K. Laukens, S. Naulaerts, and W. Vanden Berghe. Bioinformatics approaches for the functional interpretation of protein lists: from ontology term enrichment to network analysis. *Proteomics*, 15(5-6):981–96, Nov. 2014.

[15] P. Meysman, P. Sonego, L. Bianco, Q. Fu, D. Ledezma-Tejeida, S. Gama-Castro, V. Liebens, J. Michiels, K. Laukens, K. Marchal, J. Collado-Vides, K. Engelen, and F. Qiang. COLOMBOS v2.0: An ever expanding collection of bacterial expression compendia. *Nucleic Acids Res.*, 42(1):D649–53, Jan. 2014.

[16] G. Musso, M. Costanzo, M. Huangfu, A. M. Smith, J. Paw, B.-J. San Luis, C. Boone, G. Giaever, C. Nislow, A. Emili, and Z. Zhang. The extensive and condition-dependent nature of epistasis among whole-genome duplicates in yeast. *Genome research*, 18(7):1092–9, July 2008.

[17] S. Naulaerts, P. Meysman, W. Bittremieux, T. N. Vu, W. Vanden Berghe, B. Goethals, and K. Laukens. A primer to frequent itemset mining for bioinformatics. *Briefings in bioinformatics*, 16(2):216–31, Mar. 2015.

[18] S. Nijssen and J. N. Kok. The Gaston Tool for Frequent Subgraph Mining. *Electronic Notes in Theoretical Computer Science*, 127(1):77–87, Mar. 2005.

[19] B. Ozdemir, W. Abd-Almageed, S. Roessler, and X. W. Wang. iSubgraph: integrative genomics for subgroup discovery in hepatocellular carcinoma using graph mining and mixture models. *PloS one*, 8(11):e78624, Jan. 2013.

[20] S. Raychaudhuri, J. T. Chang, P. D. Sutphin, and R. B. Altman. Associating genes with gene ontology codes using a maximum entropy analysis of biomedical literature. *Genome research*, 12(1):203–14, Jan. 2002.

[21] I. Rivals, L. Personnaz, L. Taing, and M.-C. Potier. Enrichment or depletion of a GO category within a class of genes: which test? *Bioinformatics (Oxford, England)*, 23(4):401–7, Feb. 2007.

[22] M. C. Teixeira, P. T. Monteiro, J. F. Guerreiro, J. P. Gonçalves, N. P. Mira, S. C. Dos Santos, T. R. Cabrito, M. Palma, C. Costa, A. P. Francisco, S. C. Madeira, A. L. Oliveira, A. T. Freitas, and I. Sá-Correia. The YEASTRACT database: an upgraded information system for the analysis of gene and genomic transcription regulation in Saccharomyces cerevisiae. *Nucleic acids research*, 42(1):D161–6, Jan. 2014.

[23] N. Vanetik, S. E. Shimony, and E. Gudes. Support measures for graph data. *Data Mining and Knowledge Discovery*, 13(2):243–260, May 2006.

[24] S. Wernicke and F. Rasche. FANMOD: a tool for fast network motif detection. *Bioinformatics (Oxford, England)*, 22(9):1152–3, May 2006.

[25] X. Yan and J. Han. gSpan: graph-based substructure pattern mining. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 721–724. IEEE Comput. Soc, 2002.