MINING REPRESENTATIVE COHESIVE DENSE SUBGRAPHS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Tyler Brazier

In Partial Fulfillment
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

June 2014

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Mining Representative Cohesive Dense Subgraphs

By

Tyler Brazier

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Saeed Salem
Chair

Sameer Abufardeh

Gokhan Egilmez

Approved:

6-23-14	Dr. Kenneth Megel
Date	Department Chair

ABSTRACT

Data mining techniques have an important implication in social and biological network analysis, were we're interested in finding related complexes and communities. A modern paradigm for solving this problem involves finding densely connected interacting members such as bound proteins in a PPI. It's important to also consider the properties of members. In the context of social networks, we might be interested in finding groups of friends of similar age and sharing common interests. This information can lead to better targeting for advertising and suggestions.

In this paper, we introduce an algorithm that can be applied to mining entity-relationship networks. Our approach discovers relevant subnetworks by considering density among entities as well as their similar attribute properties. We apply two distinct methods of forming subnetworks in order to find as many relevant complexes as possible. In addition, we supply techniques for summarization and reduction of nearly-redundant subnetworks in the results.

ACKNOWLEDGMENTS

Firstly, I would like to thank my advisor, Dr. Saeed Salem. Only by his leadership was I able to produce this work. His ambition and passion for the subject inspired my motivation.

I extend special thanks to my undergraduate advisor and good friend Dr. David DeMuth Jr. for being a wonderful guide throughout and after my college experience. His admonition has given me direction and led me to pursue further education and a graduate degree.

I would also like to thank my committee members, Dr. Gokhan Egilmez and Dr. Sameer Abufardeh, for taking the time to evaluate my project and helping me to graduate from the University.

I also want to express my thanks for North Dakota State University and the Department of Computer Science for offering me the opportunity to study at such a great school. The people and the environment have made it a great place to study and learn.

Finally, I thank my fellow students, Kutub Syed and Aditya Goparaju for working with and beside me during my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

G	A graph of vertices and edges
V	The complete set of vertices for a given G
E	The complete set of edges for a given G
U	Commonly used to denote a subset of V
l	Labeling function which maps vertices to real vectors
$\mathbb{R}^d$	A vector of real values with size d
S	A subset of dimensions {1,...,d}
$\|S(U)\|$	The support of a set U
$\rho(U)$	A function denoting the density of a set U
$sim(U, U\prime)$	A function denoting the similarity score of two given subgraphs
θ	Density threshold parameter
α	Weight distribution parameter used for finding similarity scores
β	Similarity threshold parameter
s_{min}	Minimum dimensionality threshold parameter

1. INTRODUCTION

Due to the large amount of data involved in biological and social network analysis, researchers have turned to data mining techniques to filter out meaningful and relevant portions. Social media on the Internet is rapidly gaining popularity [2]. Millions of online profiles exist on popular websites like Facebook, Twitter, Reddit, etc. This trend has a positive impact on marketing; personalized recommendation systems like those used by Amazon or Netflix can take advantage of social network interactions for better results, e.g. making suggestions based on the tastes of your peers [1]. In a similar manner, these techniques can also be used to improve personalized advertising [7]. Network analysis is also often applied on a molecular level. We know that proteins play an essential role in performing vital functions and bond together to form important biological structures such as hemoglobin [4, 23]. Research has shown that by observing protein-protein interaction (PPI) networks, that densely connected subnetworks are often correlated with real and meaningful protein complexes [25]. By making more thorough analyses of PPI networks, we can attempt to make new discoveries of unknown complexes.

In addition to observing interactions, it is often important to consider the attribute profiles of entities [14, 10]. Attribute values in biological networks often represent gene expression data which can give more contextual information regarding an interaction, such as location and time of an incident. In a social network, an attribute profile might correspond to the personal profile of a member and include things such as age, interests, locale, etc. In reality, attribute values can denote almost any kind of additional information about an entity. Including these kinds of data in the mining process has an outstanding benefit.

By combining information about the relational structure of the network and the properties of each member in it, we are able to extract more meaningful results as we take more variables into account. To make this apparent, consider a community such as Facebook, where a single family would often be a densely connected sub-community. If advertisers of Facebook published advertisements based solely on dense community similarity, for example, female product interests expressed by the mother of the family might often cause advertisements for similar products to be displayed to her son. From this scenario, it becomes clear why it is important to consider attributes of members in addition to community structure. The advertising approach could observe the gender difference between the mother and son as well as the age dissimilarity to determine more suitable advertisements to display. Similarly, when mining networks, it is exceptionally helpful to consider the connectivity of the networks in combination with the attribute profiles of the entities.

For the remainder of this paper, we begin by reviewing the related works of several notable graph mining algorithms and give some examples of their processes in Section 2. Section 3 provides some necessary technical definitions to understand the problem. In Section 4 we introduce our own proposed algorithm and explain its operation and benefits, along with examples. In Section 5 we make some comparisons against similar algorithms with experiments and analyze the results. Finally, Section 6 outlines the conclusion and future work for possible improvements to our methods.

2. RELATED WORK

Traditionally, a network is often be represented as a graph where a person in a social network or a protein in a PPI network can be represented as a vertex and an interaction between them is denoted by an edge. In the case of a social network like Facebook, a two-way *friendship* correlates to an undirected edge and in the case of a site like Twitter, for example, one member *following* another is usually denoted as a directed edge. The attribute profiles of entities are often represented as a function mapping of vertices to vectors of real number values. There are a number of existing approaches that use graph mining techniques to discover meaningful patterns in large (and possibly attributed) networks. These approaches are generally divided into two groups: graph partitioning and enumeration based methods.

2.1. Graph Partitioning Methods

Graph partitioning algorithms generally operate in a 'natural' sort of way, where the structure of the graph as a whole has a great influence on how patterns are discovered. In 2004, Newman and Girvan [20] presented several algorithms for graph partitioning which discover edges that exhibit high *edge betweenness* and remove them. Edges with high *betweenness* can be thought of as roads that connect major cities. Intuitively, by removing the links (the roads) between dense communities (the cities) in a graph, you are simply left with the communities, which are your discovered patterns. Betweenness is found by calculating shortest paths between vertices and assigning scores to the edges based on how often they're included in a shortest path. In general, the algorithm operates as follows:

1. The betweenness score of each edge in the graph is calculated.

2. The edge with the highest score is removed.

3. The betweenness scores of remaining edges are again recalculated.

4. Steps 2 through 4 are repeated.

Each time an edge is removed, the result is a new graph in which the dense communities become more apparent. The process can be stopped at any time and the resulting graph can be used to select prominent clusters.

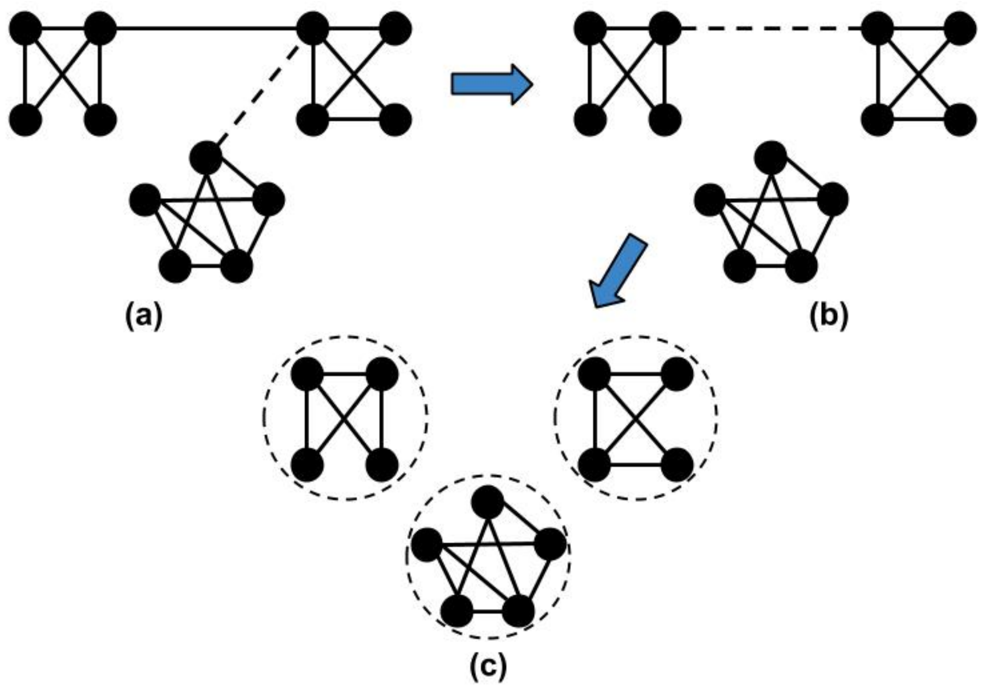
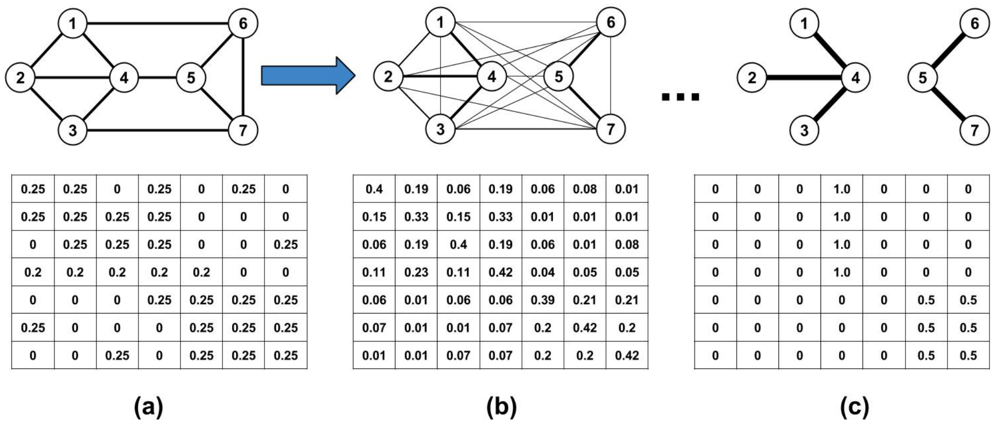


Figure 1. Three stages of Newman's algorithm. In (a), the dashed edge has the highest betweenness score so it will be removed, resulting in the two communities shown in (b). Again, the dashed edge in (b) exhibits the greatest betweenness so it will be removed. (c) shows three communities as the result of step (b).

Another famous work published by van Dongen [27] introduced Markov Clustering (MCL). MCL simulates a random walk from each vertex in the graph to itself and its neighbors. The entire graph is represented as a stochastic matrix and the probability of each walk is an entry M_{ij} in the matrix. Figure 2 (a) shows an example graph and its corresponding right stochastic matrix (self loops are implied). To determine partitioning, the idea is to inflate the probabilities, i.e. make high probabilities higher and low probabilities lower — exaggeration reveals the clusters. MCL accomplishes this through a series of *expansion* and *inflation* operations performed on the matrix. During expansion, the matrix is simply multiplied by itself in order to distribute probabilities more evenly, even between unconnected vertices. Inflation exaggerates probabilities; an inflation operator $r \geq 1$ is given and the expanded matrix is raised to the power r. Then the resulting matrix is normalized; each M_{ij} is divided by the sum of the row. This process continues until a convergence is reached, meaning a sequence of expansion and inflation produces little or no change in the matrix. Figure 2 (b) shows the resulting matrix and corresponding graph illustration after one sequence of expansion and inflation. Here, an inflation operator of $r = 2.5$ is used. Figure 2 (c) shows the final result after six iterations and convergence. Two discovered clusters are clearly visible.



(a)

0.25	0.25	0	0.25	0	0.25	0
0.25	0.25	0.25	0.25	0	0	0
0	0.25	0.25	0.25	0	0	0.25
0.2	0.2	0.2	0.2	0.2	0	0
0	0	0	0.25	0.25	0.25	0.25
0.25	0	0	0	0.25	0.25	0.25
0	0	0.25	0	0.25	0.25	0.25

(b)

0.4	0.19	0.06	0.19	0.06	0.08	0.01
0.15	0.33	0.15	0.33	0.01	0.01	0.01
0.06	0.19	0.4	0.19	0.06	0.01	0.08
0.11	0.23	0.11	0.42	0.04	0.05	0.05
0.06	0.01	0.06	0.06	0.39	0.21	0.21
0.07	0.01	0.01	0.07	0.2	0.42	0.2
0.01	0.01	0.07	0.07	0.2	0.2	0.42

(c)

0	0	0	1.0	0	0	0
0	0	0	1.0	0	0	0
0	0	0	1.0	0	0	0
0	0	0	1.0	0	0	0
0	0	0	0	0	0.5	0.5
0	0	0	0	0	0.5	0.5
0	0	0	0	0	0.5	0.5

Figure 2. Visual example of the MCL algorithm. The entire graph is represented as a right stochastic matrix where each entry value is the probability of travel for a random walk. Through a series of *expansion* and *inflation* operations, eventually a convergence is reached.

2.2. Enumeration Based Methods

Enumeration based algorithms offer better control over what will be included in the result set. For example, several graph mining algorithms take some sort of *connectedness* threshold parameter that all subnetworks that end up in the results must pass. There are several different measures for defining what constitutes a subgraph.

2.2.1. Cliques

The notion of a *clique* was originally created to measure the cohesiveness of social networks [29, 15]. In order for a subgraph to be considered a clique, each vertex in the subgraph must be connected to each other vertex, i.e. it must be a fully-connected subgraph. Derényi et al. [5] introduced the *k-clique percolation method*, which is used to discover densely connected overlapping communities. A k-clique is a clique containing k vertices and two k-cliques are considered neighbors if they share $k-1$ vertices. The k-clique percolation method essentially groups neighboring k-cliques into communities. Figure 3 demonstrates an example of this process with $k = 3$. From the initial graph in (a), first find all cliques of size k (Figure 3 (b)). Then combine neighboring cliques as shown in (c) to create communities. Continue combining neighbors in this way until each community has no more neighbors. The final step (d) shows the resulting graph of two communities with one vertex that belongs in both; this overlap often happens when finding percolated cliques.

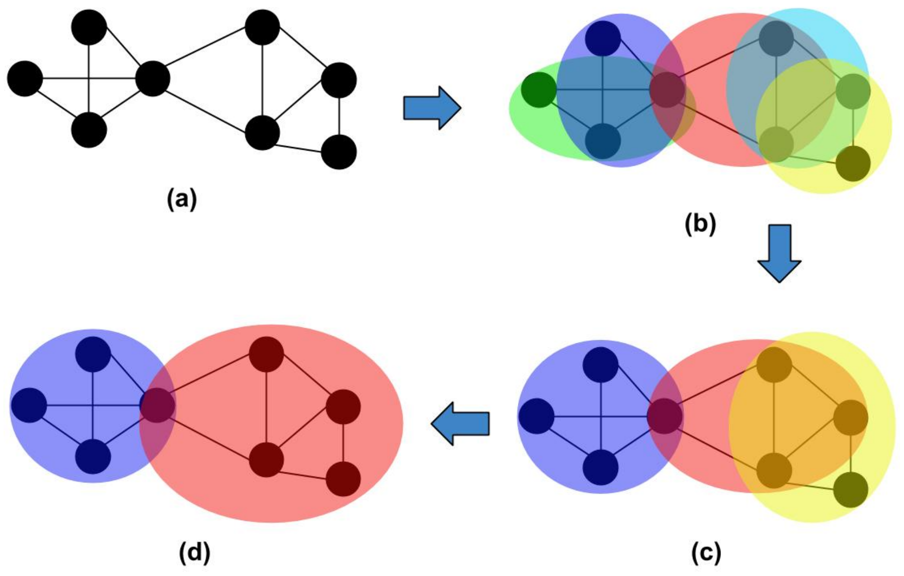


Figure 3. An example process of the k-clique percolation method. First find all k-cliques (here $k = 3$) as shown in (b). Then combine neighboring cliques into communities like (c) until each community has no more valid neighbors. The result contains two communities (d) with one vertex belonging to both.

2.2.2. Quasi-Cliques

Several authors [18, 16, 17] have used the term γ-*quasi-clique* to refer to a subgraph in which the degree of each vertex is γ percent of its greatest possible degree. Naturally, the connectedness of a quasi-clique does not need to be as strong as a clique, and this measure is adjustable by γ. Mining quasi-cliques has traditionally been accomplished by using a set enumeration tree, similarly to itemset mining. Figure 4 gives an example of a small graph and its associated enumeration tree. Each node in the tree represents a set of vertices in the graph as well as a *candidate* set of other vertices in the graph after the last vertex in lexicographical order. In the example, the candidate sets for level 1 of the tree are shown and underlined. It is these candidate vertices that the tree node will expand with to produce the next level in the tree. For example, in Figure 4, {B, C} has the candidate set of {D, E} so {B, C} will branch twice — into {B, C, D} and {B, C, E}. Generally, a *min_size* parameter is given in order to exclude unnecessarily small patterns (of size $<$ *min_size*) from the results. A major problem that occurs, unlike in itemset mining, is that the enumeration tree is not *anti-monotone* — if one pattern in the tree is not a quasi-clique, there is no guarantee that none of its children will be. This makes it difficult to apply pruning strategies to skip enumerating over unnecessary branches of the tree as is done in itemset mining. This problem becomes much more hindering for large graphs since without pruning, the search space is equal to the power set of vertices in the graph. However, there are several other pruning strategies that have been developed to address this issue [21, 17, 13]. Two of them, introduced by Pei et al. [21], will be discussed here;

1. Since each quasi-clique in the results must have more than *min_size* vertices, the degree of a single vertex in any quasi-clique cannot be less than $min\_degree = \lceil \gamma(min\_size - 1) \rceil$. All vertices with degree less than this threshold can be safely

pruned. This can be observed in the example in Figure 4 where vertex {A} has been pruned since the degree of {A} is $1 < \lceil 0.6(3-1) \rceil = 2$ when $\gamma = 0.6$ and $min\_size = 3$.

2. The maximum diameter of a γ-quasi-clique can be calculated with respect to γ, e.g., when $\gamma \geq 0.5$ then the maximum diameter is 2. Figure 4 shows an example case; notice how the candidate set for vertex {B} does not include {F} and {G} since the shortest path from {B} to each of them is longer than 2 when $\gamma = 0.6$.

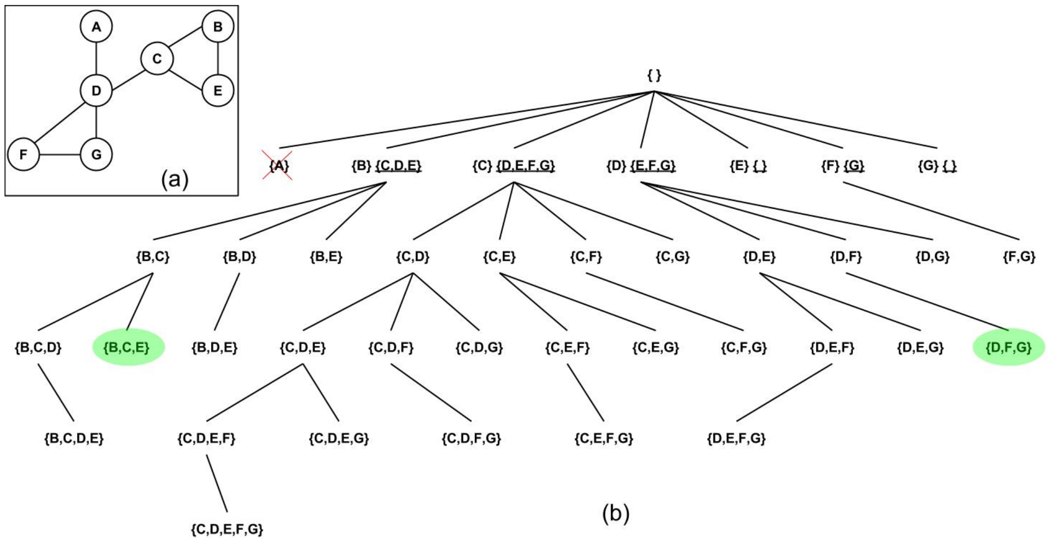


Figure 4. Example graph (a) and its enumeration tree (b) for mining γ-quasi-cliques. Here, $\gamma = 0.6$ and $min\_size = 3$. The tree node for vertex {A} and its entire subtree can be pruned according to pruning strategy 1. Note that {B} does not include in its candidate set vertices which can be reached on a path of size ≤ 2 as per strategy 2. In this example, the two discovered γ-quasi-cliques happen to also be cliques (they are circled in the figure).

As mentioned in the introduction, it is often very important to consider additional information about entities when graph mining. Gunnemann et al. [10] proposed a solution to address this issue with their algorithm GAMer. Their method combines γ-quasi-clique subgraph discovery with subspace clustering in order to find dense communities that also share similar attributes. They define an attributed graph $G = (V, E, l)$ where V is the set of vertices in G and E is the set of edges. l is a labeling function $l : V \rightarrow \mathbb{R}^d$ which maps a vertex to a real vector in d dimensions. In other words, each entity has additional data associated with it which is represented by a vector in d-dimensional space. See Figure 6 for a visual representation. For example, in a simple social network, the collection of subspaces may be something like $(age, sex, marital\_status)$ and one vertex's vector of associated attributes be $(24, Male, Single)$. In order to discover coherent patterns, two more threshold parameters must be given: t is amount by which two attributes can differ but still be considered similar, s_{min} is the minimum number of similar dimensions a quasi-clique must share in order to be a coherent quasi-clique. For a set of vectors $X = \{x_1, ..., x_d\}$ where $\forall x_i \in X : x_i \subseteq \mathbb{R}^d$ and dimensions $S \subseteq \{1, ..., d\}$, they define (X, S) to be a subspace cluster if

- $\forall i \in S : \forall x_1, x_2 \in X : |x_1[i] - x_2[i]| \leq t$
- $\forall i \in \{1, ..., d\} \backslash S : \exists x_1, x_2 \in X : |x_1[i] - x_2[i]| > t$

GAMer builds a γ-quasi-clique enumeration tree as previously discussed but at each node in the tree, attribute profiles of the current vertices are compared using subspace clustering. Fortunately, subspace clustering is an anti-monotone process; if at any point in the tree, the number of similar dimensions $|S|$ falls below s_{min}, the current node and all subtrees can be pruned.

2.2.3. Dense Subgraphs

An issue with the quasi-clique definition is that *every* vertex in the quasi-clique must have at least a certain degree. In real life, data is noisy and communities can often contain members that do not quite meet the minimum degree requirements. In 2007, Uno [26] introduced an algorithm for mining dense modules using the traditional *density* definition. In a θ-dense subgraph, the ratio of the number of edges to the total possible number of edges is at least $0 \leq \theta \leq 1$, where θ is a user-defined minimum threshold. This definition allows for more flexibility than the quasi-clique definition because it observes the density of the pattern as a whole rather than concerning each individual member. Similarly to γ-quasi-clique discovery, dense subgraph enumeration is not anti-monotone. However, Uno [26] presented a method for efficiently enumerating dense subgraphs and later contributed development of the Dense Module Enumeration (DME) algorithm [9]. DME mines maximal dense subgraphs (or *modules*) from weighted networks. They provide the definition for the weighted degree of a vertex as the sum of its edge weights. For example, in Figure 5, the weighted degree of C = 0.6. In addition, they define a *module density* equation $\rho(U)$ which amounts to the sum of the edge weights in the module U divided by the number of possible edges in the module. Formally,

$$\rho(U) = \frac{\sum_{i,j \in U, i<j} w_{ij}}{|U|(|U|-1)/2}$$

where w_{ij} is the weight of the edge between i and j. The algorithm takes advantage of a property for these dense graphs which states that the density of a module does not increase when a vertex added to the module has weighted degree that is no larger than the weighted degree of each other vertex already in the module; i.e. if $v \in U$ is a node with minimum wighted degree in U: $\forall u \in U : deg_U(u) \geq deg_U(v)$. Then, $\rho(U \backslash \{v\}) \geq \rho(U)$. Inversely, it is also true that removing a vertex with minimum weighted degree in U does not decrease the density of U. They provide a proof for this property in [26, 9] and explain how they are able to leverage it in order to prune unnecessary branches in the enumeration tree. With this knowledge, patterns can be discovered in such a way that, as the enumeration tree is expanded from top to bottom, module sizes are increasing while their densities are guaranteed to be decreasing or remaining the same. Therefore, if a module with density less than a given threshold θ is found in the tree, we can stop extending it since none of its children can pass the threshold. Before running the algorithm, the vertices in the graph must be given a strict order. The example in Figure 5 uses an ordering of $ord(A) < ord(B) < ord(C) < ord(D) < ord(E)$. Then, the procedure begins with the empty set and builds the enumeration tree. Let U be the set of vertices at the current node in the tree and $Z = V \backslash U$ be the remaining vertices in the graph that are not in U. At each stage in the enumeration, a branch of the tree is extended with $z \in Z$ to produce $U' = U \cup \{z\}$ if one of the following conditions are met:

- The weighted degree of z w.r.t. U' is strictly less than each other vertex in U.
- The weighted degree of z w.r.t. U' is equal to the weighted degree of each other vertex in U **and** the order of z is less than the order of each other vertex in U.

More formally,

$$\forall u \in U : (deg_{U'}(z) < deg_{U'}(u)) \vee (deg_{U'}(z) = deg_{U'}(u) \wedge ord(z) < ord(u))$$

Figure 5 shows an example graph and the associated enumeration tree. DME traditionally mines maximal patterns, which are circled in the image. The pruned branches are crossed out.

Similarly to GAMer, DME additionally includes limited subspace clustering of the attribute profiles of vertices. However, the data in feature vectors can only be represented as integer values. In that same year, Moser et al. [19] introduced their algorithm CoPaM, which discovers cohesive maximal dense patterns of vertices with real attribute values. The ability to process this real data is an important feature for many network mining algorithms.

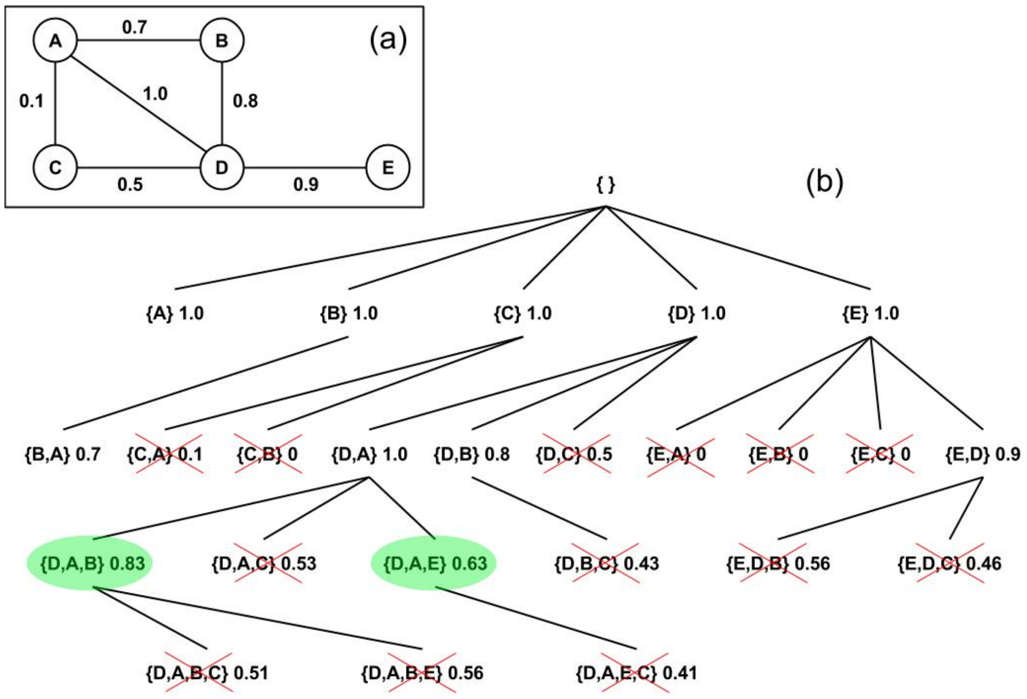


Figure 5. Weighted graph (a) and its DME enumeration tree (b). For this tree, $\theta = 0.6$ and the order of vertices are lexicographical. Crosses show which branches are able to be pruned. The discovered maximal patterns have been circled.

3. PROBLEM DEFINITION

In this section, we give some necessary definitions which are used to better understand the problem. Similarly to all of the aforementioned related approaches, we represent a network as a graph.

Definition 1. *The graph $G = (V, E, f)$ is an unweighted, attributed graph with no self-loops. $V = \{v_1, ..., v_n\}$ is the set of vertices in G, $E \subseteq V \times V$ is the set of edges, and $f : V \to \mathbb{R}^d$ is a function that maps a vertex to a real vector in d dimensions. We will use the vector to represent the attributes of entities. The number of vertices and number of edges in G are denoted as $|V|$ and $|E|$ respectively. As an example, refer to Figure 6; here, V = {Tim, Sam, Ben, Jay, May}, E = {{Tim,Sam},{Tim,Jay},{Jay,Ben},...}, and f(Tim) = (0.5, 1.0, -1.9).*

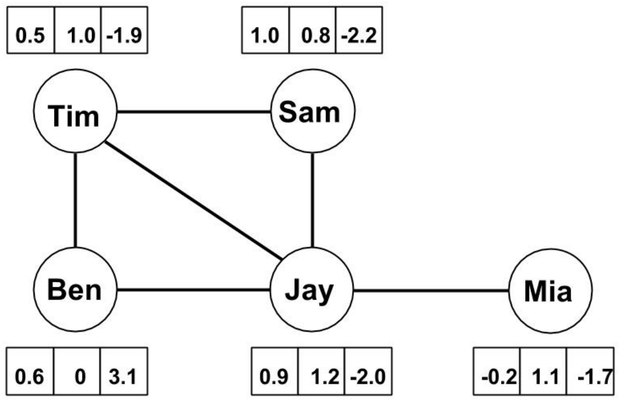


Figure 6. An example of an attributed graph with three subspaces.

Definition 2. *We define the density property (denoted as ρ) of a subgraph U similarly to the definition provided by Uno [26]:*

$$\rho(U) = \frac{2|E|}{|U|(|U| - 1)}$$

In other words, the density of a subgraph U is equal to the number of its edges divided by the number of total possible edges in U. In the case of Figure 6, *ρ({Sam, Jay, Mia}) = $\frac{2}{3} = 0.66$. The density of a single vertex is always 1. Intuitively, a subgraph with many edges will have a higher density value than a subset with fewer edges for the same set of vertices.*

Definition 3. *To consider attribute profiles, we define the notion of a subspace cluster. Given a set of vectors $X = \{x_1, ..., x_d\}$ where $\forall x_i \in X : x_i \subseteq \mathbb{R}^d$, a set of dimensions (subspaces) $S \subseteq \{1, ..., d\}$, tolerance threshold t, and minimum dimensionality threshold s_{min}, (X, S) forms a valid subspace cluster if:*

- $\forall i \in S : \forall x_1, x_2 \in X : |x_1[i] - x_2[i]| \leq t$
- $\forall i \in \{1, ..., d\} \backslash S : \exists x_1, x_2 \in X : |x_1[i] - x_2[i]| > t$
- $|S| \geq s_{min}$

The tolerance threshold t is the maximum amount two attribute values in a single subspace can differ by in order to still be consider similar. The dimensionality threshold s_{min} is the minimum number of similar dimensions a set of vectors can have in order to form a valid subspace cluster. The second condition ensures that the cluster is maximal, i.e., there are no more dimensions outside of S in which two attributes are similar. Referring to Figure 6,

$$f(Jay) = (0.9,\ 1.2,\ -2.0)$$

$$f(Mia) = (-0.2,\ 1.1,\ -1.7)$$

Given threshold parameters $t = 0.5$ and $s_{min} = 2$, for example, these two vectors will make up a valid subspace cluster with similar attributes in dimensions {2, 3}.

Definition 4. *Additionally, we also introduce the notion of support. The support of a subgraph U, denoted as $|S(U)|$, is equal to the maximum number of similar subspaces for all attribute vectors associated with U. Formally,*

$$|S(U)| = |S| : \forall i \in S : \forall u_1, u_2 \in U : |f(u_1)[i] - f(u_2)[i]| \leq t$$

For example, the support $|S(\{Jay,\ Mia\})| = |\{2, 3\}| = 2$. We will use this definition when discovering closed subgraphs.

Definition 5. *A coherent dense subgraph $U \subseteq V$ is an induced subgraph of G which satisfies each of the following conditions.*

1. *Each vertex $u \in U$ is connected to at least one other vertex in U.*
2. *Given a density threshold parameter $0 < \theta \leq 1$, the subgraph U has a density value $\rho(U) \geq \theta$.*
3. *The attribute profiles of all $u \in U$ form a subspace cluster given a set of dimensions S, and threshold parameters t and s_{min}, i.e.*

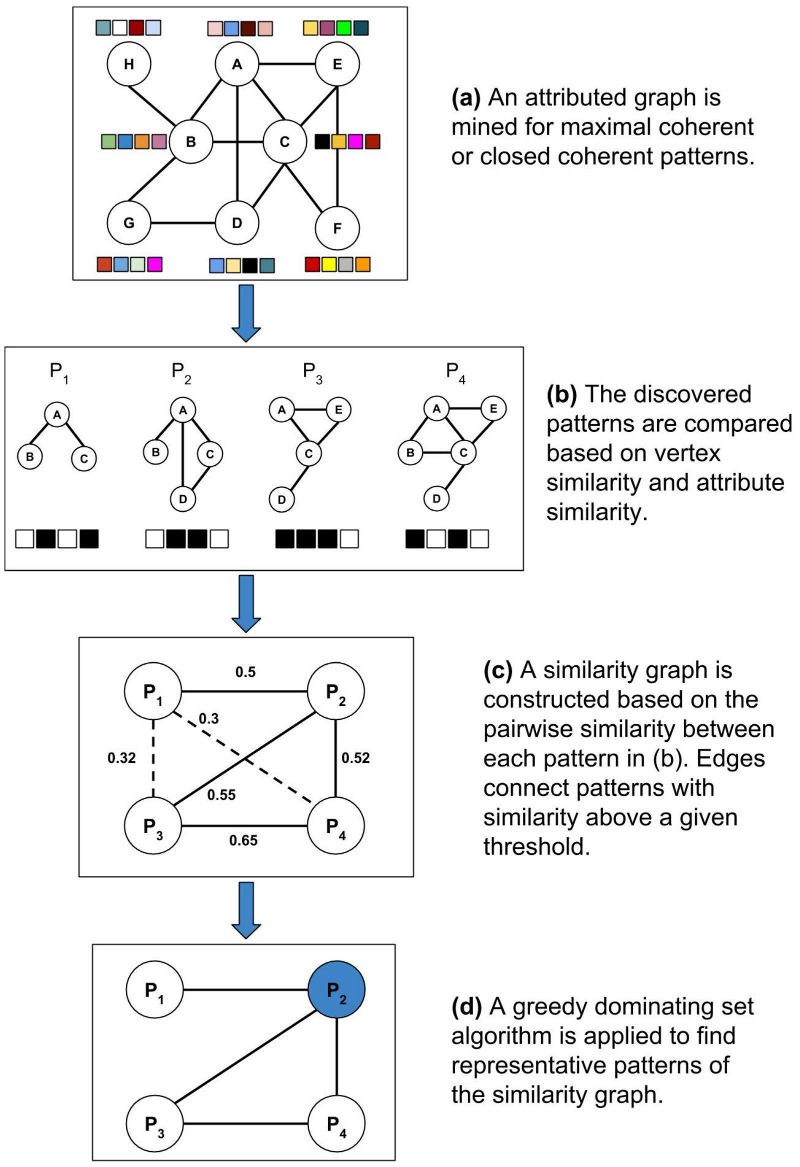
$$\forall u_1, u_2 \in U : \forall i \in S : |f(u_1)[i] - f(u_2)[i]| \leq t, |S| \geq s_{min}$$

In Figure 6, the subgraph $U = \{Sam,\ Jay,\ Mia\}$, for example, forms a coherent subgraph with given thresholds $\theta = 0.6$, $t = 0.5$, and $s_{min} = 2$ in dimensions $S = \{2, 3\}$.

4. PROPOSED ALGORITHM

In this section we introduce our algorithm **Co**hesive **De**nse **N**etwork **E**numeration (CoDeNE). As the name suggests, the algorithm discovers cohesive dense structures in a network — that is, dense patterns with similar attribute profiles. CoDeNE operates on an attributed graph in the form described by Definition 1. In general, the algorithm performs a reverse search method to discover *closed* dense and *maximal* dense coherent patterns with real attribute values. In addition, we provide some post-processing methods to reduce redundancy in the results and to summarize similar communities. Figure 7 gives a visual overview of CoDeNE.

We adopt a pattern mining approach similar to the method described in [9]. Algorithm 1 shows pseudo code for our pattern discovery process. The recursive function builds an enumeration tree like the example in Figure 5. Note that we only consider tree node expansion if the three conditions given in Definition 5 are met (Algorithm 1, line 8). The child of each node in the tree is obtained (Algorithm 1, line 13) in the same manner as [9] and described previously in Section 2.2.3. The result is the set of maximal coherent patterns M and closed coherent patterns C.



(a) An attributed graph is mined for maximal coherent or closed coherent patterns.

(b) The discovered patterns are compared based on vertex similarity and attribute similarity.

(c) A similarity graph is constructed based on the pairwise similarity between each pattern in (b). Edges connect patterns with similarity above a given threshold.

(d) A greedy dominating set algorithm is applied to find representative patterns of the similarity graph.

Figure 7. Overview of CoDeNE.

4.1. Finding Maximal Coherent Patterns

According to Definition 5, a coherent subgraph is any subgraph that can satisfy the three conditions. However, a singular vertex is, by itself, completely dense by definition and has absolute similarity among its own attributes. Also, a pattern of two vertices with a single edge has a density of 1, for example. It needs only to satisfy the third condition of Definition 5 in order to be considered a coherent subgraph. It is obvious that we need a way to keep these kinds of unmeaningful subgraphs out of our result set. A common solution is to mine for the *maximal* patterns. A subgraph is considered maximal if it has no direct superset with a density that satisfies the density threshold condition. For example, in Figure 8, $\rho(\text{a}) = \frac{5}{6} = 0.833$ and $\rho(\text{b}) = \frac{6}{10} = 0.6$. If (b) is the only superset of (a) and $\theta = 0.7$ then graph (a) is maximal since it has no dense superset.

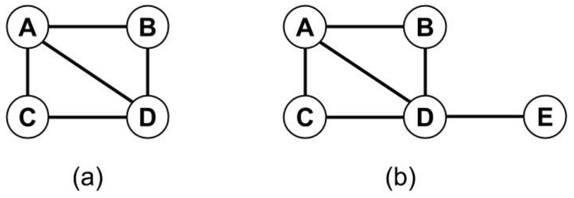


Figure 8. An example subset. (a) is a subset of (b). The density of (a) $\rho(\{\text{A,B,C,D}\}) = \frac{5}{6} = 0.833$ and the density of (b) $\rho(\{\text{A,B,C,D,E}\}) = \frac{6}{10} = 0.6$. If (b) is the only superset of (a) and $\theta = 0.7$, then (a) is maximal subgraph.

Incorporating attribute profiles into the maximal subgraph discovery process allows us to find more meaningful *coherent* maximal dense subgraphs. In order for a subgraph $U \subseteq V$ to be considered coherent maximal it must meet each of the following conditions:

1. It is a maximal dense subgraph, i.e.,

$$\nexists Z : U \subseteq Z, \rho(Z) \geq \theta$$

2. U is a coherent subgraph according to Definition 5.

4.2. Finding Closed Coherent Patterns

As is often the case in itemset mining, we might also be interested in discovering *closed* patterns, as some important clusters that are not maximal would not be included in the results of the previous algorithm. For this reason, we developed

CoDeNE with the ability to additionally find coherent closed patterns. Similarly to itemset mining, we define a set of vertices in a graph to be coherent closed if there is no direct superset of vertices with the same *support*. The support of a subgraph U (denoted as $|S(U)|$) is is equal to the number of similar subspaces for all attribute vectors associated with U (Definition 4). Formally, we say a subgraph $U \subseteq V$ is coherent closed if

$$\nexists Z : U \subset Z, |S(U)| = |S(Z)|$$

Figure 14 shows the result of mining the example graph from Figure 6 with CoDeNE using parameters $\theta = 0.6$, $s_{min} = 2$, and $t = 0.5$.

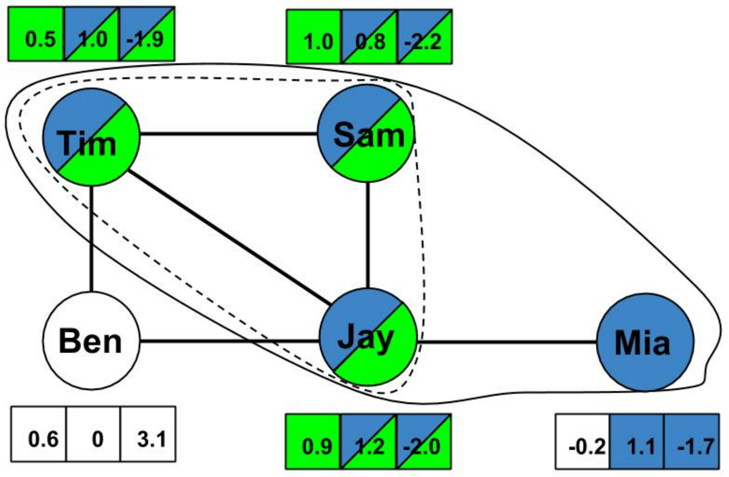


Figure 9. Closed coherent and maximal coherent subgraphs example. With thresholds $\theta = 0.6$, $s_{min} = 2$, and $t = 0.5$, the pattern {Tim, Sam, Jay} is a closed coherent subgraph because there is no superset with the same support. {Tim, Sam, Jay, Mia} is closed *and* maximal coherent since it does not have a superset that satisfies the threshold conditions.

The algorithm has discovered two patterns. The first, {Tim, Sam, Jay}, is a closed coherent pattern with a density of 1 and support of 3. It was found to be closed since its only two direct supersets, namely {Tim, Sam, Jay, Ben} and {Tim, Sam, Jay, Mia}, have support of 1 and 2 respectively. Note that Ben does have one similar attribute but since $s_{min} = 2$ he will not be included in the valid pattern. The second resulting cohesive pattern {Tim, Sam, Jay, Mia} is closed *and* maximal with a density of 0.66 and support of 2. Its only direct superset, which is the whole graph, has a support of 0 and a density of 0.6. Although this superset still meets the density criteria, it does not have enough similar attribute subspaces to be considered maximal coherent. As is the case in itemset mining, the set of maximal patterns is always a subset of the set of closed patterns. This small example provides a good demonstration of a case where it is important to additionally mine for closed subgraphs. The non-maximal closed pattern {Tim, Sam, Jay} is extremely coherent as it has complete similarity in all attributes and has the greatest possible density of 1. Had we been searching for only maximal coherent subgraphs, this important pattern would have been omitted.

Algorithm 1 Pattern Discovery

```
Input:
$G = (V, E, f)$: an attributed graph
$min\_size$: the minimum size of pattern to include in results
$\theta$: density threshold
$s_{min}$: minimum number of similar dimensions per pattern
Output:
$M$: maximal coherent patterns
$C$: closed coherent patterns
 1: $M = \{\}$
 2: $C = \{\}$
 3: MinePatterns($\{\}$)
 4: function MinePatterns($U$)
 5:     $locally\_maximal \leftarrow true$
 6:     $locally\_closed \leftarrow true$
 7:     for $v \in V \setminus U$ do
 8:         if isConnected($U$, $v$) and $\rho(U \cup \{v\}) > \theta$ and $|S(U \cup \{v\})| \geq s_{min}$ then
 9:             $locally\_maximal \leftarrow false$
10:             if $|S(U)| = |S(U \cup \{v\})|$ then
11:                 $locally\_closed \leftarrow false$
12:             end if
13:             if isChild($U \cup \{v\}$, $U$) then
14:                 MinePatterns($U \cup \{v\}$)
15:             end if
16:         end if
17:     end for
18:     if $locally\_closed$ then
19:         if $|U| \geq min\_size$ then
20:             $C = C \cup U$
21:             if $locally\_maximal$ then
22:                 $M = M \cup U$
23:             end if
24:         end if
25:     end if
26: end function
27: return $M$, $C$
```

Figure 10. Pattern Discovery algorithm. This algorithm presents pseudo-code for CoDeNE's pattern mining process.

4.3. Overlap Detection

A large dataset often yields a large number of coherent patterns, and examining those results often means filtering through overlapping subsets. These kinds of patterns are especially prominent in the closed result set, which always has at least as many patterns as the maximal set. For this reason, it is very desirable to involve some post-processing techniques for handling this problem. We apply an overlap detection method that consists of two distinct steps.

4.3.1. Finding Similarity Scores

In the first step, we apply a function which is used to assign similarity scores between pairs of cohesive subgraphs in the results. The function requires a parameter $0 \leq \alpha \leq 1$ which is used to distribute weight between the vertex similarity and attribute similarity. For a pair of coherent subgraphs U, U' and their respective attribute subspace clusters $S(U)$, $S(U')$, the pairwise similarity score can be obtained by:

$$sim(U, U') = \alpha \left(\frac{|U \cap U'|}{|U \cup U'|} \right) + (1 - \alpha) \left(\frac{|S(U) \cap S(U')|}{|S(U) \cup S(U')|} \right)$$

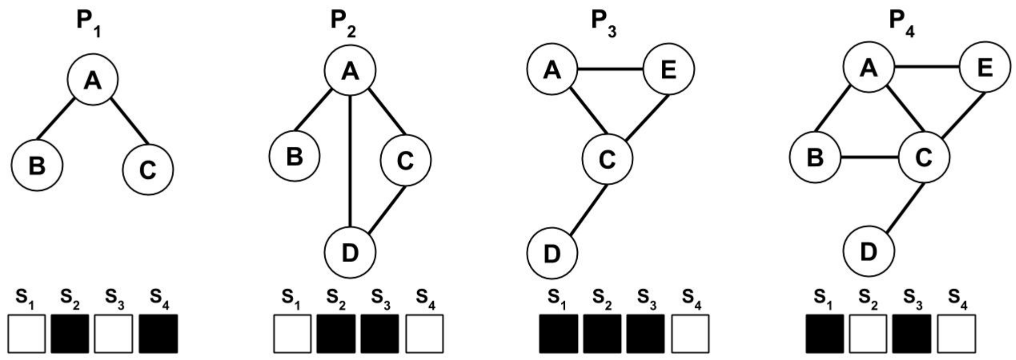


Figure 11. Cohesive subgraph pairwise similarity example. These four overlapping patterns are candidates for similarity comparison. A dark box indicates shared similarity in the subgraph at that attribute subspace.

For example, the similarity function can be applied to each pair of the four subgraphs in Figure 11. Assume $\alpha = 0.5$.

$$sim(P_1, P_2) = 0.5\left(\frac{3}{4}\right) + 0.5\left(\frac{1}{4}\right) = 0.5$$

$$sim(P_2, P_3) = 0.5\left(\frac{3}{5}\right) + 0.5\left(\frac{2}{4}\right) = 0.55$$

$$sim(P_1, P_4) = 0.5\left(\frac{2}{5}\right) + 0.5\left(\frac{0}{4}\right) = 0.3$$

Intuitively, if $\alpha = 0.5$ then we consider vertex similarity as equally important as attribute subspaces similarity. In an extreme case where $\alpha = 1$ for example, then we do not consider attribute subspaces when determining the score. If we apply $\alpha = 1$ to the same example in Figure 11, we obtain new similarity scores which depend only on the vertex composition similarity of the two subgraphs:

$$sim(P_1, P_2) = \left(\frac{3}{4}\right) = 0.75$$

$$sim(P_2, P_3) = \left(\frac{3}{5}\right) = 0.6$$

$$sim(P_1, P_4) = \left(\frac{3}{5}\right) = 0.6$$

At this point, we introduce a new threshold parameter $0 \leq \beta \leq 1$ which is used to determine if two patterns are actually considered similar; if $sim(U, U') \geq \beta$, then we say U and U' are indeed similar. In our example, if $\alpha = 0.5$ and $\beta = 0.5$, then $sim(P_1, P_2) = 0.5 = \beta$ so P_1 and P_2 are similar. In the same manner, P_2 and P_3 ($sim(P_2, P_3) = 0.55 > \beta$) are also similar but P_1 and P_4 ($sim(P_1, P_4) = 0.3 < \beta$) are not. Intuitively, as β increases, fewer patterns will be considered similar as their similarity scores fall below the threshold; inversely, if β is set to a smaller value, the number of similar patterns will be greater as even low similarity scores meet the threshold.

4.3.2. Similarity Graph

The second step of our overlap detection method involves building a new graph $G' = (V', E')$ where each vertex $v \in V'$ represents a pattern from the results and an edge between them means they are similar according to the criteria explained in Section 4.3.1 ($sim(v_1, v_2) \geq \beta$). The size $|V'|$ should be equal to the number of patterns in the result set.

We now find *dominating subsets* of this new graph. A dominating set $D \subseteq V'$ is a subset in which every vertex of V' not in D is connected to at least one $d \in D$. We can say that D is a representative of V' since every vertex in V' is represented by a vertex in D for which it is similar. Figure 12 shows two examples of valid dominating sets. However, Figure 12 (b) shows the *best* solution, as it is the solution with the fewest dominating vertices. Finding an exact solution for the dominating set is a classical NP-complete problem [8]; however, we adopt a greedy algorithm which can give us an approximation of the best solution [28].

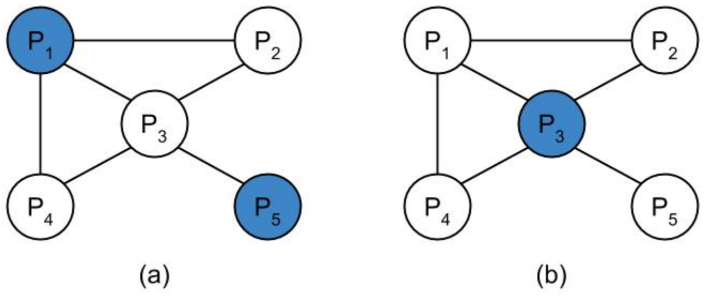


Figure 12. Two examples of dominating sets. In each, the shaded vertices make up the dominating set. (b) is the *best* solution because it has the fewest possible dominating vertices.

We can now use the similarity graph as input to the dominating set procedure (although this algorithm could be applied to any undirected graph). The relevant pseudo code is given in Algorithm 2. Initially, each input vertex is marked as *uncovered*. The process continues by iteratively scanning the graph and selecting a vertex with the greatest *uncoverd degree* — that is, a vertex who is connected to the most uncovered vertices (lines 6 - 11). Note that the uncovered degree of a vertex considers the state of itself - i.e., if a vertex is uncovered, then its uncovered degree will be 1 plus the number of uncovered neighbors it has. In the first iteration, we simply select a vertex with the highest degree since all vertices are uncovered. When a vertex has been selected, we mark it and its neighbors as covered (line 13) and add the vertex to our new summary result set R (line 14). At this point, we also update the uncovered degree of the selected vertex, its neighbors, and its neighbors' neighbors. The algorithm proceeds to the next iteration and continues this process until each vertex in the graph has been marked as covered. When finished, our summary result set will contain one vertex from each iteration and this new set represents our concise results.

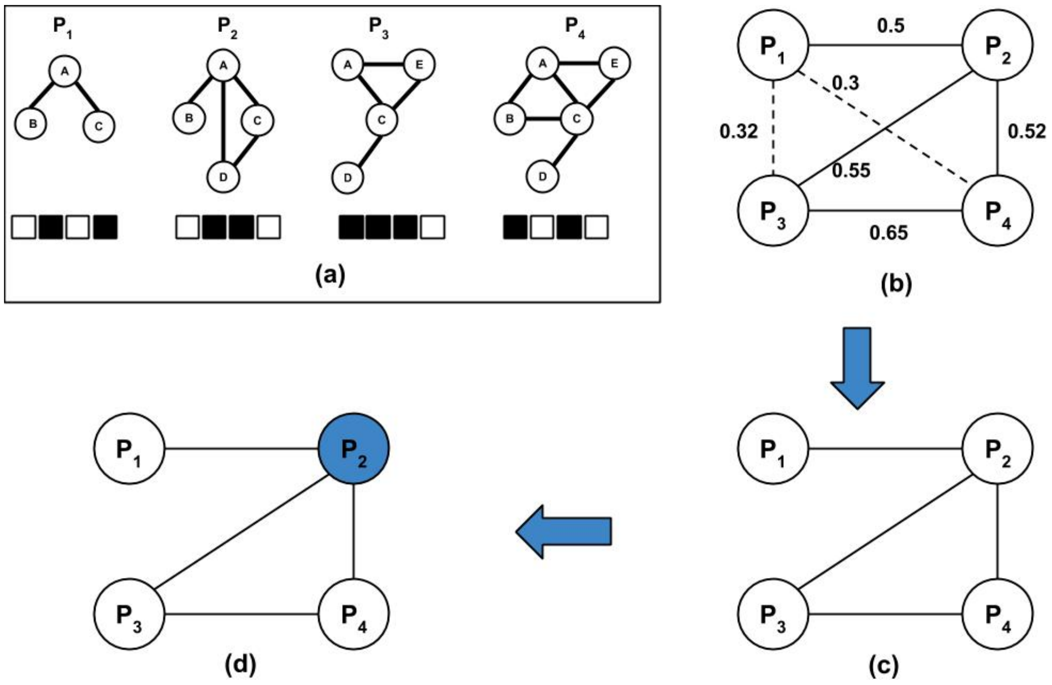


Figure 13. Visual example of the redundancy reduction process. The four coherent subgraphs in Figure 11 are shown in (a). Pairwise similarity between each coherent subgraph is calculated with $\alpha = 0.5$ and shown in (b). Applying threshold parameter $\beta = 0.5$ results in the similarity graph (c). The greedy dominating set algorithm results in P_2 being the sole representative pattern for the four cohesive subgraphs (d).

Algorithm 2 Dominating Set

```
Input:
$G = (V, E)$: a graph
Output:
$R$: representative subset of $V$
 1: $R = \{\}$
 2: $finished \leftarrow false$
 3: repeat
 4:     $covers\_most \leftarrow null$
 5:     $greatest\_ud \leftarrow 0$
 6:     for all $v \in V \backslash R$ do
 7:         if $ud(v) > greatest\_ud$ then
 8:             $covers\_most \leftarrow v$
 9:             $greatest\_ud \leftarrow ud(v)$
10:         end if
11:     end for
12:     if $covers\_most \neq null$ then
13:         coverSurrounding($v, V$)  ▷ Also updates uncovered degree for $v$ and neighbors of $v$.
14:         $R \leftarrow R \cup \{covers\_most\}$
15:     else
16:         $finished \leftarrow true$
17:     end if
18: until $finished$
19: return $R$
```

Figure 14. Dominating Set algorithm. This algorithm presents pseudo-code for CoDeNE's greedy dominating set function.

5. EXPERIMENTS

We compare our algorithm against GAMer [10] using three real-world networks and their associated attribute data: *Genes*, *High Confidence Yeast (YeastHC)*, and the *Human Protein Reference Database (HPRD)*. Table 1 outlines the information relating to each dataset.

Table 1. Experimental datasets.

Dataset	Vertices	Edges	Dimensions
Genes	3548	8334	115
HC	4008	9857	22
HPRD	9465	37039	79

For each experimental dataset, we also provide statistics of CoDeNE's maximal and closed results, as well as the summarized results for each. There are many possible permutations of parameters and statistics that could be applied and measured here. Primarily, we are concerned with measuring the size and number of resulting patterns as the minimum dimensionality and density threshold parameters are varied. It should be noted that, in each experiment, we provide the same density threshold parameter value for both algorithms. Since GAMer uses this parameter in a more restrictive way, it can be expected that the two algorithms will yield very different results. All experiments were run independently on an Arch Linux operating system with an Intel Core i5-2500K (3.3GHz) processor and 8 Gigabytes of main memory.

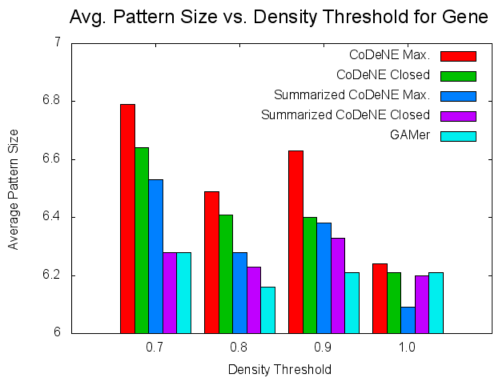
5.1. Gene Dataset

The Gene dataset was used by Gunnemann et al. [10] to compare GAMer to other algorithms and benchmarks. It consists of a protein-protein interaction network as well as expression information taken from human tissue samples. A single interaction is represented by an edge and each tissue sample is a dimension. For each algorithm, a constant minimum pattern size parameter of 6 and tolerance of 1.0 were given. Summary for CoDeNE was provided with α and β values of 0.5. Table 2 shows the statistics of the of the reported maximal dense cohesive patterns by CoDeNE before and after summary as well as GAMer. In the table, the number of resulting patterns (denoted as $|N|$) and average pattern density ($\overline{\rho}$) are given as the density threshold and minimum dimensionality parameters are varied.

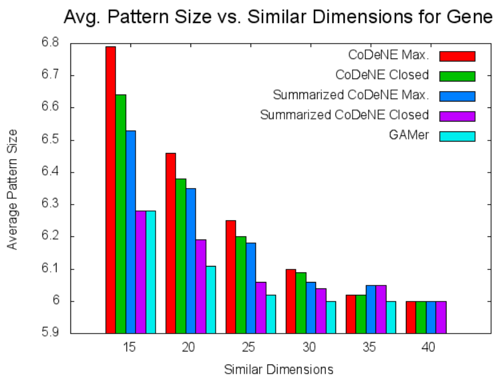
Table 2. Statistics of cohesive maximal dense patterns in the Gene dataset.

Parameters		CoDeNE		After Summary		GAMer	
θ/γ	s_{min}	$\|N\|$	$\overline{\rho}$	$\|N\|$	$\overline{\rho}$	$\|N\|$	$\overline{\rho}$
0.8	15	2556	0.82	183	0.83	719	0.9
0.8	30	314	0.82	26	0.83	35	0.9
0.9	15	287	0.94	24	0.96	180	1.0
0.9	30	19	0.94	2	0.93	3	1.0
1.0	15	70	1.0	11	1.0	180	1.0
1.0	30	3	1.0	1	1.0	3	1.0

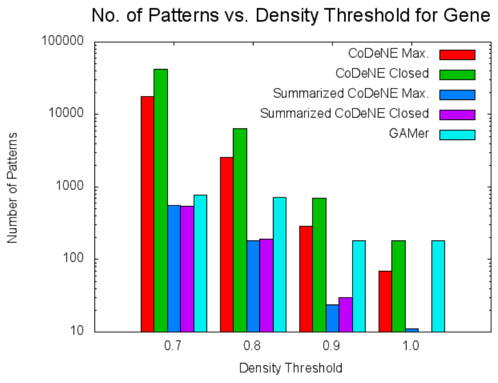
Figure 15 (a) shows that CoDeNE's maximal patterns are always the largest on average. This is to be expected; when mining maximal patterns, we select as many vertices as possible while still satisfying the threshold conditions, thus *maximizing* each result set. Because of this, and because the set of closed patterns is always a superset of the maximal patterns, we should expect to see the overall average size of closed patterns to be equal to or less than the maximal patterns. This can be verified by observing (a) and (b). Figure 15 (b), also demonstrates that increasing the dimension similarity threshold tends to result in a steady decrease of average pattern size. This happens because it is more common to find entities with fewer likenesses and more rare to find entities with many likenesses.



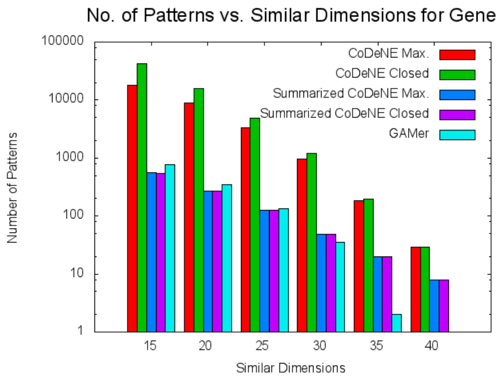
(a)



(b)



(c)



(d)

Figure 15. Results for varying θ and s_{min} for experiments on Gene dataset.

Figure 16 shows the trend for the number of resulting patterns as β increased from 0.1 to 1.0 while all other parameters remained constant.

We can visually see how, as β becomes larger, fewer patterns are considered similar, since the threshold becomes more difficult to meet. Note how the number of summarized closed and maximal patterns remains roughly the same when $\beta \leq 0.8$.

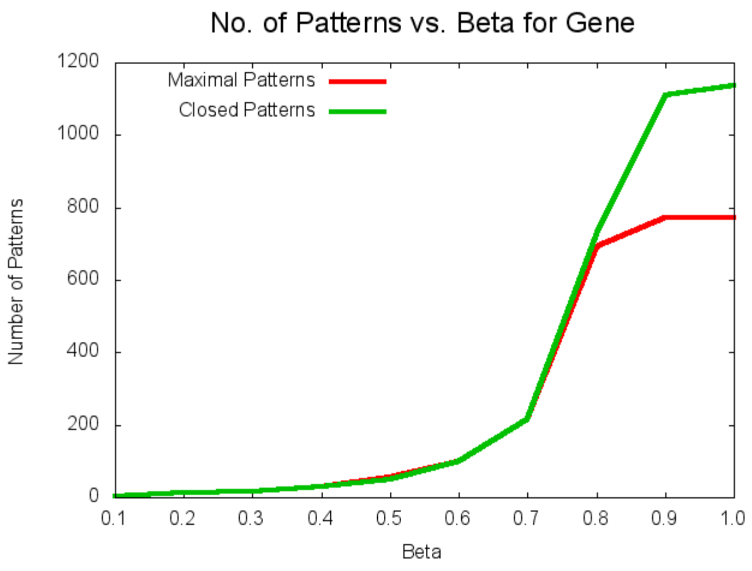
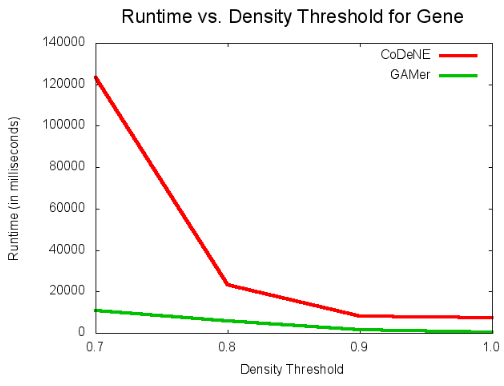
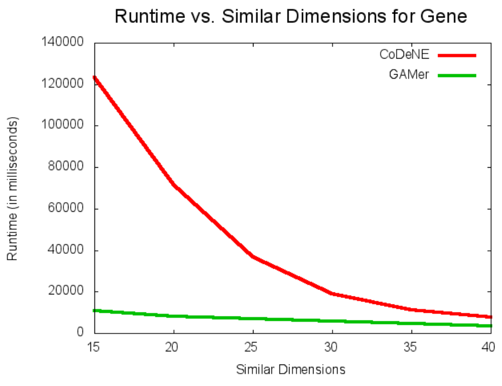


Figure 16. Beta's impact on number of summarized patterns for Gene dataset.



(a)



(b)

Figure 17. Runtimes for Gene experiments.

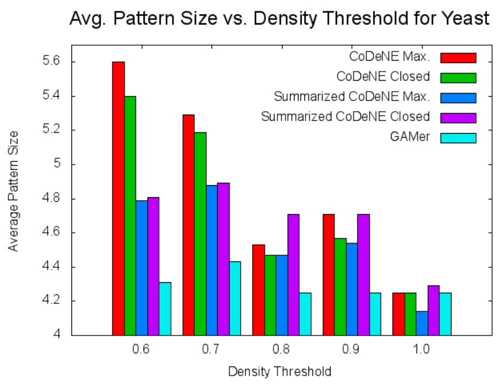
Figure 17 shows that GAMer is able to run much faster than CoDeNE, especially when density and dimensionally thresholds are lower. This is primarily because GAMer is able to prune much more often, as it uses the density threshold parameter in a much more restrictive way. Since GAMer mines for γ-quasi cliques, it can prune the enumeration tree whenever a vertex that does not meet the quasi-dense criteria is found. Because of this, GAMer will also generally report fewer resulting patterns that CoDeNE is able to discover, which can be observed in (c) and (d) of Figures 15, 18, and 19 as well as Tables 2, 3, and 5.

5.2. High Confidence Yeast (YeastHC) Dataset

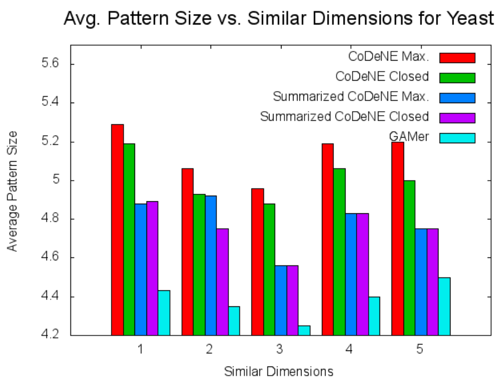
We also performed experiments on the YeastHC interaction network (Batada et al. [3]). In addition, we include growth profile information which was obtained by exposing each gene to 21 different environmental conditions (Dudley et al. [6]) and observing either a growth or growth defect. For this dataset, we represent the yeast interaction network as the graph and each environmental condition as a dimension. Since the profile data is binary, the tolerance threshold t is set to 0 in each experiment and we only consider growth when finding subspace clusters. In other words, a growth defect under a particular environmental condition will invalidate that subspace for the cluster. Table 3 and Figure 18 outline topological properties reported by CoDeNE and GAMer for this dataset. In the table, $|N|$ is used to denote the number of resulting patterns and $\overline{\rho}$ represents the average density.

Table 3. Statistics of cohesive maximal dense patterns for Yeast dataset.

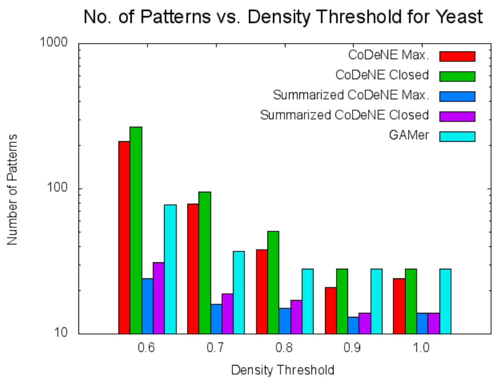
Parameters		CoDeNE		After Summary		GAMer	
θ/γ	s_{min}	$\|N\|$	$\overline{\rho}$	$\|N\|$	$\overline{\rho}$	$\|N\|$	e $\overline{\rho}$
0.7	2	34	0.75	12	0.78	17	0.9
0.7	3	24	0.77	9	0.82	8	0.9
0.8	2	13	0.87	11	0.88	13	1.0
0.8	3	12	0.89	8	0.88	7	1.0
0.9	2	8	0.95	8	0.95	13	1.0
0.9	3	6	0.98	5	0.98	7	1.0



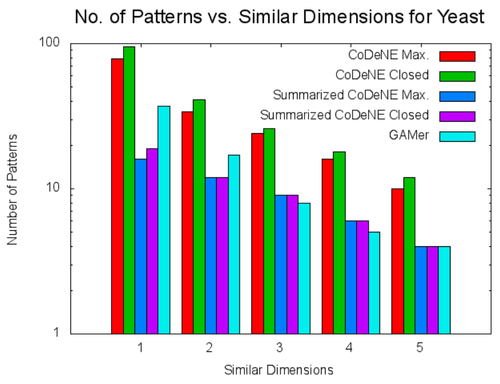
(a)



(b)



(c)



(d)

Figure 18. Results for varying θ and s_{min} for experiments on Yeast.

Figure 18 shows a visual analysis of the results. Similarly to the results of the Gene dataset, we see that the overall average size of the closed result sets are always less than or equal to the average size of the maximal sets, as expected. Note, however, that is not always the case with the summarized results since larger patterns, regardless of their density, can be removed if they are redundant, which skews the overall average.

Using the YeastHC dataset, we additionally performed some biological enrichment analysis using the **D**atabase for **A**nnotation, **V**isualization, and **I**ntegrated **D**iscovery — DAVID [11, 12]. In order to verify the significance of our results, we attempt to find enrichment of Gene Ontology process terms (GOTERMS) as well as KEGG pathway enrichment in our resulting patterns. We say that a pattern is enriched with a biological process function if that function is overrepresented in the genes from the pattern. In other words, the probability of there being a number of genes in a pattern that are involved in that process function by chance is statistically low, yet we have found them in that pattern. Because of this, we can say with a fair degree of certainty that those genes were not included in the pattern by chance — the algorithm discovered them because there is a correlation between that biological function and the density and attribute similarity of the genes in the pattern. For example, using the maximal resulting patterns discovered by providing a density threshold of 0.9, we found that four out of eight genes were enriched with the protein transport GOTERM. This is a much larger ratio than can be expected by simply choosing eight random genes from the entire dataset and checking the expression.

We selected four of our maximal result sets to be samples for the analysis. The enrichment for a given result set is calculated by dividing the number of enriched patterns by the total number of patterns in the set. For each sample result set, we found that every pattern was enriched with at least one GOTERM function, thus resulting in an enrichment score of 100% for each set. Some common biological process GOTERMS that were enriched in these patterns include protein transport (GO:0015031), establishment of protein localization (GO:0045184), and protein localization (GO:0008104).

The KEGG pathway database is a set of mappings which represent molecular interactions. To be enriched, a function must be overrepresented in the genes of the pattern and those genes must also share an interaction according to the KEGG database. Because of this, the KEGG database is more restrictive than GOTERM and higher enrichment scores are more difficult to achieve. Table 4 outlines the results for the KEGG pathway analysis using our four sample result sets. These enrichment scores for the GOTERM and KEGG database analysis pose convincing evidence that our algorithm produces results with meaningful biological significance.

Table 4. Pathway enrichment for YeastHC dataset using CoDeNE.

θ	s_{min}	$\|N\|$	KEGG
0.6	2	55	44%
0.7	2	34	35%
0.8	2	13	38%
0.9	2	8	25%

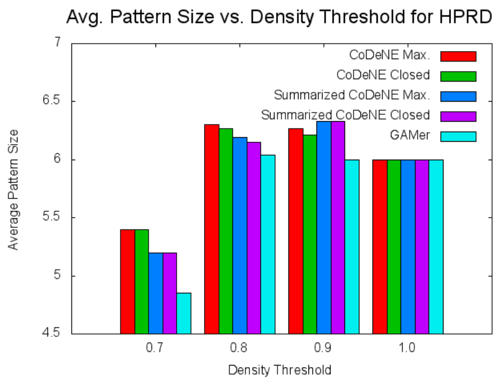
5.3. Human Protein Reference Database (HPRD)

The final experimental dataset we used was the HPRD (Peri et al. [22]) along with gene expression profile data from 79 different human tissues (Su et al. [24]). Like the YeastHC dataset, the profile information is binary; either the gene is expressed or suppressed and the tolerance threshold t was set to 0. Only the expressed genes were accepted when forming subspace clusters. Table 5 outlines the results.

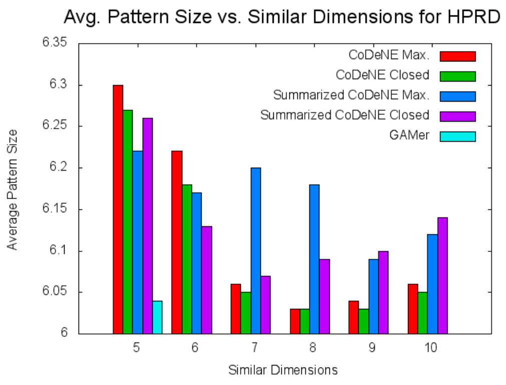
Table 5. Statistics of cohesive maximal dense patterns for HPRD dataset.

Parameters		CoDeNE		After Summary		GAMer	
θ/γ	s_{min}	$\|N\|$	$\overline{\rho}$	$\|N\|$	$\overline{\rho}$	$\|N\|$	$\overline{\rho}$
0.7	9	3084	0.7	148	0.7	129	0.84
0.7	10	2454	0.7	124	0.7	100	0.85
0.8	9	313	0.8	45	0.8	12	0.98
0.8	10	240	0.8	40	0.8	11	0.99
0.9	9	38	0.9	16	0.9	9	1.00
0.9	10	29	0.9	14	0.9	9	1.00

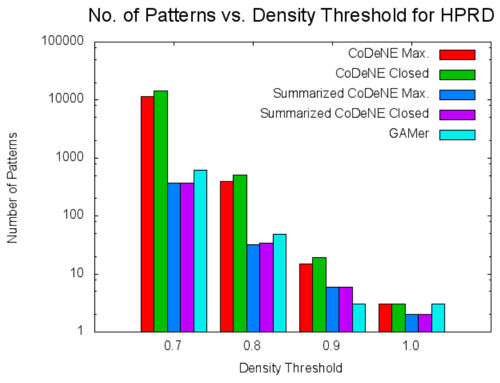
Figure 19 presents four plots for an visual overview of the different experiments. As the density threshold increases, we can observe a drastic decrease in the number of resulting patterns (c). This is natural, as it is more common to find spares communities and more rare to find closely-connected communities. Also note the impact that summarization has on the number of results in (d). This is highly beneficial since many redundant patterns can be removed from the results.



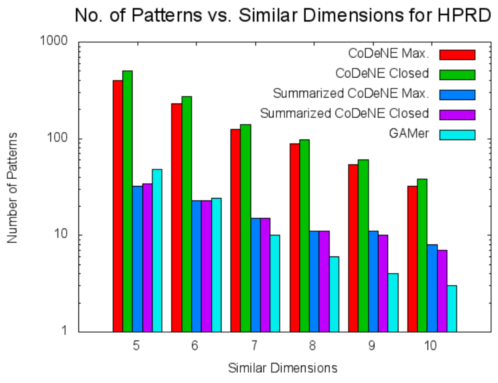
(a)



(b)



(c)



(d)

Figure 19. Results for varying θ and s_{min} for experiments on HPRD.

6. CONCLUSION AND FUTURE WORK

In this paper, we introduced a new graph mining algorithm **Co**hesive **De**nse **N**etwork **E**numeration. CoDeNE mines undirected, attributed graphs and discovers both maximal and closed cohesive dense patterns by integrating dense subgraph discovery with real value subspace clustering. In addition, CoDeNE provides post processing techniques for summarization of redundant results. We made comparisons with a related algorithm, GAMer, with three datasets and gave a summary of the results. Finally, we performed enrichment analysis to assess the functional homogeneity, and thus, the accuracy of our results.

Currently CoDeNE has the potential for a major future improvement. The nature of it using a reverse search approach allows the algorithm to be parallelized when discovering dense patterns. Since each branch in the enumeration tree is independent and does not rely on the results of another branch, each could potentially be forked to another thread and/or another machine for processing. This could lead to incredible improvements in overall performance.

7. BIBLIOGRAPHY

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] Sitaram Asur and Bernardo A Huberman. Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 492–499. IEEE, 2010.

[3] Nizar N Batada, Teresa Reguly, Ashton Breitkreutz, Lorrie Boucher, Bobby-Joe Breitkreutz, Laurence D Hurst, and Mike Tyers. Still stratus not altocumulus: further evidence against the date/party hub distinction. *PLoS biology*, 5(6):e154, 2007.

[4] Patrick Cramer, David A Bushnell, Jianhua Fu, Averell L Gnatt, Barbara Maier-Davis, Nancy E Thompson, Richard R Burgess, Aled M Edwards, Peter R David, and Roger D Kornberg. Architecture of rna polymerase ii and implications for the transcription mechanism. *Science*, 288(5466):640–649, 2000.

[5] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. *Physical review letters*, 94(16):160202, 2005.

[6] Aimée Marie Dudley, Daniel Maarten Janse, Amos Tanay, Ron Shamir, and George McDonald Church. A global view of pleiotropy and phenotypically derived gene function in yeast. *Molecular systems biology*, 1(1), 2005.

[7] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3): 75–174, 2010.

[8] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[9] Elisabeth Georgii, Sabine Dietmann, Takeaki Uno, Philipp Pagel, and Koji Tsuda. Enumeration of condition-dependent dense modules in protein interaction networks. *Bioinformatics*, 25(7):933–940, 2009.

[10] Stephan Gunnemann, Ines Farber, Brigitte Boden, and Thomas Seidl. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 845–850. IEEE, 2010.

[11] Da Wei Huang, Brad T. Sherman, and Richard A. Lempicki. Systematic and integrative analysis of large gene lists using david bioinformatics resources. *Nature Protoc.*, 4(1):44–57, 2009.

[12] Da Wei Huang, Brad T. Sherman, and Richard A. Lempicki. Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Res.*, 37(1):1–13, 2009.

[13] Daxin Jiang and Jian Pei. Mining frequent cross-graph quasi-cliques. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(4):16, 2009.

[14] Ruoming Jin, Scott Mccallen, C Liu, Y Xiang, E Almaas, and XH Zhou. Identify dynamic network modules with temporal and spatial constraints. In *Pacific Symposium on Biocomputing*, 2009.

[15] Victor E Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, pages 303–336. Springer, 2010.

[16] Jinyan Li, Kelvin Sim, Guimei Liu, and Limsoon Wong. Maximal quasi-bicliques with balanced noise tolerance: Concepts and co-clustering applications. In *SDM*, volume 8, pages 72–83. SIAM, 2008.

[17] Guimei Liu and Limsoon Wong. Effective pruning techniques for mining quasi-cliques. In *Machine Learning and Knowledge Discovery in Databases*, pages 33–49. Springer, 2008.

[18] Hideo Matsuda, Tatsuya Ishihara, and Akihiro Hashimoto. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theoretical Computer Science*, 210(2):305–325, 1999.

[19] Flavia Moser, Recep Colak, Arash Rafiey, and Martin Ester. Mining cohesive patterns from graphs with feature vectors. In *SDM*, volume 9, pages 593–604. SIAM, 2009.

[20] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[21] Jian Pei, Daxin Jiang, and Aidong Zhang. On mining cross-graph quasi-cliques. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 228–238. ACM, 2005.

[22] Suraj Peri, J Daniel Navarro, Troels Z Kristiansen, Ramars Amanchy, Vineeth Surendranath, Babylakshmi Muthusamy, TKB Gandhi, KN Chandrika, Nandan Deshpande, Shubha Suresh, et al. Human protein reference database as a discovery resource for proteomics. *Nucleic acids research*, 32(suppl 1):D497–D501, 2004.

[23] Max F Perutz, MG Rossmann, Ann F Cullis, Hilary Muirhead, and Georg

Will. Structure of hæmoglobin: a three-dimensional fourier synthesis at 5.5-å. resolution, obtained by x-ray analysis. *Nature*, 185:416–422, 1960.

[24] Andrew I Su, Tim Wiltshire, Serge Batalov, Hilmar Lapp, Keith A Ching, David Block, Jie Zhang, Richard Soden, Mimi Hayakawa, Gabriel Kreiman, et al. A gene atlas of the mouse and human protein-encoding transcriptomes. *Proceedings of the National Academy of Sciences of the United States of America*, 101(16): 6062–6067, 2004.

[25] Amy Hin Yan Tong, Becky Drees, Giuliano Nardelli, Gary D Bader, Barbara Brannetti, Luisa Castagnoli, Marie Evangelista, Silvia Ferracuti, Bryce Nelson, Serena Paoluzi, et al. A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules. *Science*, 295 (5553):321–324, 2002.

[26] Takeaki Uno. An efficient algorithm for enumerating pseudo cliques. In *Algorithms and Computation*, pages 402–414. Springer, 2007.

[27] Stijn Marinus van Dongen. Graph clustering by flow simulation. 2000.

[28] Vijay V Vazirani. *Approximation algorithms.* springer, 2001.

[29] Stanley Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.