Spectral Inference of a Directed Acyclic Graph Using Pairwise Similarities

Allison Paul

November 11, 2015

Abstract

A gene ontology graph is a directed acyclic graph (DAG) which represents relationships among biological processes. Inferring such a graph using a gene similarity matrix is NP-hard in general. Here, we propose an approximate algorithm to solve this problem efficiently by reducing the dimensionality of the problem using spectral clustering. We show that the original problem can be simplified to the inference problem of overlapping clusters in a network. We then solve the simplified problem in two steps: first we infer clusters using a spectral clustering technique. Then, we identify possible overlaps among the inferred clusters by identifying maximal cliques over the cluster similarity graph. We illustrate the effectiveness of our method over various synthetic networks in terms of both the performance and computational complexity compared to existing methods.

1 Introduction

A gene ontology (GO) is a tool to categorize genes of different biological processes to facilitate their study and interpretation. Genes are classified under gene ontology terms, which describe their biological components, molecular functions, and cellular components [1]. Examples of gene ontology terms include oxygen binding, response to x-ray, and sympathetic nervous system development [2]. Each gene ontology term is defined as a subset of genes involved in a specific biological process. Genes belonging to term A can be a subset of genes that belong to term B. These relationships have been characterized using a directed acyclic graph (DAG), which we refer to as a GO-graph. Nodes of this graph are different GO terms, while an edge $A \to B$ means genes of term B are all in term A as well. We represent these biological relationships using DAGs rather than trees, because, based on relationships among genes, GO terms must be able to have more than one parent (i.e. $A \to C$, and $B \to C$ can exist at the same time [3].

Researchers use a worldwide GO database, the Gene Ontology Consortium [2] to obtain information about previously studied genes, as well as to predict the function of previously unstudied genes [3]. The GO Consortium has become the trusted source for researching the functions of any experimental system in biology [4]. The inference of a GO-graph is essential to maintaining and updating these GO databases. Traditional methods of forming a GO-graph rely on experimental studies of genes and their involvement in different biological processes. However, the generation of GO-graphs is limited by the time needed to create the graphs manually and human bias against genes that are unstudied. Recently, researchers have proposed a way of algorithmically inferring such graphs using similarities between genes computed from other functional associations [4]. One can use these approximated GO-graphs generated to populate and improve the accuracy of the original graphs generated by human efforts. Furthermore, fast and efficient methods of GO-graph inference would make new

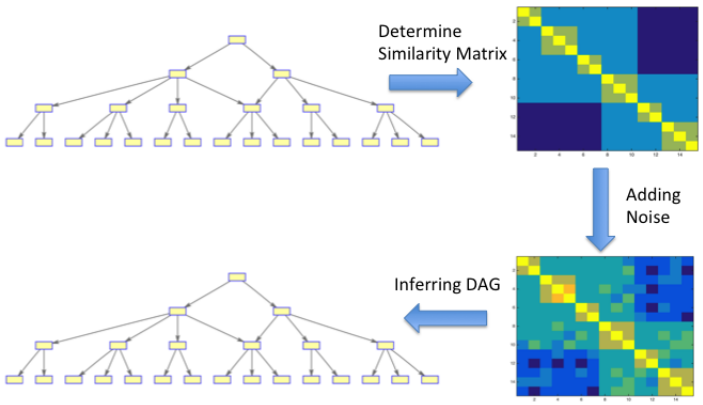

Figure 1: An illustration of DAG Inference methods. Notice the connections between the actual DAG and the similarity matrix generated. The cluster that has more than one parent corresponds to the overlap between the clusters in the image. We show that the DAG inference problem is equivalent to the problem of overlapping clusters in a network.

research techniques possible, such as to generating and contrasting GO-graphs from diseased and healthy samples [4]. These GO-graphs would be a revolutionary way to study the effects of disease on the relationships between complex biological processes.

We have two sources of uncertainty in the GO-graph inference problem using gene similarity data. One is the experimental noise in real gene expression data used to compare gene similarities. The second is the topological noise introduced in the model relating the underlying GO-graph to gene similarities.

Recently, [3] has investigated the problem of inferring a GO-graph from gene similarity data. Reference [3] shows that nodes at different layers of the GO-graph correspond to maximal cliques of the thresholded similarity matrix of the genes. In addition, [3] proposes an algorithm to solve the GO-graph inference problem iteratively by constructing nodes at each layer based on such maximal cliques using a threshold greater than the threshold of the layer below. A maximal clique of a graph corresponds to a maximal independent

set over the complement graph with edges between all non-neighbors in the original graph. One disadvantage of the algorithm proposed in [3] is its high computational complexity, partially due to multiple uses of maximal clique solvers, which have worst case computational complexity $\mathcal{O}(3^{n/3})$.

Here, we propose a new spectral algorithm to approximately solve the GO-graph inference problem that can be efficiently applied to large and noisy gene similarity data sets. We show that the GO-graph inference problem can to simplified to the inference problem of overlapping clusters in a network. We then solve this problem in two steps: first, we infer clusters using a spectral clustering technique. Using these inferred clusters, we form a cluster-similarity graph which has significantly smaller dimension (nodes) than the number of genes. Then, we show that nodes at the next layer of the GO-graph with multiple parents correspond to maximal cliques of the cluster-similarity graph. We merge these overlapping clusters, and then proceed to infer the next (lower) layer in the GO-graph.

This dimensional reduction saves significant amounts of time. Our method infers for instance, a 3,000 gene DAG in approximately 100 seconds. We account for noise and test the accuracy of our model under different noise levels. Finally, we propose techniques to learn model parameters from input pairwise similarity data.

2 Materials and Methods

2.1 Notation

Suppose $G = (V, E, W)$ is a DAG where V, E, and W represent nodes, edges, and edge weights respectively. Suppose there exist n terminal nodes in the graph (i.e., nodes without outgoing edges) denoted by $V_T \subseteq V$. Non-terminal nodes are represented by $V_N = V - V_T$. Let $d(u, v)$ be the length of the shortest path between nodes v and u ignoring edge orientations. Let $children(u)$ denote terminal descendants of node u.

In [3], a layered directed acyclic graph (DAG) has been defined to model genome ontology (GO) terms as follows:

Definition 1 (Layered DAG) *Graph $G = (V, E, W)$ is called a layered DAG if it is a DAG, and*

- *(i) A non-terminal node u has constant distance to all its terminal descendants. I.e., $\forall u \in V_N$, and $\forall v_1, v_2 \in children(u)$, we have $d(u, v_1) = d(u, v_2)$.*
- *(ii) For a non-terminal node u, and all terminal nodes $v_1 \notin children(u)$, there exists $v_2 \in children(u)$ such that $d(v_1, v_2) > 2d(u, v_1)$.*

Let $\mathscr{D} \triangleq \{G \mid G \text{ is a layered DAG}\}$. A distance matrix $D \in \mathbb{R}^{n \times n}$ of a graph $G = (V, E, W)$ is defined such that $D_{i,j} = d(i, j)$. Then similarity matrix $S \in \mathbb{R}^{n \times n}$ is defined such that $S_{i,j} = \max(D) - D_{i,j}$, where $\max(D)$ is the maximum element of the matrix D (i.e. the maximum distance between terminal nodes in the graph). One can scale these similarities linearly to have all elements between 0 and 1. By abuse of notation, we use S for the scaled similarity matrix as well. We define a function $f : \mathscr{D} \to \mathbb{R}^{n \times n}$ such that $f(G) = S$, where S is the scaled similarity matrix of G.

2.2 Problem Statement

Suppose S_{true} is a similarity matrix of a layered DAG, $G_{true} = (V_{true}, E_{true}, W_{true})$, i.e. $S_{true} = f(G_{true})$. We observe a noisy version of S_{true}, i.e. $S_{obs} = S_{true} + Z$, where Z represents noise. Our goal is to find a layered DAG, $G = (V, E, W) \in \mathscr{D}$, whose similarity matrix approximates the observed similarity matrix closely:

$$\min_{G \in \mathscr{D}} \| S - S_{obs} \|_{l1} \tag{2.1}$$

where $S = f(G)$.

2.3 Reduction to Maximal Clique Problem

Recall that reference [3] has proposed a deterministic algorithm to infer a DAG by solving for maximal cliques. Reference [3] suggests that maximal cliques of a thresholded similarity matrix correspond to the terms of a layer in the DAG (Figure 1). Here, we prove that maximal cliques of a smaller submatrix corresponding to a node in the above layer correspond to the children of that node. We identify these cliques and then merge similar cliques to account for overlaps, as terms with multiple parents create overlaps among cliques. We show that the GO-graph inference problem can to simplified to the inference problem of overlapping cliques in a network. In the following, we detail the process that solves this inference problem deterministically. We rely on the fact that $S_{obs} = S_{true}$, i.e., that $Z = 0$. In 3.1, we extend this method to solve the more realistic, noisy method.

To simplify notation, we denote S_u as the subgraph of the similarity matrix corresponding to a non-terminal node u located at layer l. The size of S_u is equal to the number of terminal descendants of node u. Suppose we have constructed the underlying DAG until layer l (which corresponds to threshold τ_l). Note that $\tau_{l+1} \geq \tau_l$ so that the maximal cliques identified are smaller and have higher densities than the cliques of the previous layer. Next, we wish to infer nodes at layer $l+1$ (which corresponds to threshold τ_{l+1}). To find children of u, one needs to compute maximal cliques over subgraph $S_u^{\tau_{l+1}}$, the binary graph generated by thresholding S_u by τ_{l+1}. Note that subgraph sizes decrease at subsequent layers, reducing the size of the problem at each layer. After finding children of all nodes at layer l separately, we check for overlaps among those children and merge overlapping nodes. The algorithm terminates upon reaching the terminal nodes (leaves).

Theorem 1 *Suppose S_{true} is a similarity matrix of a layered DAG, $G_{true} = (V_{true}, E_{true}, W_{true})$. Let the output DAG of the above described algorithm be $G = (V, E, W)$. Then, $V = V_{true}$ and $E = E_{true}$ (up to permutations of labels).*

Proof

Let D be the distance matrix of G_{true}. WLOG, let the length of each edge in G_{true} equal 1. Let the maximum element of the distance matrix D be γ, and the minimum nonzero element be δ. Then the number of layers in G_{true}, L, not including the layer of terminal nodes, equals $\frac{\gamma}{\delta}$, as the size of γ, the maximum distance between nodes in G_{true}, is proportional to the length of the path from a node to the root node, with a scaling factor of size δ, the minimum nonzero distance between nodes. WLOG, we ignore cases where the root has only one child in the next layer because we can treat the first node with more than one child as the root.

There are exactly $L+1$ distinct values in D, as there are exactly $L+1$ layers in G_{true} if we include the layer of terminal nodes. Then there are $L+1$ distinct values in S as well. Let the threshold density at layer k, τ_k be the $k+1^{th}$ smallest of the $L+1$ values in S.

Then $\forall$ nodes u, $d(u,u) = 0$, so $0 \in S$ and the minimum element of S is 0. At layer 0, all genes have similarity greater than or equal to $\tau_0 = 0$, so the maximal clique identified is the complete graph on the genes. Our method correctly determines that all terminal nodes have the same root node. Thus, as our DAG inference method evaluates G correctly for layer 0, we assume that it evaluates G correctly up to and including layer k. If $k = L$, we are done. Else, we consider layer $k+1$.

Let $C_k = \{\text{clusters } c \text{ in layer } k\}$ in G_{true}. WLOG, consider any cluster c. Let the node corresponding to this cluster be u. We determine the maximal cliques over subgraph $S_u^{\tau_{k+1}}$. We know that $\forall s \in S_u$, $s \geq \tau_k$, as $u \in C_k$. Let $C_{k+1} = \{\text{clusters } c \text{ in layer } k+1\}$ in G_{true}. Then $\forall c \in C_{k+1}$, $\forall s \in S_c$, $s \geq \tau_{k+1}$. WLOG, choose any $c \in C_{k+1}$. Let the node corresponding to this cluster be v. The maximum distance between genes in S_v is one interval less that the maximum distance between genes in S_u, as we have reduced the subgraph by one layer. Thus, the minimum element in S_u, which is τ_k, is one increment less than the minimum element in S_v. Thus, the minimum element in S_v is τ_{k+1} by construction of the thresholds, and $S_v^{\tau_{k+1}}$ is a clique. Similarly, $\forall c \in C_{k+1}$, we can find a corresponding clique in $S^{\tau_{k+1}}$. Note

that these cliques may overlap, so we combine cliques that belong to different parents if the cliques form larger cliques under threshold τ_{k+1}.

All that is left to show is that these cliques are maximal. Assume $\exists$ terminal node $t \notin c$ such that $S_{v \cup t}^{\tau_{k+1}}$ is a complete graph on $n+1$ vertices, where n is the number of terminal nodes corresponding to node v. This implies that t and the terminal nodes corresponding to node v belong to one cluster in layer $j \geq k+1$. Let the node corresponding to this cluster be w. Then $S_v \subset S_w$, so node v is in a layer below the layer containing w. But $c_v \in C_{k+1}$, $c_w \in C_j$ and $k+1 \leq l$. $\Rightarrow\Leftarrow$

Thus, cliques corresponding to the nodes in layer $k+1$ in G_{true} are maximal, and the determining the maximal cliques of $S_u^{\tau_{k+1}}$ $\forall$ such u such that $c_u \in C_k$ correctly determines the structure of layer $k+1$.

■

3 Results and Discussion

3.1 Spectral DAG Inference (SDI) Algorithm

In 2.3, we showed that in the noiseless case, the DAG inference problem is equivalent to identifying overlapping clusters among terminal nodes. However, this is a NP-hard problem and this process is limited by the computational complexity of the maximal clique problem. There are some algorithms to identify maximal cliques efficiently by finding maximal independent sets of the complement graph instead of maximal cliques of the original graph. The complement graph will be sparse, and fast approximate algorithms based on graph colorings can be used to discover maximal independent sets. There exist efficient approximate and randomized algorithms to solve the maximal independent set problem using Monto Carlo sampling [5] and graph colorings [6, 7]. However, references [8, 9] show that a graph with n

nodes can have up to $\mathcal{O}(3^{n/3})$ maximal cliques, implying that algorithms to find all maximal cliques of a graph have worst case computational complexity $\mathcal{O}(3^{n/3})$. The maximal clique problem can be solved in polynomial time only in some special cases such as claw-free or quasi-line [10, 11], or growth-bounded [12] graphs.

Here, we propose a spectral method which approximately solves the DAG inference problem and can be used efficiently for large and noisy gene similarity data. Our key idea is reducing the dimension of the DAG inference problem using spectral clustering and then using a maximal clique inference technique over smaller subspaces.

Recall that a layered DAG has a root node in the first layer and some terminal nodes in the last layer. Throughout this section, we assume that we know the number of layers and the threshold densities at each layer of the underlying GO-graph. In section 3.2, we discuss learning these parameters using the gene similarity data.

Suppose S_{true} is a similarity matrix of a layered DAG, $G_{true} = (V_{true}, E_{true}, W_{true})$. The problem is to infer a layered DAG, $G = (V, E, W)$, using S_{obs} where $S_{obs} = S_{true} + Z$. Our key idea is to use spectral clustering to reduce the dimension of the maximal clique inference step of the algorithm present in 2.3. A cluster C is a quasi-clique, or a subset of terminal nodes with high density, where density, ρ, is defined as

$$\rho(C) \triangleq \frac{\sum_{i,j \in C} S_{i,j}}{\sum_{i,j \in C} \mathbb{1}} \tag{3.1}$$

In the SDI Algorithm, we infer these clusters at each step before determining the maximal cliques over the clusters themselves. By performing the maximal clique algorithm on the clusters rather than the terminal nodes, we significantly reduce the dimension of the problem, while still accurately inferring the structure of the DAG.

Recall that S_u denotes the subgraph of the similarity matrix corresponding to a non-terminal node u. Suppose we have constructed the underlying DAG until layer l. Next, we

wish to construct nodes at layer $l+1$. Suppose u is a node at layer l whose corresponding terminal nodes are I_u. Then, we partition S_u to k groups by clustering top $k-1$ elements of the eigenvectors v_i of the modularity matrix M $\forall i \in I_u$. To capture overlapping clusters, one can use a k-means algorithm to perform clustering over these real vectors. We choose the minimum number of clusters k so that cluster densities are above a threshold τ_{l+1}. These clusters correspond to quasi-cliques of the matrix S_u, or children of node u. After finding children of all nodes at layer l separately, we check for overlaps among those children and merge overlapping nodes. The algorithm terminates upon reaching the terminal nodes.

Our algorithm iteratively infers nodes in a lower layer using nodes in the previous layer. WLOG, we explain the algorithm only to infer the next layer. The first layer has just one node, the root node. Our algorithm has two steps:

1. Spectral Clustering using Modularity Transformation
2. Maximal Clique Inference over Cluster-Similarity Graph

Below, we explain these steps with more details:

Spectral Clustering using Modularity Transformation: In this step, we use a standard spectral clustering algorithm based on a modularity transformation [13]. We briefly explain this method as follows. Suppose we wish to infer k clusters (non-overlapping) in the similarity matrix S. First, we form its modularity matrix as follows:

$$M = S - \frac{d^T d}{m} \tag{3.2}$$

where d represents the vector of node degree in the weighted similarity graph S and m is the total sum of similarity pairs.

Then, we compute the spectral decomposition of M to its eigenvectors and eigenvalues, $M = V\Sigma V^T$, where columns of the matrix V represent eigenvectors and diagonal elements of

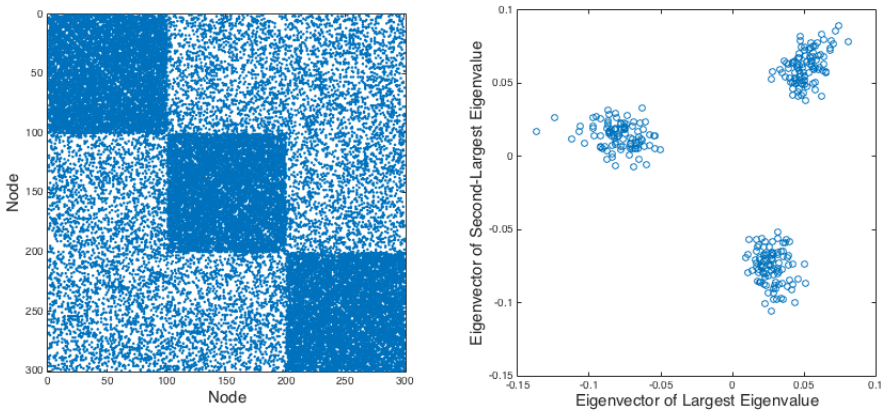

Figure 2: An example of Spectral Clustering. To the right is a depiction of the a network with three distinct clusters. Each blue dot represents an edge between the node of the x-axis and the node of the y-axis. To the right, the values of the eigenvectors of a similarity matrix corresponding to each node of the network. It is apparent that the top two eigenvectors partition the network into three clusters, as desired.

the matrix Σ represent eigenvalues (denote by λ_i). Eigenvalues are ordered from the largest to the smallest. Then, to infer k clusters, we perform a standard k-means algorithm [14] over the $k-1$ eigenvectors corresponding to the $k-1$ largest eigenvalues of the modularity matrix M. We set $k = 1$, and increase k until the mean density of the k clusters is greater than τ_l. We shall discuss the methods of selecting a maximum value for k to improve the computational complexity of the spectral decomposition step in Section 3.2.

If a node in G_{true} has multiple parents, it corresponds to an overlap between these clusters (Figure 1). Spectral clustering methods are unable to identify overlapping clustering (Figure 3). We account for these overlaps below.

Maximal Clique Inference over Cluster-Similarity Graph: To find overlapping clusters corresponding to nodes with multiple parents in the graph, we propose the following algorithm:

First, we form a cluster similarity graph where nodes represent the inferred clusters and are connected by weighted edges. If two clusters are more likely to overlap with each other,

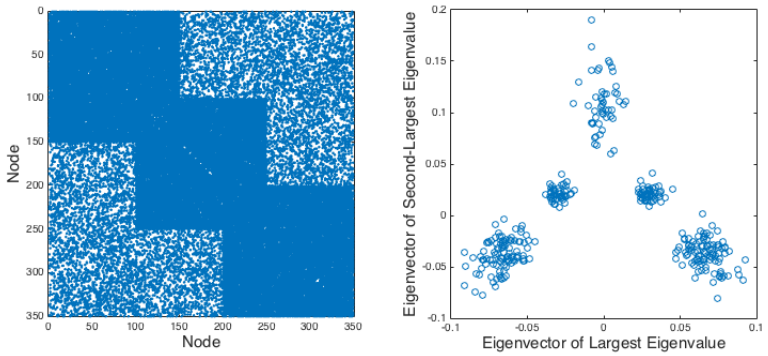

Figure 3: An example of Spectral Clustering when applied to overlapping clusters. Though three clusters are clearly visible, five clusters are detected when the k-means algorithm is applied to the top two eigenvectors of the similarity matrix. These five clusters correspond to the three groups of nodes that belong to exactly one cluster, as well as the two overlaps.

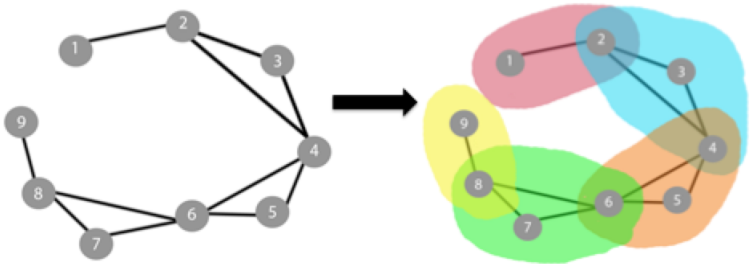

Figure 4: An illustration of the maximal clique step. We map each cluster to a node in a new network, and assign edges between nodes that are similar. Then, we determine the maximal cliques of this clusters. We merge clusters that belong to to the same clique.

their corresponding nodes in the cluster-similarity graph are connected with a higher edge weight, W, where

$$W(C_1, C_2) \triangleq \rho(C_1 \cup C_2) \tag{3.3}$$

and $C_1 \cup C_2$ refers to the cluster in S formed by combining the terminal nodes of both clusters 1 and 2 into a single cluster.

We threshold this graph to get a binary graph. We set the threshold at $\alpha\tau_l$, where α is

a parameter slightly less than 1 to account for the fact that the merged cluster may have slightly lower density than the smaller clusters. Then, we infer the maximal cliques of this graph by solving the equivalent maximal independent set problem. These maximal cliques correspond to the nodes of the layer we are inferring, as desired.

In the following, we summarize the described algorithm. Let V_N^l be non-terminal nodes at level l. $V_N^0 = \{r\}$ only includes the root node. Let τ_l be the threshold density for clusters at level l.

Algorithm 1 Spectral DAG Inference

Input: S, M, α, n_{layers}, τ_i for $0 \leq i \leq n_{layers}$
Initialization: , $V_N^0 = \{r\}$, $l = 0$
while $l \leq n_{layers}$ **do**
  for $i \in V_N^l$ **do**
    Eigen Decomposition: $M_i = U\Sigma U^T$
    $C_i =$ **SpectralClustering** (S_i, U, τ_l)
    $E = E \bigcup \{i \to C_i\}$
    $V_N^{l+1} = V_N^{l+1} \bigcup C_i$
  end for
  MergeClusters $(V_N^{l+1}, \alpha\tau_l)$
  $l = l + 1$
end while
Output: $G = (V, E)$

Algorithm 2 Spectral Clustering

Input: S, U, τ
Initialization: $\tau' = \text{density}(S)$, $k = 1$
while $\tau' < \tau$ **do**
  $k = k + 1$.
  $u_i = U[i, 1 : (k-1)]$.
  $C_k =$ **KMeans**$([u_i], k)$
  $\tau' = \text{density}(S, C_k)$
end while
Output: C_k

3.1.1 Complexity Analysis

In the spectral algorithm, we decompose the similarity matrix at each step of the algorithm. The worst case complexity of this step is $\mathcal{O}(n^3)$, where n is the number of genes. However, a full eigen decomposition of the similarity matrix S is not necessary because only top eigenvectors and eigenvalues are required in the method, decreasing the computational complexity of the eigen decomposition step substantially (up to a linear complexity in n). Moreover, each iteration of the k-means clustering at node u has a computational complexity $\mathcal{O}(|I_u|k^2)$ where k is the number of clusters and I_u is the number of terminal descendants of node u (i.e., its complexity is linear in terms of the size of subgraphs). Lastly, the maximal clique solver has worst case computational complexity $\mathcal{O}(3^{k/3})$ where k is the number of clusters in the layer, which is significantly less than the total number of genes. Thus, we see that our SDI Algorithm greatly improves on the efficiency of existing methods.

3.2 Learning Model Parameters

Our algorithm as stated, requires an input number of layers and a set of threshold densities for each layer. While it is feasible to input the desired number of layers and thresholds as in [3], we propose a learning model for the program to infer the following parameters directly from the similarity data given:

1. Determining Number of Layers
2. Determining Threshold Densities at each Layer
3. Selecting the Cut-off for the K-means Algorithm

Determining Number of Layers We use distance matrix D of G_{true} to infer the number of layers in the inferred DAG G. Let the maximum element of the distance matrix D be γ, and the minimum nonzero element be δ. Then the number of layers in G_{true} equals $\frac{\gamma}{\delta} + 1$,

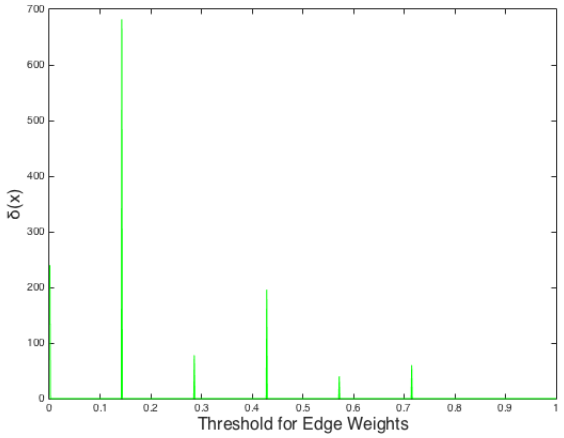

Figure 5: Values of $\delta(x)$ where $\delta(x) = \#\{(i,j) \mid S(i,j) \leq x + \Delta x\} - \#\{(i,j) \mid S(i,j) \leq x\}$ and we set $\Delta x = .001$. The DAG in this example had three layers, not including the terminal nodes, so we select the three highest values of $\delta(x)$ as the threshold densities for the three layers (0, 0.1420, 0.482). Using these three values, we will partition the genes into c clusters such that c is the minimum value such that the mean density of the clusters is greater than or equal to the threshold.

as the size of γ is proportional to the length of the path from a node to the root node, with a scaling factor of size δ. The length of this path equals the number of layers - 1. Note that with significant experimental noise, this learning model is unreliable, as this method is sensitive to noise. Thus, the method must be run multiple times and optimized over the number of layers. This does not have a large effect on the efficiency of the program. With only topological noise, this model gives a good approximation of the number of layers in the inferred DAG.

Determining Threshold Densities at each Layer: We choose the threshold densities in order to accurately determine the number of clusters at each layer. We can infer the ideal thresholds based on the similarity matrix S. We examine the values of the elements in S and choose values based on the most frequently occurring values. We define $\varphi(x) = \#\{(i,j) \mid S(i,j) \leq x\}$. Then we examine the distances between the values of $\varphi(x)$, using $\delta(x) = \varphi(x+\Delta x) - \varphi(x)$ for $0 \leq x, x+\Delta x \geq 1$. We set $\Delta x = .001$ or an arbitrary small number,

as this step has minimal computational complexity. We observe that the greatest values for $\delta(x)$ occur near the average clusters densities of each layer (See Figure 5).

We choose a threshold equal to the cluster densities of the layers in order to ensure that the correct number of clusters is determined for use in the k-means algorithm. Thus, we choose the thresholds of the layers as the x coordinates of the greatest n values of $\delta(x)$, where n = number of layers.

In the noisy case, the greatest values of $\delta(x)$ do not necessarily correspond to the correct threshold densities. We instead determine the maximally distributed numbers. We identify the maximal value (x, $\delta(x)$) and then ignore all values within a r-neighborhood of x, and repeat until we have determined a threshold for each layer of the DAG. The value of the radius r is dependent on the number of layers in G.

When inferring a layer of a DAG, we apply Spectral Clustering, starting at $k = 1$ to the submatrix of S, increasing the value of k until the mean density of the k clusters is greater than or equal to the determined threshold density. For this reason, we only need to compute the top m eigenvectors of the modularity matrix, where m is the size of the maximum number of children for one node in the DAG. In the following, we discuss our model to predict the number of eigenvectors to compute.

Selecting the Cut-off for the K-means Algorithm: We wish to compute only the top c eigenvectors, where c is the cutoff for the k-means algorithm, to decrease the computational complexity of the eigen decomposition step substantially (up to a linear complexity in n).

We aim to select a cut-off, c, for the value of k in the k-means algorithm, so that we only compute the top c eigenvectors of the similarity matrix at each step. In a DAG with no overlapping clusters, we see that the least possible cutoff, $c_{min} = \sqrt[l]{n}$, where l is the number of layers in the DAG (excluding the layer of terminal nodes) and n is the number of genes in the DAG. To account for the case where a DAG has overlapping clusters, the cutoff must be increased because two clusters may overlap, increasing the possible size of an individual

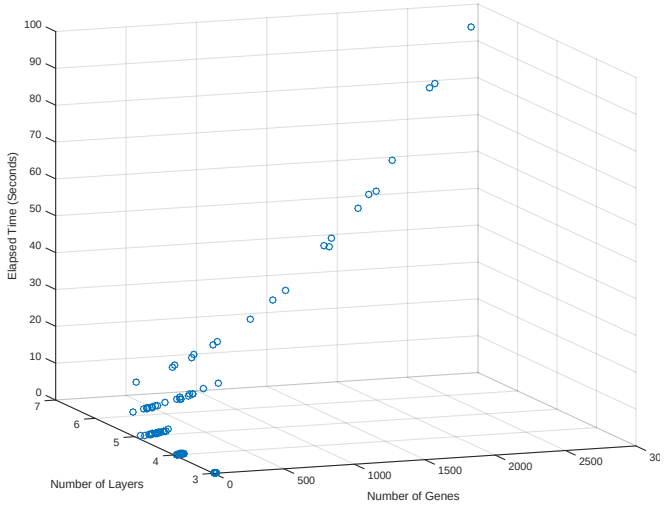

Figure 6: Seconds required to apply the SDI algorithm to DAGs of varying sizes. We tested DAGs with 3-7 layers and DAGs with up to 3000 genes. We see that the SDI algorithm is significantly faster than the previous algorithm in [3].

cluster.

3.3 Performance Evaluation Over Synthetic Networks

We evaluate the performance of the proposed algorithm in Section 3.1. First, we generate synthetic DAGs as follows: We input the mean number of children, the number of layers, and the percent of overlapping, and we generate a random DAG using those parameters. We avoid highly irregular and complex graphs by restricting overlapping structure to structures found in actual GO-graph formations. For instance, we do not generate graphs where C_A overlaps with C_B, C_B overlaps with C_C, and C_C overlaps with C_A.

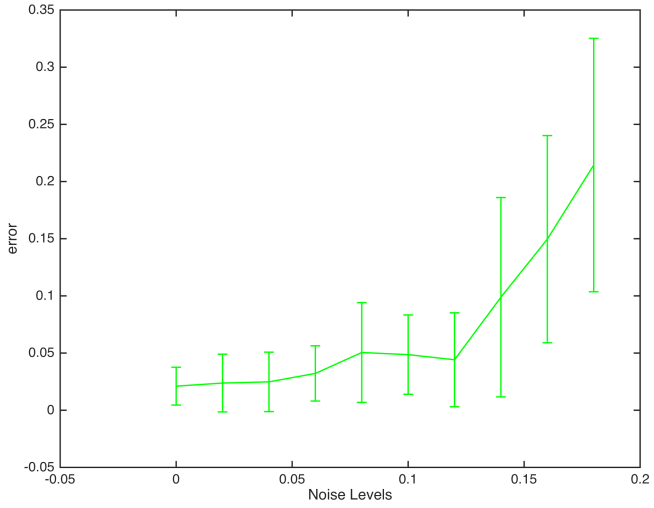

Figure 7: Performance of SDI Algorithm with increasing noise levels. We ran the SDI algorithm 50 times for each value for $\beta \in [0, 0.2]$.

Then, we add noise to the calculated similarity matrix, S_{true}, so that $S_{obs} = S_{true} + Z$, where Z is Gaussian noise and $Z_{i,j} \sim N(0, \beta d(i,j))$, a gaussian random number with mean 0 and variance $\beta d(i,j)$. Let β be the noise parameter, and recall that $d(i,j)$ is the distance between nodes i and j in the graph. We use this noise model because we consider the fact that similar nodes will have less noise, but nodes that are far apart have more room for error. It is reasonable that the noise level would be proportional to the distance between the nodes.

We constructed simulated networks with various levels of overlapping. We then tested the SDI algorithm for efficiency and accuracy. The algorithm proved to be remarkably efficient, even with large DAGs (See Figure 6), which can be attributed to the low computational complexity of our methods.

We use the following error metric to test the accuracy of the algorithm. We compute the distance matrix D_{inf} of the inferred DAG G and compare it to the distance matrix D_{true} of the original DAG G_{true}. We add the differences between corresponding elements of the two matrices, and then scale this number by the number of elements in both of the matrices

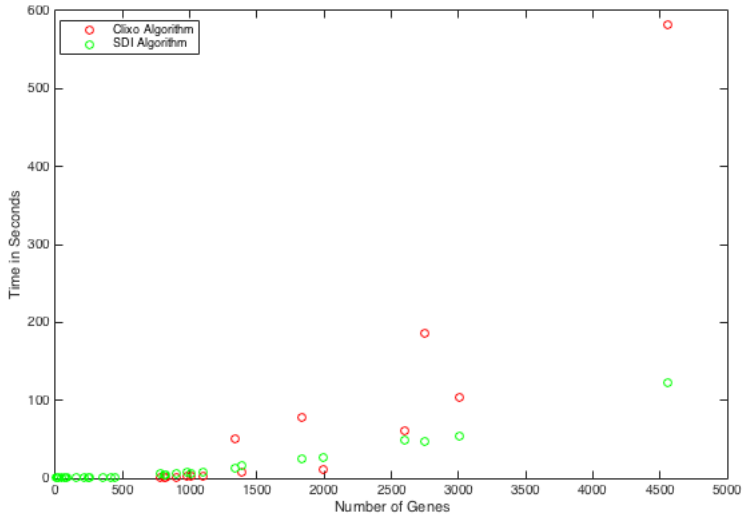

Figure 8: Time in seconds needed to infer directed acyclic graphs with varying numbers of terminal nodes.

D_{inf} and D_{true}, as well as the maximum distance in D_{true}, to account for the different sizes of graphs evaluated.

For smaller noise levels, the inferred DAG matches closely with the original DAG (See Figure 7). However, for noise parameter β greater than or equal to approximately 0.15, noise levels create more significant error.

3.3.1 Comparison to Current Methods

We compared the performance of the SDI algorithm and the performance of the Clixo algorithm presented in [3]. We see that the algorithms perform comparably over small networks. The Clixo algorithm performs faster with networks under around 1500 genes. However, the SDI algorithm takes significantly less time when inferring larger networks. We can see from Figure 8 that the SDI algorithm has a greatly improved computational complexity over the Clixo algorithm, as estimated previously. In regards to accuracy, both algorithms performed well when applied to small networks. We used the following error metric to compare the two

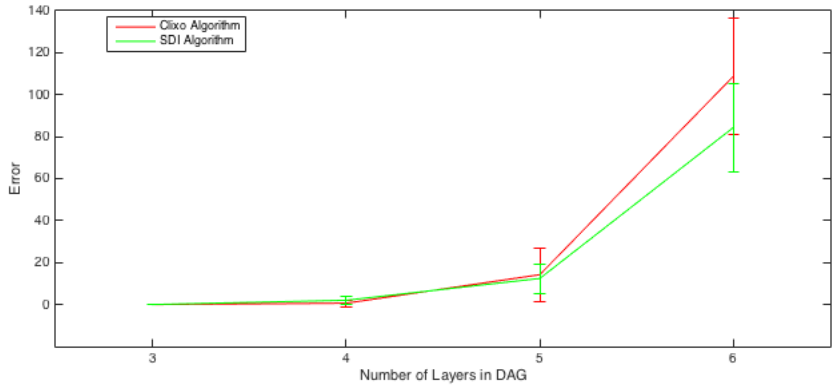

Figure 9: Performance of SDI and Clixo algorithms when tested against the same set of directed acyclic graphs. As the size of the graphs increase, the amount of error in both algorithms increase. Also observe that the SDI algorithm shows a significant reduction in error of the Clixo algorithm.

algorithms: we compared the ranks of the similarity matrices of the inferred DAGs with the ranks of similarity matrices of the actual DAG. In other words, we sorted the values in each column of the similarity matrices and used the rank of the values to compare the accuracy of the algorithms. We check to see if genes that are very similar in S_{true} are also similar in the similarity matrices of the inferred DAGs. One can see in Figure 9 that the SDI algorithm improves on the accuracy of the Clixo algorithm with large networks.

4 Conclusions and Future Work

We introduced methods to infer a layered DAG using pairwise similarity data of the terminal nodes. Our Spectral DAG Inference algorithm significantly reduces the computational complexities of existing DAG-inference algorithms by using a spectral clustering method to identify clusters among the terminal nodes instead of the expensive maximal clique algorithm. Moreover, our algorithm is a top-down algorithm, rather than a bottom-up algorithm, like the one presented in [3]. By moving down the DAG, we restrict quasi-clique inference to smaller and smaller subgraphs, and thus, we only require a partial eigen decomposition

of the similarity matrix of the terminal nodes.

Some techniques can be used to approximate the top eigenvectors of submatrices of a matrix using the eigenvectors of the full matrix. This is an interesting future direction, and would further improve the computational complexity of DAG-inference. Since our algorithm is fast (i.e. it can infer a 3,000 gene DAG in approximately 100 seconds), we can apply the SDI algorithm to large gene datasets. Our SDI method currently infers DAGs well under moderate noise levels, and we plan to investigate ways to further improve the performance of the SDI algorithm under higher levels of noise. We believe that our Spectral DAG Inference algorithm can significantly accelerate current gene ontology development, enhancing these tools for gene classification and study.

Acknowledgements

I would like to thank the MIT PRIMES leaders who run this wonderful program and assigned me a great mentor, Soheil Feizi. Thank you to Prof. Manolis Kellis for thinking of this project, and thank you to Soheil for meeting with me weekly, giving advice, and guiding me throughout this process.

References

[1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig *et al.*, "Gene ontology: tool for the unification of biology," *Nature genetics*, vol. 25, no. 1, pp. 25–29, 2000.

[2] G. O. Consortium *et al.*, "The gene ontology (go) database and informatics resource," *Nucleic acids research*, vol. 32, no. suppl 1, pp. D258–D261, 2004.

[3] M. Kramer, J. Dutkowski, M. Yu, V. Bafna, and T. Ideker, "Inferring gene ontologies from pairwise similarity data," *Bioinformatics*, vol. 30, no. 12, pp. i34–i42, 2014.

[4] K. Dolinski and D. Botstein, "Automating the construction of gene ontologies," *Nature biotechnology*, vol. 31, no. 1, pp. 34–35, 2013.

[5] M. Luby, "A simple parallel algorithm for the maximal independent set problem," *SIAM journal on computing*, vol. 15, no. 4, pp. 1036–1053, 1986.

[6] D. Karger, R. Motwani, and M. Sudan, "Approximate graph coloring by semidefinite programming," *Journal of the ACM (JACM)*, vol. 45, no. 2, pp. 246–265, 1998.

[7] J. M. Byskov, "Enumerating maximal independent sets with applications to graph colouring," *Operations Research Letters*, vol. 32, no. 6, pp. 547–556, 2004.

[8] R. E. Miller and D. E. Muller, "A problem of maximum consistent subsets," IBM Research Report RC-240, JT Watson Research Center, Tech. Rep., 1960.

[9] J. W. Moon and L. Moser, "On cliques in graphs," *Israel journal of Mathematics*, vol. 3, no. 1, pp. 23–28, 1965.

[10] F. Eisenbrand, G. Oriolo, G. Stauffer, and P. Ventura, "The stable set polytope of quasi-line graphs," *Combinatorica*, vol. 28, no. 1, pp. 45–67, 2008.

[11] M. Chudnovsky and A. Ovetsky, "Coloring quasi-line graphs," *Journal of Graph Theory*, vol. 54, no. 1, pp. 41–50, 2007.

[12] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer, "Fast deterministic distributed maximal independent set computation on growth-bounded graphs," in *Distributed Computing*. Springer, 2005, pp. 273–287.

[13] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[14] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Applied statistics*, pp. 100–108, 1979.