Detangling PPI Networks to Uncover Functionally Meaningful Clusters

Senior Honors Thesis

submitted by

Sarah Hall-Swan, Degree

In partial fulfillment of the requirements
for the degree of

Bachelor of Science

in

Computer Science

TUFTS UNIVERSITY

May 2018

ADVISOR: Prof. Lenore Cowen

Acknowledgments

We thank the Tufts BCB group for helpful discussions, and the organizers of the CNB-MAC workshop, where preliminary results were presented, for helpful feedback.

Portions of this work have been previous published as Detangling PPI Networks to Uncover Functionally Meaningful Clusters in BMC Systems Biology. I thank my co-authors Jake Crawford, Rebecca Newman, and Lenore Cowen for their permission to include our joint published work in this thesis.

Sarah Hall-Swan

TUFTS UNIVERSITY

May 2018

Detangling PPI Networks to Uncover Functionally Meaningful Clusters

Sarah Hall-Swan

ADVIS0R: Prof. Lenore Cowen

We compare computational methods for decomposing a PPI network into non-overlapping modules. A method is preferred if it results in a large proportion of nodes being assigned to functionally meaningful modules, as measured by functional enrichment over terms from the Gene Ontology (GO). We compare the performance of three popular community detection algorithms that produce non-overlapping clusters with the same algorithms run after the network is pre-processed by removing and reweighting based on the diffusion state distance (DSD) between pairs of nodes in the network. We call this detangling the network. In some cases, we find that detangling the network based on the DSD distance reweighting provides more meaningful clusters. We look at extending to methods that produce overlapping clusters.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Network Clustering

Clustering of protein-protein interaction networks is one of the most common approaches to predicting modules of genes and proteins that work together in functional roles [SS09]. However, the low network diameter and dense interconnection structure in these networks confounds a notion of local neighborhood in these networks; it is difficult to partition a network into clusters representing local neighborhoods when the network best resembles a tangled hairball, and most nodes are close to all other nodes in shortest path distance, a problem termed the "ties in proximity problem" by Arnau et al [AMM05]. There are nonetheless many notions of clustering that have been developed for the so-called "community detection" problem in biological or social networks; many of them seek to maximize the *modularity* of the clusters, a quantity defined by Girvan and Newman [GN02] that measures the relative denseness of interconnections within a cluster as compared to the connection of that cluster to the rest of the network, or alternatively the *conductance* of the clusters [VM03]. Other clustering methods have been proposed based on random walks, successive removal of cut edges, spectral embeddings and so on [For10, LLM10, HBG$^+$14].

In 2013, Cao et al. introduced a new distance measure called Diffusion State Distance, or DSD, designed to be a more fine-grained distance measure for protein-

protein interaction networks [CZP+13]. In contrast to the typical shortest path metric, which measures distance between pairs of nodes by the number of hops on the shortest path that joins them in the network, DSD was shown to spread out the pairwise distances, making for a more fine-grained notion of graph local neighborhood. We hypothesized that re-embedding the PPI network by first reweighting its edges according to their DSD distance in the original network might lead to better clusters. Before we can test this hypothesis, however, we need to think about how to measure the overall quality of a set of clusters: only then can we talk about once method producing *better* clusters than some other method.

1.1.1 Measuring quality of a clustering

In the current study, we consider the problem of separating the yeast protein-protein association network (as downloaded from the STRING database version 10 on 2/7/2017 [Szk15]) into non-overlapping clusters. Some proposed ways to measure the quality of a clustering are purely graph-theoretic, based on minimizing quantities such as *modularity* or *conductance*. In this study, instead, we wish to judge the quality of the clustering we obtain by how "meaningful" the clusters are biologically– where the standard way to measure this would be based on measuring functional enrichment of the resulting clusters. In this study, we measure functional enrichment of the clusters over the GO using the FuncAssociate tool [BBC+09], with appropriate multiple testing correction for the number of clusters in our set. We declare a cluster to be functionally enriched if it is enriched for at least one and no more than 50 different GO terms, at an appropriate level of specificity in the GO hierarchy.

However, while it is easy to declare one particular cluster to be known to be meaningful if it is enriched for at least one and no more than 50 biological functions, it is not immediately clear how to use this to compare the overall quality of different clusterings, particularly when the number and distribution of cluster sizes is different across the different clustering algorithms. Observe that in particular, the *percentage of enriched clusters* is not a good statistic: any algorithm that picks off small good clusters around the periphery of the network, and then puts all the remaining nodes into a giant single cluster in the center, will score all but one of its clusters enriched (the large center cluster), for a very large percentage of enriched clusters. Restricting the maximum size of a cluster (as we do for some of the experiments) can ameliorate this behavior to a large extent, but we still are faced with the need to find a meaningful overall statistic even when the distribution of cluster sizes is highly non-comparable.

When we are restricting ourselves to *non-overlapping* clusterings, we choose as the main statistic by which we judge the quality of a clustering to be *the number (or percent) of network nodes that are placed within enriched clusters.* We abbreviate this as $\#NEC$ and $\%NEC$. We note that this NEC statistic can be measured across clusterings with different numbers of clusters, size of clusters, and different cluster size distributions. However, even these NEC statistics are most meaningful when comparing clusterings when the number of clusters and their ranges of sizes are approximately matched; in particular, adding some number of unrelated nodes arbitrarily to an enriched cluster will improve the NEC statistics, even if it dilutes the cluster enrichment, as long as it doesn't cause the enrichment to dip below the enrichment threshold. See figure 2.1 for a simple example demonstrating this case.

Thus we add a second statistic that we call NEC S (for *number of enriched clusters, same label*), for the number (or percent) of nodes whose label *matches* a label of its enriched cluster. This is a more stringent condition met by a smaller number of nodes in enriched clusters and more precisely measures how well our clustering recapitulates existing knowledge. In the case where there is no bound on cluster sizes, this is the more meaningful statistic, because the ordinary NEC statistics will tend to inflate the quality of the clustering. Figure 2.2 shows the NEC S statistic computed on an example cluster.

1.1.2 Overlapping Clusters

Many proteins have more than one function and belong to multiple functional groups. For biological problems strict partitioning of a graph is not an accurate method of predicting protein function because it forces each protein into only one cluster. Nodes that share a community are more densely connected to each other than to nodes outside their community, and this logic can be extended to say that nodes in the overlaps between communities are more connected to each other than nodes in non-overlapping parts [YL13, BRCG12]. Therefore, techniques that allow for overlapping clusters are more useful for identifying protein functions.

To measure the quality of overlapping clusters, we penalize each node based on the number of clusters it is in. Each node is worth 1/(number of clusters), such that a node in only one cluster is worth 1, a node in 2 clusters is worth 0.5, and so on. We then calculate the same statistics as decribed above, where the number of nodes is instead the combined worth of the nodes.

Review of DSD

Consider the undirected graph $G(V, E)$ on the vertex set $V = \{v_1, v_2, v_3, ..., v_n\}$ and $|V| = n$. Now $He^{\{k\}}(A, B)$ is defined as the expected number of times that a simple symmetric random walk starting at node A and proceeding for some fixed k steps (including the 0th step), will visit node B.

We now take a global view of the $He^k(A, B)$ measure from each vertex to all the other vertices of the network.

More specifically, we define a n-dimensional vector $He^k(v_i), \forall v_i \in V$, where

$$He^k(v_i) = (He^k(v_i, v_1), He^k(v_i, v_2), ..., He^k(v_i, v_n)).$$

Then, the Diffusion State Distance (DSD) between two vertices u and v, $\forall u, v \in V$ is defined as:

$$DSD^k(u, v) = ||He^k(u) - He^k(v)||_1.$$

where $||He^k(u) - He^k(v)||_1$ denotes the L_1 norm of the He^k vectors of u and v.

We showed in [CZP$^+$13] for any fixed k, that DSD is a true distance metric, namely that it is symmetric, positive definite, and non-zero whenever $u \neq v$, and it obeys the triangle inequality. Thus, one can use DSD to reason about distances in a network in a sound manner. Further, we showed that when the network is ergodic, DSD converges as the k in $He^{\{k\}}(A, B)$ goes to infinity, allowing us to define DSD independent from the value k, and to compute the converged DSD matrix tractably, with an eigenvalue computation, where we can compute

$$DSD(u, v) = ||(1_u - 1_v)(I - D^{-1}A + W)^{-1}||_1$$

where D is the diagonal degree matrix, A is the adjacency matrix, and W is the constant matrix where each row is a copy of π, the degrees of each of the vertices, normalized by the sum of all the vertex degrees.

The above treatment does not consider edge weights; DSD was generalized to handle edge-weighted graphs in [CPF$^+$14]. To incorporate edge weights, the random walk is modified where instead of choosing all edges at a vertex with equal probability, the walk instead chooses edges in proportion to their confidence weights, namely we define a new 1-step transition matrix with (i, j)th entry given by:

$$p'_{ij} = \frac{w_{ij}}{\sum_{l=1}^{n} w_{il}}$$

Then we redefine $He^k(A, B)$ as the expected number of times that the weighted random walk starting at node A and proceeding for k steps will visit B, which can be calculated as the (i, j)th entry of the kth power of the transition matrix. The n-dimensional vector $He^k(v_i)$ can be constructed as before, and then the DSD is calculated the same as before, just based on the modified He vectors.

1.2 Outline of This Work

Following is the outline of individual chapters in this thesis.

In Chapter 2, we show the effect of DSD on some popular clustering algorithms.

In Chapter 3, we discuss possible algorithms for producing overlapping clusters, and the effect of DSD on those methods.

Chapter 2

Non-Overlapping Clusters

Methods

The network

The protein-protein association network for *S. Cerevisiae* was downloaded from STRING version 10 on 2/7/2017 [Szk15]. We removed all edges that had no direct experimental verification. Edge weights were taken directly from from the "escore" confidence values given by STRING. There are 2 nodes that are isolated from the rest of the network, and after we remove them the network has 6096 nodes.

Enrichment calculation

Functional enrichment was measured in Gene Ontology terms using the FuncAssociate 3.0 web API [BBC+09]. All GO terms that were level 5 or below in specificity from all three hierarchies (molecular function, biological process, and cellular component) were considered. FuncAssociate uses Fisher's exact test to calculate an enrichment p-value, and we used a p-value cutoff of 0.05 to determine if a cluster was significantly enriched for a term. To correct for multiple testing, FuncAssociate uses an approach based on Monte Carlo sampling from the background gene space, as described in [BBC+09] (note that because of the stochastic sampling, different runs of FuncAssociate can give slightly different results, but we mostly observe differences of only fractions of a percentage point).

The clustering algorithms

We considered the following popular clustering algorithms, each of which will return a non-overlapping set of clusters. In our study, we restricted cluster sizes to be at least 3; any cluster of size less than 3 created by an algorithm was discarded. We considered all three algorithms with no restriction on maximum cluster size; we then modified each of the three algorithms to set a maximum cluster size of 100. Bounds on minimum and maximum cluster size were set in order to make the clusterings returned by different methods more comparable; the specific values of 3 and 100 were set to be consistent with the recent DREAM community "disease module identification" challenge [Con16]. For each clustering method, we run it natively on the network from STRING. We then run it on a transformed network, preprocessed with DSD as follows: 1) We form the DSD matrix of distances in the original network. 2) We create a new graph by placing edges between pairs of nodes whose DSD distance is less than r, with edge weight $1/r$. We then run the clustering algorithm on the new DSD-based detangled graph. We considered a range of different values of the threshold r (between 4 and 6).

The Louvain Algorithm

For a partition of a network into two pieces, consider the quantity

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where A_{ij} is the matrix of edge weights, m is the sum of all the edge weights, $k_i = \sum_j A_{ij}$ is the sum of all the edge weights emanating from vertex i and δ is an indicator function that is 1 iff i and j have been placed in the same cluster. Then Q measures the *modularity* in a weighted graph, based on the weight of links within a cluster as compared to the links between clusters (see [GN02]).

The Louvain Algorithm, first defined in [BGLL08], is a heuristic that repeatedly tries to move individual nodes across cluster boundaries in order to improve the value of Q. Starting from a partition of the network into clusters (initially, every node is placed into its own cluster), the first phase of the Louvain algorithm considers nodes i that are adjacent to some node j which has been placed in a different community. i is moved into j's community if and only if doing so would increase the modularity Q described above. Nodes are considered multiple times until the quantity Q can no longer be improved by moving any individual nodes. The second phase of the algorithm consists in building a new network whose nodes are now the communities found during the first phase. The weights between these new supernodes are now set to be the sum of the weight of the links between nodes in the corresponding two communities (where links between nodes of the same community are retained as self-loops). Then the first phase of the Louvain algorithm is run again on the new nodes.

In our implementation, clusters with fewer than 3 nodes were discarded. We also modified the Louvain algorithm to force clusters to have at most 100 nodes by re-running Louvain separately on each cluster with more than 100 nodes, in order to split the cluster into multiple clusters of size under 100 nodes.

The Walktrap Algorithm

Consider the random walk on G where at each time step, the walker moves from a node to a new node chosen randomly and uniformly among its neighbors (with probability in proportion to edge weights). When D is the matrix that has the ith diagonal entry be the degree of vertex i, and 0's off the diagonal, then one can define the transition matrix of the random walk as $P = D^{-1}A$ where A is the adjacency matrix. Fix t, the length of a random walk and let $P^t_{i\circ}$ denote the ith row of the matrix P^t The Walktrap algorithm [PL06] defines an an (i, j) distance $r_{i,j}$ depending on the L_2 distance between the two probability distributions $P^t_{i\circ}$ and $P^t_{j\circ}$. This internode distance is then generalized to a distance between communities in a straightforward way, by choosing a starting node randomly and uniformly among the nodes of the community. This defines the probability $P^t_{C_j}$ to go from community

C to vertex j in t steps and an associated probability vector $P^t_{C_j \circ}$. Then the distance $r_{C_1C_2}$ is defined as the L_2 distance between the two probability distributions $P^t_{C_1\circ}$ and $P^t_{C_2\circ}$..

This algorithm is initialized by putting each vertex into its own cluster. Then two adjacent communities (joined by at least one edge) are merged according to which gives the lowest value of the quantity $\Delta\alpha$, where the change in $\Delta\alpha$ that would result when clusters C_1 and C_2 are instead merged into a new cluster C_3 is given by:

$$\Delta\alpha(C_1, C_2) = \frac{1}{n}\frac{|C_1||C_2|}{|C_1| + |C_2|} r^2_{C_1C_2}$$

Walktrap can produce clusters of size > 100. We therefore also consider a modified version of Walktrap (again setting t=4) that prevents the merging clusters if the merge would create a cluster of of size > 100. Modified Walktrap is run until no more merges are possible, which can be represented as a forest dendrogram (not a tree, because there are multiple clusters at the top level that cannot merge because their union would contain more than 100 nodes). We then cut the dendrogram at a lower level to produce some lower number of output clusters: the final number of clusters output is all the clusters at that level of size ≥ 3 (discarding clusters of size 1 or 2).

Spectral Clustering

Spectral Clustering was introduced by Ng, Jordan and Weiss [NJW$^+$01] in 2001. It takes as input a similarity matrix, and does a low-dimensional embedding of the nodes according to that similarity matrix. Then K-means clustering is run on the nodes in the embedded space, where K, the number of clusters, is an input to the algorithm. In our case we construct the similarity matrix by computing 1/(the DSD distance). The final number of clusters we produce is not K, since we discard any cluster of size < 3. We consider also a modified version of spectral clustering where we recursively split any cluster of size > 100, recursively calling spectral clustering with $K = 2$ clusters, until all cluster sizes are less than 100 nodes.

Clustering Implementations

In the case of Louvain, we used the implementations in the popular igraph package [CN06]. In the case of spectral clustering, our implementation came from scikit-learn [PVG+11]. In the case of Walktrap, we used the Walktrap source code from [PL06], and for the modified Walktrap algorithm (which restricted cluster sizes to be < 100 nodes), we worked directly from the Walktrap source code.

Results

For each algorithm we consider, we compare what would be obtained by running that algorithm directly on the PPI network with weights taken directly from the STRING confidence values, with no filtering or pre-processing, to what is obtained by first running DSD on the network, filtering out edges where the DSD distance between their endpoints exceeded a threshold, and otherwise running the algorithm with edges weighted by 1/(DSD distance).

We first considered the Louvain algorithm without any restriction on maximum cluster size. The Louvain algorithm is highly sensitive to the order in which nodes are considered [BGLL08], so we report median results over 10 independent runs of the algorithm (mean results over the 10 runs are highly similar and not shown). Louvain creates a multilevel structure of clusters, so we report the results of the first level of clusters created because the clusters were the smallest. The results appear in Tables 2.1 The best results occur when the network is pre-processed with DSD at an appropriate threshold, however, run directly on the PPI network as well as some of the DSD thresholds, these algorithms unmodified produce some large, uninformative clusters. For example, in every one of the 10 times we ran Louvain directly on the PPI network, the largest cluster had size greater than 1000 nodes.

We also considered the Walktrap algorithm without any restriction on maximum cluster size. These results appear in Table 2.3. The algorithm outputs a dendrogram that can be cut at different levels, so we sought to control the number of clusters by cutting the dendrogram at different levels. The dendrogram level corresponds to the number of clusters before removing clusters of size < 3, so the final number of clusters may be less than the dendrogram cut level. The results appear in Table 2.5. For both $\%NEC$ and $\%NECS$, Walktrap with DSD performs better than Walktrap run on the unproccessed PPI network for every dendrogram cut level. Walktrap performs best when at the dendrogram cut level 700.

We also considered modified versions of Louvain and Walktrap, as described above, that force cluster sizes between 3 and 100 nodes (where again, the specific values of 3 and 100 were set to be consistent with the recent DREAM community "disease module identification" challenge [Con16].) Louvain also creates a multilevel structure of clusters, so we report the results of just the first level of clusters created because the clusters were the smallest, and the level with the best modularity. These results appear in Tables 2.2 and 2.4. DSD plus Louvain performs worse than Louvain alone, with bounded cluster sizes. Note that while modified Louvain can bound cluster sizes, and we can choose which level of the final hierarchical strucure we can use, it really has no way to tune the exact number of clusters that are output by the algorithm. On the other hand, the number of clusters that are output by modified Walktrap can be tuned by cutting the cluster dendrogram at different levels.

Thus, in order to explore our chosen measure of cluster quality, namely, the percent of the 6096 network nodes placed into an enriched cluster of size between 3 and 100 further, for Walktrap modified to have bounded cluster size run directly on the PPI network versus run after pre-processing with various DSD thresholds, we explored cutting the Modified Walktrap dendrogram at different numbers of clusters (before filtering small clusters, so the resulting numbers of clusters may not necessarily be exactly the same as the dendrogram cut level). The results appear in Table 2.6 for both the $\%NEC$ and $\%NEC\ S$ statistics. For the $\%NEC$ statistic, the modified Walktrap algorithm with DSD preprocessing performs better for every dendrogram cut level. For the $\%NEC\ S$ statistic, the algorithm with DSD preprocessing performs better for lower dendrogram cut levels (i.e. fewer clusters), but for a dendrogram cut level of 700, the algorithm run directly on the PPI network performs better, although DSD with a cutoff of 5.5 performs nearly comparably for this statistic.

Figure 2.3 gives some intuition for how the DSD thresholds were chosen: it shows a histogram of all pairwise DSD distances between nodes in the PPI network; setting the DSD threshold removes a fraction of these edges and sparsifies the network. For example, setting the edge removal threshold to 4.5 will result in direct edges from a vertex only to a small fraction of its close neighbors in DSD distance. Setting the edge removal threshold to 6, on the other hand, preserves roughly half the pairwise network distances.

Figure 2.4 directly compares the clusters at different size ranges by enrichment for Louvain directly, and DSD followed by Louvain, with an edge removal threshold of 5, and cluster sizes bounded to lie between 3 and 100. Detangling with DSD decreased the percentage of nodes placed within enriched clusters.

We next sought to make the comparison for spectral clustering, but spectral clustering has an additional parameter that must be set, namely K, the number of clusters. We look at both a version of spectral clustering that does not restrict maximum cluster size, as well as a variant of spectral clustering that recursively splits clusters of size greater than 100, in order to produce a clustering with clusters of size between 3 and 100 nodes, as before. Note that the final number of clusters output by our spectral clustering method will be different than K, the input number of cluster centers, because our implementation of spectral clustering recursively splits any cluster of size > 100. Figure 2.5 shows that the number of clusters that spectral clustering plus DSD (modified to force a maximum cluster size of 100) produces based on the number of input clusters is robust to the threshold cutoff. In all cases, the number of output clusters rises for awhile based on the number of input cluster centers, and then falls off. It rises compared to the number of input clusters when cluster sizes are too large and get split by our method for having > 100 nodes. It falls off when K is set large enough that many of the clusters that spectral clustering produces have < 3 nodes, which we then discard and do not include as output clusters according to the cluster size restrictions of our methods. Based on this figure, we report results for $K = 300$ at different DSD thresholds in Tables 2.7 and 2.8.

Figure 2.6 gives the number of clusters and the percentage of enriched clusters for spectral clustering (with a maximum cluster size bounded at 100) and DSD+spectral clustering for $K = 300$. As can be seen, DSD+spectral clustering has a higher percentage of nodes in enriched clusters than spectral clustering using both NEC and NEC S statistics when cluster sizes are bounded by 100; with unbounded cluster sizes, the results are mixed: the NEC statistic is better for spectral run directly on the PPI network than DSD+spectral, because more nodes are placed into a large enriched central cluster. However, DSD+spectral is better for the NEC S statistic, which is the more informative statistic in the case of unbounded cluster sizes. On the human network, DSD+spectral outperforms spectral run natively on the PPI network by every statistic (see Discussion).

Discussion

We have shown that some popular clustering methods appear to perform better when DSD is applied as a pre-processing step to help detangle the network. In particular, we tested Louvain, Walktrap, and Spectral Clustering methods, both native as well as modified to keep the maximum cluster size bounded by 100 nodes, run on the yeast PPI network directly, and then run on the PPI network after using DSD to sparsify and detangle the network, for a total of 6 different methods. For four of the six methods, applying the DSD pre-processing method at an appropriate threshold improved the percentage of network nodes that were placed into clusters enriched for their own functional label. For the fifth method, spectral clustering with no modification to large clusters, the DSD detangling sometimes improved performance slightly or sometimes hurt performance slightly, depending on other parameter settings. For the sixth method, Louvain with bounded cluster sizes, the DSD detangling was inferior to running Louvain directly on the PPI network. Measuring the number of nodes placed into enriched clusters (not necessarily enriched for their own label) showed similar trends regardless of whether or not we filtered out the most general GO terms; these statistics were also often improved at the appropriate DSD threshold when sizes and and number of clusters were approximately matched.

It is hard to definitively answer which of the six methods is best, since it is hard to control the range of cluster sizes exactly. With both bounded and unbounded cluster sizes, it is not clear which method has best performance overall. Spectral clustering plus DSD and Louvain alone, both modified to bound maximum cluster sizes, are able to produce an impressive percent of nodes in enriched clusters, in a setting where it is very easy to control the and size range of the clusters that are returned. For spectral clustering, it is also easy to control the number of clusters returned. For this reason, the spectral clustering method was probably our favorite, though all three modified algorithms also performed quite well, both with and without DSD.

It is natural to ask if our results were peculiar to the yeast network, or whether they would generalize to other organisms. We were particularly interested in the human network, which has more nodes but is more sparsely annotated. We thus also downloaded the protein-protein interaction network for *H. sapiens* from STRING version 10 on 2/7/2017. As before, we removed all edges that had no direct experimental verification. Edge weights were taken directly from the 'escore' confidence values given by STRING. In the human network, we consider only the largest connected component which has 15,129 nodes.

Because there are fewer known edges and this is a sparser network than yeast, we set higher DSD thresholds, ranging from 6 to 8. See Figure 2.7 for the corresponding histogram of all pairwise DSD distances in this network.

As can be seen in Table 2.9, the advantages of detangling the network with DSD before applying Spectral clustering seem even clearer on the human network. For both of the $\%NEC$ thresholds, and robust to the exact value of the DSD cutoff, results are better when the network is pre-processed with DSD.

Method	Enriched Clusters	# NEC	**% NEC**	# NEC S	**% NEC S**
PPI	29.5/47.5 (62.11%)	799.0	13.10%	548.5	8.99%
4.0	130.0/192.0 (67.71%)	1144.0	18.77%	1011.0	16.58%
4.5	175.0/265.5 (65.91%)	1960.5	**32.16%**	1562.0	**25.62%**
5.0	106.5/173.0 (61.56%)	1736.0	28.48%	967.0	15.86%
5.5	15.0/45.5 (32.97%)	361.5	5.93%	288.0	4.72%
6.0	5.0/21.5 (23.26%)	221.0	3.63%	178.5	2.93%

Table 2.1: The performance of Louvain run directly on the PPI network versus Louvain plus DSD at different edge removal thresholds; the reported results of Louvain are median values from running the algorithm over 10 random permutations of the nodes. We discard clusters of size < 3. NEC= "Nodes in Enriched Clusters". We calculate $\%NEC$ in two settings: $\%NEC$ is enrichment in the GO hierarchy with terms above the fifth level filtered out, and $\%NEC\ S$ uses the same filtered GO hierarchy, but then only gives a node credit if there is a match between one of the node's labels and one of the terms for which there is GO enrichment for the cluster. Note that without modifying Louvain to restrict the maximum cluster size, the S statistic is the most meaningful. Running directly on the PPI network and run with high DSD thresholds, Louvain produces a relatively small number of clusters, and many are of very large size. It is worth noting that with a DSD threshold of 5, nearly 175 clusters are produced, and the enrichment statistics remain reasonable.

Many open questions still remain. One way in which our problem formulation was somewhat artificial is that we required our clusters to be *non-overlapping*; however, many proteins participate in multiple pathways, complexes or processes, which would be more accurately represented by overlapping clusters or communities. The next chapter explore possible methods of producing overlapping clusters.

Tables

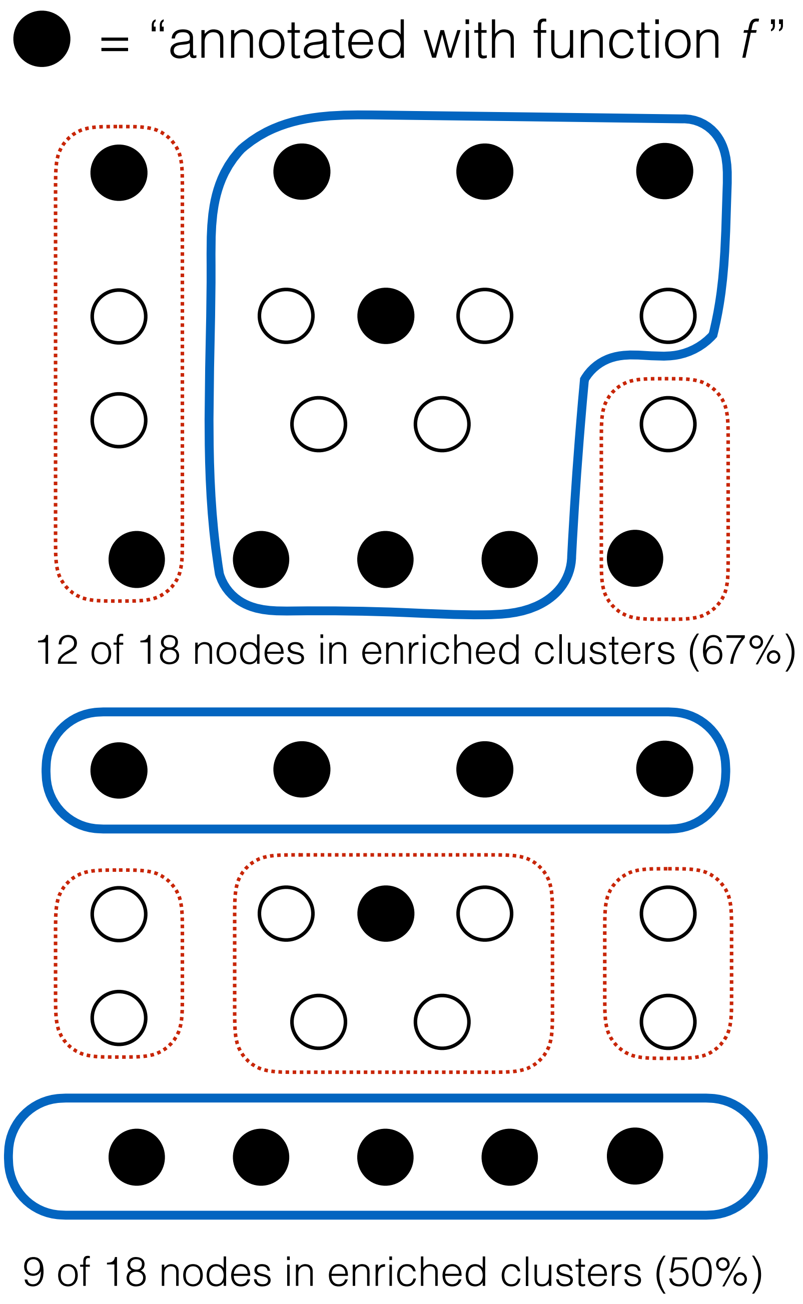
= "annotated with function *f*"
12 of 18 nodes in enriched clusters (67%)
9 of 18 nodes in enriched clusters (50%)

Figure 2.1: Comparison of two example network partitions under the NEC statistic. Edges are omitted for visual clarity and only a single function f is considered in this simple case. The clusters outlined in bold blue are "enriched" and those outlined in dotted red are not. Although the lower partition is more specific for f (i.e. its enriched clusters contain fewer false positives), by the NEC statistic it does not score as well as the upper partition. Note that in this case, the distribution of cluster sizes is indeed much different between partitions; that is, the upper partition has a single giant cluster, and the lower partition contains clusters having a more uniform size distribution.

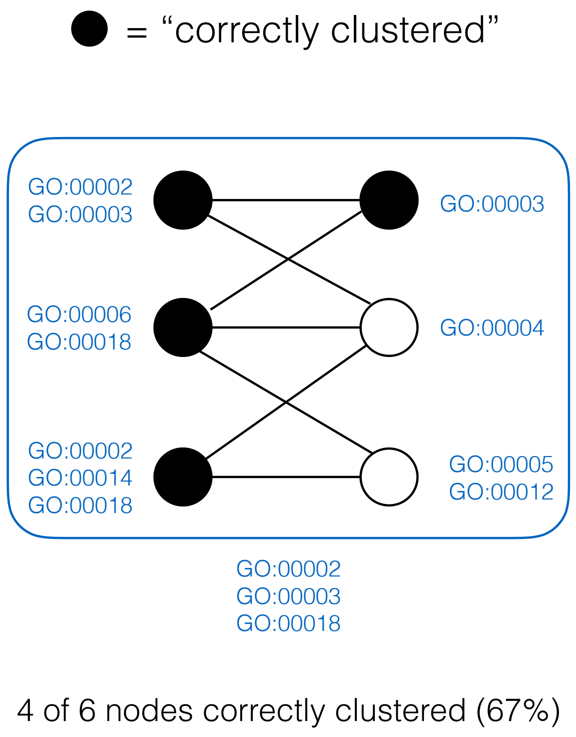

Figure 2.2: Example of scoring a single cluster using the NEC S statistic. GO annotations are listed for each node and for the cluster as a whole, and only those nodes with an annotation matching the cluster (the shaded nodes) are counted. In this case, 4 of the 6 total nodes (67%) are correctly clustered.

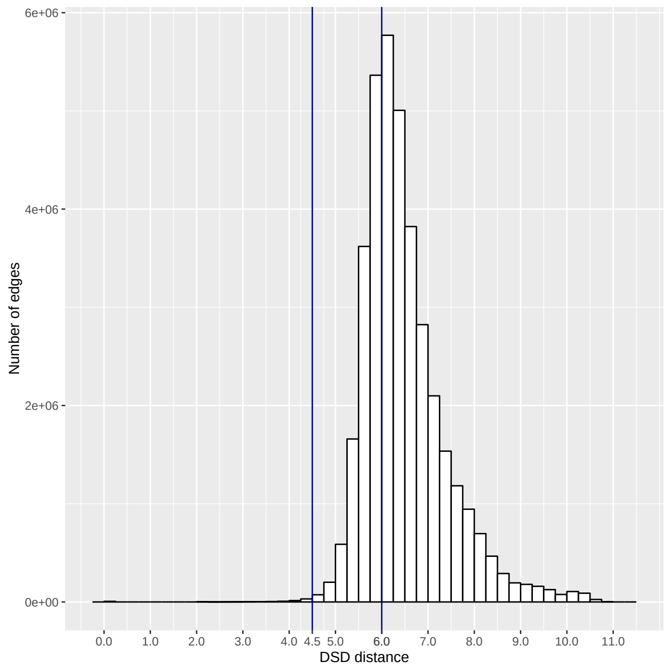

Figure 2.3: Histogram of all DSD distances in the STRING PPI network for yeast; edge removal thresholds of 4.5 and 6.0 are marked.

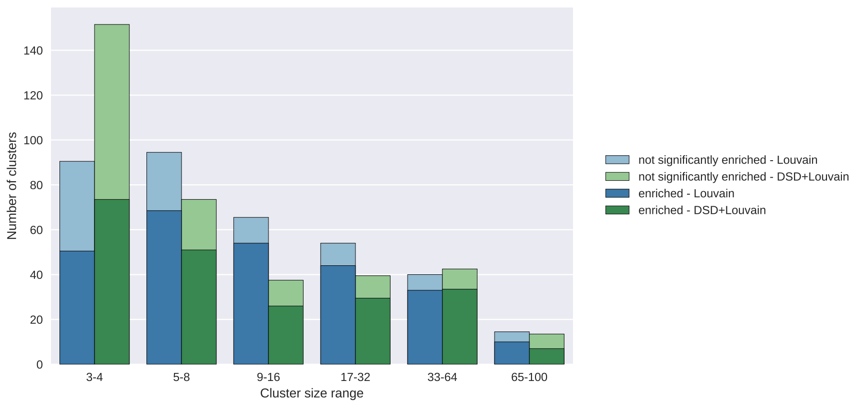

Figure 2.4: This figure compares median cluster sizes running Louvain (with cluster sizes restricted to 3-100) directly on the PPI network with Louvain running on the DSD-detangled network (again with cluster sizes restricted to 3-100), with an edge removal threshold of 5.0. The overall percentage of nodes in enriched clusters is 76.21% for Louvain directly and 70.46% for DSD+Louvain.

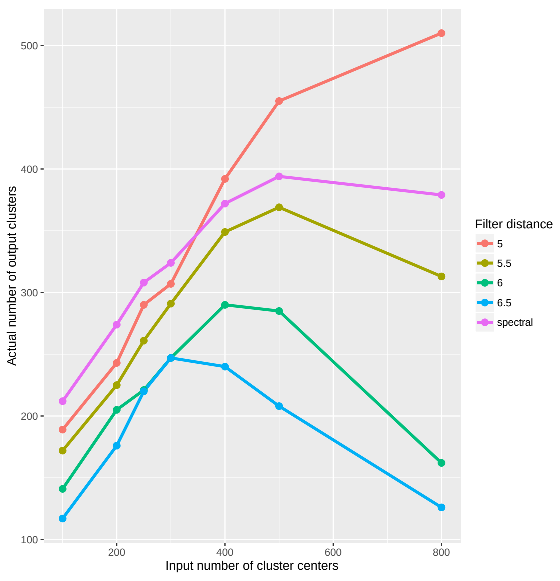

Figure 2.5: This figure plots the number of clusters output by spectral clustering and spectral clustering run on the DSD reweighted network, for different filter distance thresholds, based on the number K of clusters input to the method; in all cases, the number of output clusters starts out as less than K since clusters of size < 3 are not included in the count of output clusters. Then the number of clusters grows larger than the number of input clusters (because large clusters are recursively split) until K grows so large that the number of clusters of size < 3 counterbalances that increase.

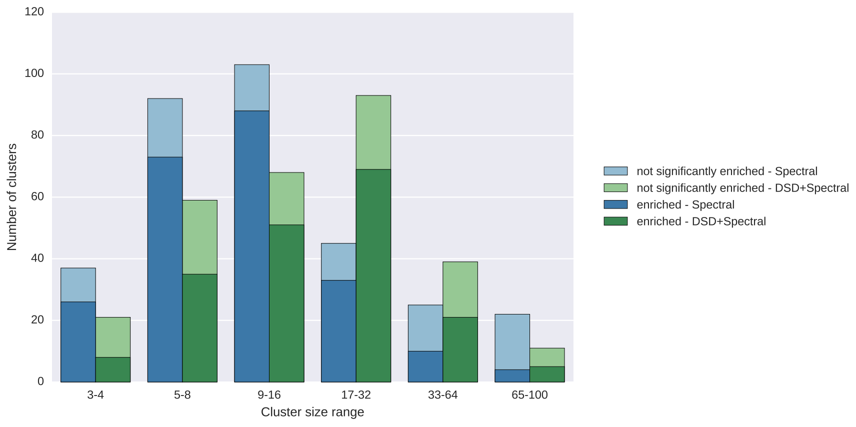

Figure 2.6: This figure compares cluster sizes running Spectral (with cluster sizes restricted to 3-100) directly on the PPI network with Spectral running on the DSD-detangled network (again with cluster sizes restricted to 3-100), with an edge removal threshold of 5.5. The percentage of nodes in enriched clusters is 50.54% for Spectral directly and 61.76% for DSD+Spectral.

Method	Enriched Clusters	# NEC	**% NEC**	# NEC S	**% NEC S**
PPI	264.5/361.0 (73.27%)	4646.0	**76.21%**	2765.5	**45.37%**
4.0	129.0/192.0 (67.19%)	1139.0	18.68%	1006.5	16.51%
4.5	207.5/304.0 (68.26%)	2220.5	36.43%	1754.0	28.77%
5.0	221.0/363.0 (60.88%)	3720.5	61.03%	2418.0	39.67%
5.5	131.0/227.0 (57.71%)	4107.5	67.38%	2380.5	39.05%
6.0	113.5/177.0 (64.12%)	4295.5	70.46%	2192.5	35.97%
PPI	161.0/200.0 (80.80%)	4944.0	**81.10%**	2697.5	**44.25%**
4.0	104.0/150.0 (69.33%)	1216.5	19.96%	996.5	16.35%
4.5	163.5/236.5 (68.89%)	2269.5	37.23%	1712.0	28.08%
5.0	167.5/256.5 (64.93%)	3909.0	64.12%	2450.5	40.20%
5.5	118.0/176.5 (65.07%)	4309.5	70.69%	2336.5	38.33%
6.0	94.0/145.0 (65.27%)	4413.5	72.40%	2263.5	37.13%

Table 2.2: The performance of Louvain versus Louvain plus DSD at different edge removal thresholds; the results of Louvain are median values from running the algorithm over 10 random permutations of the nodes. We discard clusters of size < 3 and recursively split clusters of size > 100. The values above the double line are from running Louvain and using the first level of clusters created; the values below the double line are from running Louvain and using the level with the best modularity. NEC= "Nodes in Enriched Clusters". We calculate $\%NEC$ in two settings: $\%NEC$ is enrichment in the GO hierarchy with terms above the fifth level filtered out, and $\%NEC\ S$ uses the same filtered GO hierarchy, but then only gives a node credit if there is a match between one of the node's labels and one of the terms for which there is GO enrichment for the cluster. Louvain run directly on the PPI network performs better than every DSD threshold we tested. The version that uses the level with the best modularity produces slighty better clusters than the version that uses the first level of clusters according to the $\%NEC$, but the results are more similar according to the $\%NEC\ S$.

Method	Enriched Clusters	# NEC	**% NEC**	# NEC S	**% NEC S**
PPI	71/95 (74.74%)	2388	39.17%	1309	21.47%
4.0	132/189 (69.84%)	1169	19.18%	1011	16.58%
4.5	136/195 (69.74%)	857	14.06%	703	11.53%
5.0	114/182 (62.64%)	1480	24.28%	958	15.72%
5.5	59/100 (59.00%)	1062	17.42%	754	12.37%
6.0	108/141 (76.60%)	3305	**54.22%**	1390	**22.80%**
PPI	122/165 (73.94%)	2896	47.51%	1678	27.53%
4.0	134/196 (68.37%)	998	16.37%	891	14.62%
4.5	182/264 (68.94%)	1948	31.96%	1536	25.20%
5.0	111/183 (60.66%)	1430	23.46%	957	15.70%
5.5	90.135 (66.67%)	2223	36.27%	1214	19.91%
6.0	127/175 (72.57%)	3952	**64.83%**	1690	**27.72%**

Table 2.3: The performance of Walktrap versus Walktrap plus DSD at different edge removal thresholds; we discard clusters of size < 3. The numbers above the double line are for cutting the Walktrap dendrogram at 500 clusters, and the numbers below the double line are for cutting the dendrogram at 700 clusters. NEC= "Nodes in Enriched Clusters". We calculate $\%NEC$ in two settings: $\%NEC$ is enrichment in the GO hierarchy with terms above the fifth level filtered out, and $\%NEC\ S$ uses the same filtered GO hierarchy, but then only gives a node credit if there is a match between one of the node's labels and one of the terms for which there is GO enrichment for the cluster. In both dendrogram cuts, Walktrap+DSD did better in both statistics than Walktrap directly on the PPI network, though for the cutoff of 700 clusters, Walktrap directly on the PPI network did very similarly to Walktrap+DSD according to the $\%NEC\ S$.

Method	Enriched Clusters	# NEC	**% NEC**	# NEC S	**% NEC S**
PPI	35/64 (54.69%)	3274.0	53.69%	1703.0	27.93%
3.5	56/91 (61.54%)	570.0	9.35%	468.0	7.68%
4.0	97/142 (68.31%)	1155.0	18.95%	915.0	15.01%
4.5	144/215 (66.98%)	1869.0	30.66%	1415.0	23.21%
5.0	96/174 (55.17%)	2785.0	45.69%	1724.0	28.28%
5.5	56/93 (60.22%)	4067.0	66.72%	1783.0	**29.25%**
6.0	51/81 (62.96%)	4155.0	**68.16%**	1667.0	27.35%
PPI	39/69 (56.52%)	3367.0	55.21%	1782.0	29.22%
3.5	55/91 (60.44%)	495.0	8.12%	463.0	7.60%
4.0	97/142 (68.31%)	1155.0	18.95%	915.0	15.01%
4.5	144/215 (66.98%)	1869.0	30.66%	1415.0	23.21%
5.0	95/174 (54.60%)	2686.0	44.06%	1676.0	27.49%
5.5	60/106 (56.60%)	3978.0	65.26%	1862.0	**30.54%**
6.0	66/96 (68.75%)	4077.0	**66.88%**	1680.0	27.56%

Table 2.4: The performance of Modified Walktrap versus Modified Walktrap plus DSD at different edge removal thresholds; We discard clusters of size < 3, and restrict maximum cluster size to be < 100. The numbers above the double line are for cutting the Walktrap dendrogram at 200 clusters; the numbers below the double line are for cutting the Walktrap dendrogram at 300 clusters. NEC= "Nodes in Enriched Clusters". We calculate $\%NEC$ in two settings: $\%NEC$ is enrichment in the GO hierarchy with terms above the fifth level filtered out, and $\%NEC\ S$ uses the same filtered GO hierarchy, but then only gives a node credit if there is a match between one of the node's labels and one of the terms for which there is GO enrichment for the cluster. In both cases, for the S statistic the best DSD threshold is 5.5, at which performance is slightly better than running Walktrap directly on the PPI network. For cutoffs of both 200 and 300 clusters, DSD+Walktrap is slightly better than Walktrap in the NEC measure, and in both cases the DSD version produces slightly more and smaller clusters.

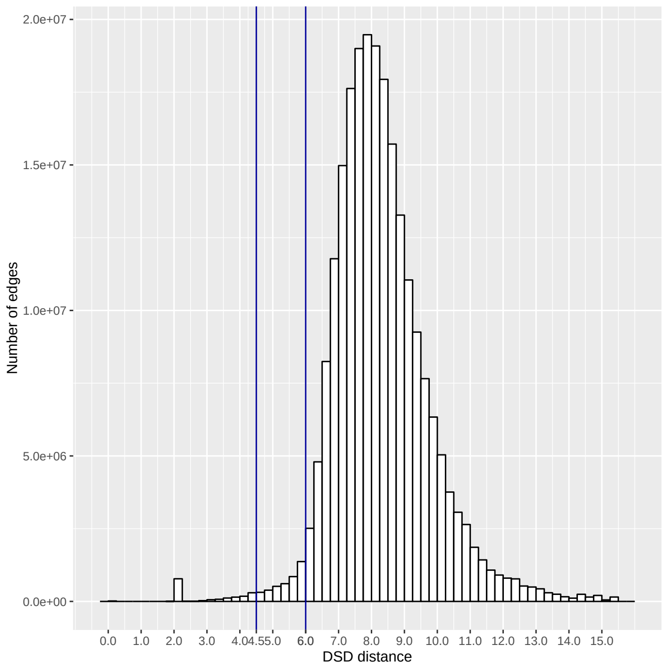

Figure 2.7: Histogram of all DSD distances in the Human STRING PPI network; previous edge removal thresholds of 4.5 and 6.0 for yeast are marked.

Dendrogram cut level	200	300	500	700
PPI	3.6%	8.9%	39.2%	47.5%
DSD 4.5	8.8%	8.8%	14.1%	31.9%
DSD 5.0	9.8%	**15.9%**	24.3%	23.5%
DSD 5.5	**13.1%**	12.8%	17.4%	36.5%
DSD 6.0	9.6%	14.6%	**54.2%**	**64.8%**
PPI	3.1%	7.7%	21.5%	27.5%
DSD 4.5	7.4%	7.4%	11.5%	21.2%
DSD 5.0	7.7%	**12.5%**	15.7%	15.7%
DSD 5.5	**10.4%**	10.4%	12.4%	19.9%
DSD 6.0	7.9%	10.5%	**22.8%**	**27.7%**

Table 2.5: Exploring the dendrogram cut level for Walktrap. Above the double line we report the percentage of nodes placed into an enriched cluster (i.e. the statistic we are calling % NEC), and below the double line we report the number of correctly clustered nodes (i.e. the statistic we are calling % NEC S). At different dendrogram cut levels, the best percentage is bolded; in every case it is Walktrap plus DSD, at varying thresholds (5, 5.5, and 6).

Dendrogram cut level	200	300	500	700
PPI	55.3%	53.6%	54.9%	55.3%
DSD 4.5	30.7%	30.7%	30.7%	30.3%
DSD 5	44.1%	44.0%	44.1%	44.2%
DSD 5.5	66.7%	66.9%	65.1%	**65.3%**
DSD 6	**72.6%**	68.3%	**66.2%**	63.0%
PPI	29.0%	28.0%	30.2%	**32.3%**
DSD 4.5	23.3%	23.2%	23.2%	24.5%
DSD 5	27.3%	27.5%	27.4%	28.9%
DSD 5.5	**29.6%**	**31.5%**	**30.6%**	31.8%
DSD 6	28.4%	27.8%	27.5%	24.8%

Table 2.6: Exploring the dendrogram cut level for modified Walktrap with a maximum cluster size of 100. Above the double line we report the percentage of nodes placed into an enriched cluster (i.e. the statistic we are calling % NEC), and below the double line we report the number of correctly clustered nodes (i.e. the statistic we are calling % NEC S). At different dendrogram cut levels, the best percentage is bolded; in every case but one it is modified Walktrap plus DSD, at varying thresholds (5.5, 6, and 6.5).

Method	Enriched Clusters	# NEC	**% NEC**	# NEC S	**% NEC S**
PPI	201/225 (89.33%)	5650.0	**92.65%**	2409.0	39.50%
4.5	185/244 (75.82%)	2190.0	35.93%	1322.0	21.69%
5.0	176/252 (69.84%)	5003.0	82.07%	2100.0	34.45%
5.5	175/251 (69.72%)	4651.0	76.30%	2223.0	36.47%
6.0	168/224 (75.00%)	4997.0	81.97%	2473.0	**40.57%**

Table 2.7: The performance of Spectral versus Spectral plus DSD at different edge removal thresholds when the input parameter K in all cases is set to 300, but then we discard clusters of size < 3. NEC= "Nodes in Enriched Clusters". We calculate $\%NEC$ in two settings: $\%NEC$ is enrichment in the GO hierarchy with terms above the fifth level filtered out, and $\%NEC\ S$ uses the same filtered GO hierarchy, but then only gives a node credit if there is a match between one of the node's labels and one of the terms for which there is GO enrichment for the cluster. In this case, the Spectral algorithm run directly on the PPI network results in a higher $\%NEC$ statistic than any of the DSD-preprocessed results. However, without cluster size restrictions $\%NEC\ S$ is the most meaningful statistic, and it is best when Spectral is run with DSD at a distance threshold of 6.0.

Method	Enriched Clusters	# NEC	**% NEC**	# NEC S	**% NEC S**
PPI	234/324 (72.22%)	3082.0	50.54%	2158.0	35.39%
4.5	194/266 (72.93%)	1647.0	27.02%	1330.0	21.82%
5.0	199/309 (64.40%)	3589.0	58.87%	2203.0	36.14%
5.5	189/291 (64.95%)	3765.0	61.76%	2228.0	36.55%
6.0	177/249 (71.08%)	4670.0	**76.61%**	2490.0	**40.85%**

Table 2.8: The performance of Spectral versus Spectral plus DSD at different edge removal thresholds when the input parameter K in all cases is set to 300, but then we discard clusters of size < 3 and split clusters of size > 100. NEC= "Nodes in Enriched Clusters". We calculate $\%NEC$ in two settings: $\%NEC$ is enrichment in the GO hierarchy with terms above the fifth level filtered out, and $\%NEC\ S$ uses the same filtered GO hierarchy, but then only gives a node credit if there is a match between one of the node's labels and one of the terms for which there is GO enrichment for the cluster. For every threshold we tested ≥ 5, the percentage of nodes in enriched clusters is better than Spectral run alone for both measures.

Method	Enriched Clusters	# NEC	**% NEC**	# NEC S	**% NEC S**
PPI	252/510 (49.41%)	4540.0	29.96%	2301.0	15.18%
6.0	268/543 (49.36%)	6632.0	43.84%	2453.0	16.21%
6.5	286/543 (52.67%)	7085.0	46.83%	2918.0	19.29%
7.0	269/537 (50.09%)	7485.0	49.47%	3092.0	20.44%
7.5	272/552 (49.28%)	7243.0	47.87%	3073.0	20.31%
8.0	268/491 (54.58%)	7689.0	**50.82%**	3208.0	**21.20%**

Table 2.9: The performance of Spectral versus Spectral plus DSD at different edge removal thresholds when the input parameter K in all cases is set to 300, but then we discard clusters of size < 3 and split clusters of size > 100 on the Human network. We calculate $\%NEC$ in two settings: $\%NEC$ is enrichment in the GO hierarchy with terms above the fifth level filtered out, and $\%NEC\ S$ uses the same filtered GO hierarchy, but then only gives a node credit if there is a match between one of the node's labels and one of the terms for which there is GO enrichment for the cluster. By both of the NEC statistics, at every DSD threshold, detangling with DSD performs better.

Chapter 3

Overlapping Clusters

Methods

Extending Louvain

The Louvain algorithm is nondeterministic, and when run with different starting nodes or different node order, produces different clusters. We can therefore use several runs of the Louvain algorithm, where each run produces a set of non-overlapping clusters, to produce a set of overlapping clusters. We do this in three ways. Each method uses the same set of ten trials from chapter one, produced by modified Louvain where the clusters are > 3 and < 100 nodes, and the clusters are from the level with the best modularity.

Concatenating Five Sets

The first method of producing overlapping clusters from Louvain results is to combine five of the ten sets of clusters into one set. Five sets are chosen at randomly from the ten sets, and these five are concatenated into one cluster set. In the final cluster set, each node is automatically in five clusters, although when clusters that are less than three nodes are removed, some nodes may be in less than five clusters. The clusters are not checked for similarity, so there may be repeated clusters in the final set. If the same cluster appears twice in a set, we can postulate that this cluster is strongly connected, as multiple runs of Louvain found it. We then measure the quality of the final cluster set. This whole process is repeated over ten trials, and the five cluster sets are chosen at random each time, so the enrichment results are the mean of the ten trials.

Top Conductance Clusters

The second method is based on the measurement of conductance of each cluster as defined by Kannan et al. in [KVV04]. Consider the graph $G = (V, E)$. The conductance of a cut $(S, \overline{S})$ is defined as:

$$\varphi(S) = \frac{\sum_{i \in S, j \in \overline{S}} a_{ij}}{min(a(S), a(\overline{S}))}$$

where a is the adjacency matrix for G, so

$$a(S) = \sum_{i \in S} \sum_{j \in V} a_{ij}$$

is the total weight of the edges incident with S.

If the conductance of a cluster is low, it is considered a "strong" cluster, and if the conductance is high, it is a "weak" cluster.

To produce overlapping clusters from a ten different cluster sets, where each node is in ten clusters, we first calculate the conductance of each cluster for each node. The conductance is calculated as if the cluster in question is the cut. Then, the five clusters for each node that have the lowest conductance are chosen as the clusters that node is in. In the final cluster set, before clusters under three nodes are removed, each node is in five clusters. These five clusters may not be identical to the clusters the node was originally in, because other nodes may have been removed from the cluster. In the final cluster set, repeat clusters are removed so every cluster is unique, and clusters under three nodes are removed.

Top Modularity Clusters

The third method is similar to the second method, but uses the measure of modularity [GN02]. Modularity is measured in the context of a partition of the graph. The modularity is the fraction of edges that are in the partitions minus the expected fraction if the edges were randomly distributed. Consider the weighted graph $G = (V, E)$, with adjacency matrix A. The modularity is defined as:

$$Q = \frac{1}{2m} * \sum_{ij} (A_{ij} - \frac{k_i * k_j}{2m}) \delta c_i, c_j$$

where m is the total edge weight of the graph and k_i is the total weight of edges adjacent to node i [BGLL08].

For our purposes, to find the top modularity clusters, we consider the modularity of the graph with respect to a partition with two groups: the cluster in question and the rest of the graph. In the same manner as with the conductance method, we calculate the modularity with respect to each of the ten clusters a single node is in, then choose the five clusters with the top modularity score as that node's final clusters. In the final set of clusters, repeat clusters and clusters under three nodes are removed.

Results

For each algorithm, we consider the results from running that algorithm on the PPI network, and on the network after it has been preprocessed using DSD. We consider the extended Louvain algorithm with a bound on cluster size for the first step (obtaining non-overlapping clusters), where the bound is 100 nodes. We then produce overlapping clusters with the three methods described above. The results appear in Tables 3.1, 3.2, and 3.3. For all three methods, the algorithm performed best when run directly on the PPI network. Of the three methods, concatenation produces the best clusters by a small margin.

Figures 3.1, 3.2, and 3.3 show the number of clusters that are enriched versus the number that are not enriched at each cluster size range, comparing the number when the method is run directly on the PPI network versus when DSD is applied with a cutoff of 6.

When compared to the results from Louvain in Chapter 2 in Table 2.2, the concatenation method produces similarly meaningful clusters. For example, when run directly on the PPI network, Louvain produces clusters with 80.10% NEC (the median of 10 runs) and the concatenation method produces clusters with 80.18% NEC (the mean of 10 runs), and when run with DSD at a distance threshold of 6.0, Louvain produces clusters with 72.40% NEC (the median of 10 runs) and the concatenation method produces clusters with 73.01% NEC. Also, compared to the non-overlapping Louvain results, the best conductance and best modularity methods produce worse clusters according to both % NEC and % NEC S.

Discussion

In this chapter, we have explored three methods of producing overlapping clusters using the Louvain algorithm, the first is concatenating five sets of non-overlapping clusters into one set of overlapping clusters, the second is, over ten sets of non-overlapping clusters, putting each node in the five clusters that have the highest conductance score, and the third is the same as the second using the modularity score instead of conductance. We ran these methods on the yeast network with a bound on cluster size of > 3 and < 100 nodes. These methods produce the best clusters when run directly on the PPI network as opposed to the network that has been processed using DSD. This is congruent with our findings in Chapter 2 that the Louvain algorithm with a bound on cluster size produces better clusters when run directly on the PPI network. We have also compared overlapping clusters produced by these methods to the non-overlapping clusters produced by Louvain algorithm with bounded cluster size. The concatenation method produces clusters of a similar quality to the non-overlapping clusters from Louvain by our measure, and the best conductance and best modularity method on produce worse clusters by our measure than the non-overlapping method on the same parameters. However, because many proteins have multiple functions or participate in multiple processes, the overlapping clusters produced through the methods explored here are perhaps more accurate representations of the biologicial reality.

Louvain is not stable in it's production of clusters, so the variance in the results may be more reflective of Louvain's variance than the variance in the methods explored here. Further exploration of Louvain can be done to quantify the variance in the clusters, and determine the significance of the variance.

Tables

Method	Enriched Clusters	# NEC	% NEC	# NEC S	% NEC S
PPI	818.7/1016.5 (80.54%)	4887.92 SD = 46.10	80.18%	2690.14 SD = 32.90	44.13%
4.0	525.6/751.2 (69.97%)	1238.50 SD = 15.01	20.32%	1007.28 SD = 14.20	16.52%
4.5	814.8/1178.3 (69.15%)	2278.92 SD = 37.38	16.20%	1723.12 SD = 14.73	28.27%
5.0	831.2/1278.6 (65.01%)	3913.80 SD = 25.16	64.20%	2435.40 SD = 37.13	39.95%
5.5	582.1/890.9 (65.34%)	4328.16 SD = 45.81	71.00%	2353.08 SD = 21.73	38.60%
6.0	468.3/716.8 (65.33%)	4450.58 SD = 41.74	73.01%	2690.14 SD = 32.90	44.13%

Table 3.1: The performance of the concatenation algorithm vs the concatenation algorithm plus DSD at different edge removal thresholds. The reported values are the mean over 10 runs, and the standard deviation (SD) from the mean. NEC= "Nodes in Enriched Clusters". We calculate $\%NEC$ in two settings: $\%NEC$ is enrichment in the GO hierarchy with terms above the fifth level filtered out, and $\%NEC\ S$ uses the same filtered GO hierarchy, but then only gives a node credit if there is a match between one of the node's labels and one of the terms for which there is GO enrichment for the cluster. Each node is worth $1/\#clusters$. The algorithm performed best in all three measures on the PPI network without preprocessing.

Method	Enriched Clusters	# NEC	% NEC	# NEC S	% NEC S
PPI	870/1341 (64.88%)	4416.73	72.45%	2498.78	40.99%
4.0	104/150 (69.33%)	1207.00	19.80%	1250.00	20.51%
4.5	236/350 (67.43%)	2238.45	36.72%	1691.02	27.74%
5.0	576/929 (62.00%)	3702.47	60.74%	2364.28	38.78%
5.5	553/924 (59.85%)	4071.98	66.80%	2300.60	37.74%
6.0	469/752 (62.37%)	4290.50	70.38%	2197.27	36.04%

Table 3.2: The performance of the top conductance algorithm vs the top conductance algorithm plus DSD at different edge removal thresholds. NEC= "Nodes in Enriched Clusters". We calculate $\%NEC$ in two settings: $\%NEC$ is enrichment in the GO hierarchy with terms above the fifth level filtered out, and $\%NEC\ S$ uses the same filtered GO hierarchy, but then only gives a node credit if there is a match between one of the node's labels and one of the terms for which there is GO enrichment for the cluster. Each node is worth $1/\#clusters$. The algorithm performed best in all three measures on the PPI network without preprocessing.

Method	Enriched CLusters	# NEC	% NEC	# NEC S	% NEC S
PPI	689/1062 (64.88%)	4424.22	72.58%	2395.68	39.30%
4.0	103/150 (68.57%)	1198.00	19.65%	963.00	16.50%
4.5	232/343 (67.64%)	2207.07	36.21%	1647.15	27.02%
5.0	512/836 (61.24%)	3733.50	61.25%	2258.92	37.06%
5.5	493/827 (59.61%)	4037.50	66.23%	2184.97	35.84%
6.0	444/725 (61.24%)	4296.67	70.48%	2154.80	39.30%

Table 3.3: The performance of the top modularity algorithm versus the top modularity algorithm plus DSD at different edge removal thresholds. NEC= "Nodes in Enriched Clusters". We calculate $\%NEC$ in two settings: $\%NEC$ is enrichment in the GO hierarchy with terms above the fifth level filtered out, and $\%NEC\ S$ uses the same filtered GO hierarchy, but then only gives a node credit if there is a match between one of the node's labels and one of the terms for which there is GO enrichment for the cluster. Each node is worth $1/\#clusters$. The algorithm performs best in all three measures on the PPI network without preprocessing.

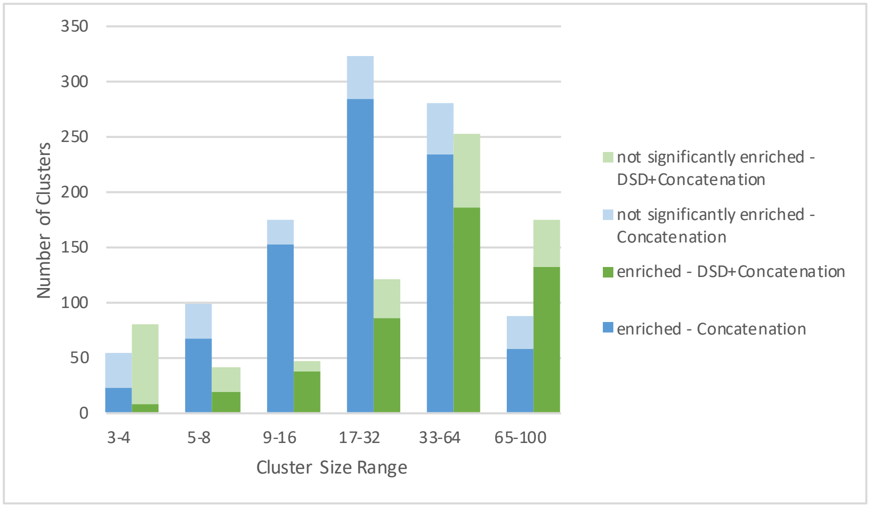

Figure 3.1: This figure compares cluster sizes running the concatenation method (with cluster sizes restricted to 3-100) directly on the PPI network with the concatenation method running on the DSD-detangled network (again with cluster sizes restricted to 3-100), with an edge removal threshold of 6. The percentage of nodes in enriched clusters is 80.54% for the method directly and 65.33% for DSD+concatenation.

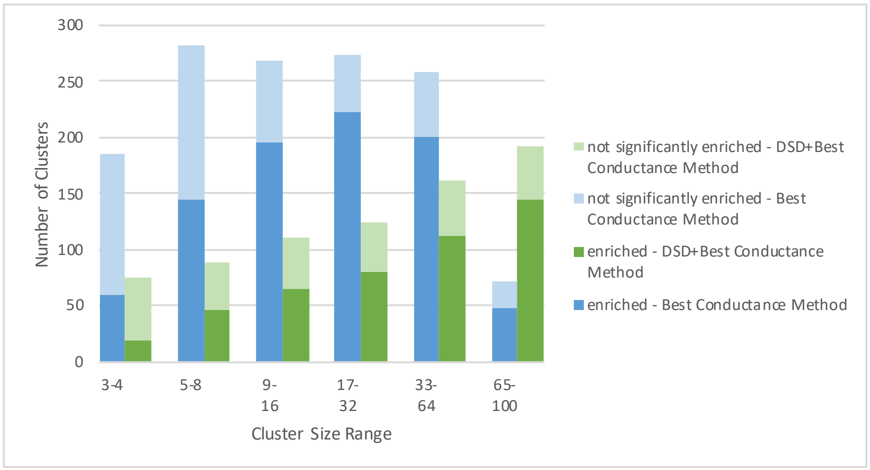

Figure 3.2: This figure compares cluster sizes running the best conductance method (with cluster sizes restricted to 3-100) directly on the PPI network with the top conductance method running on the DSD-detangled network (again with cluster sizes restricted to 3-100), with an edge removal threshold of 6. The percentage of nodes in enriched clusters is 64.88% for the method directly and 62.37% for DSD+best conductance.

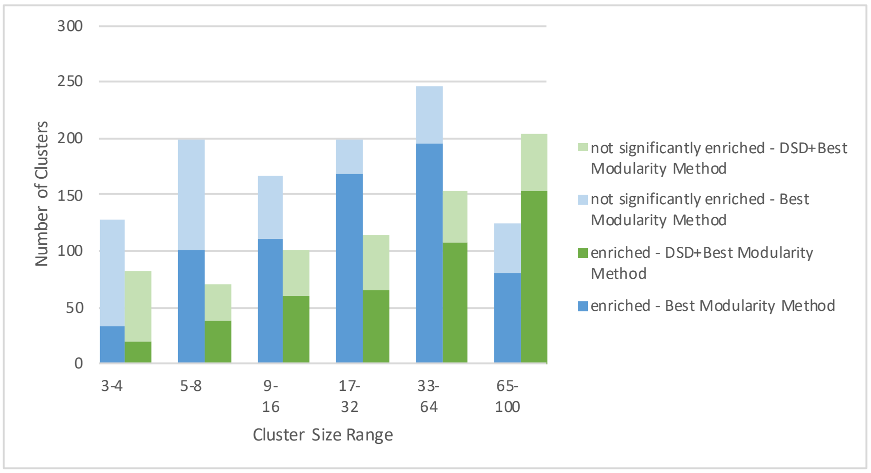

Figure 3.3: This figure compares cluster sizes running the best modularity method (with cluster sizes restricted to 3-100) directly on the PPI network with the top modularity method running on the DSD-detangled network (again with cluster sizes restricted to 3-100), with an edge removal threshold of 6. The percentage of nodes in enriched clusters is 64.88% for the method directly and 61.24% for DSD+best modularity.

Chapter 4

Conclusion

In this thesis, we have explored the performance of several clustering algorithms on a PPI network when the network is preprocessed using DSD. Some of the methods that produce non-overlapping clusters perform better when run with DSD. We also explored methods of producing overlapping clusters using the popular clustering algorithm Louvain, and showed that these methods perform best when run directly on the PPI network.

The next steps are to measure whether a similar DSD pre-processing step improves algorithms for community detection in other biological networks and verify that there are similar results on networks arising from additional species, and also seek to investigate whether the results remain true on networks built using different types of gene-gene or protein-protein association data. We will continue to study the best way to measure cluster quality when faced with a different number of clusters of different sizes.

Our work with overlapping clusters is preliminary, and can also be extended to other clustering algorithms. The method of concatenating many sets of non-overlapping clusters can be applied to other nondeterministic algorithms, or to cluster sets from multiple algorithms. Also, there are many algorithms developed to produce overlapping clusters that can be explored, such as ClusterOne [NYP12], OSLOM [LRRF11], and BigClam [YL13].

Bibliography

[AMM05] V. Arnau, S. Mars, and I. Marin. Iterative cluster analysis of protein interaction data. *Bioinformatics*, 31:364–378, 2005.

[BBC+09] Gabriel F Berriz, John E Beaver, Can Cenik, Murat Tasan, and Frederick P Roth. Next generation software for functional trend analysis. *Bioinformatics*, 25(22):3043–3044, 2009.

[BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[BRCG12] E. Becker, B. Robisson, C.E. Chapple, and A. Gunoche. Multifunctionaly proteins revealed by overlappng clustering in protein interaction network. *Bioinformatics*, 28(1):84–90, 2012.

[CN06] Gabor Csardi and Tamas Nepusz. The Igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9, 2006.

[Con16] The DREAM Consortium. The dream disease module challenge. Manuscript in preparation. See https://www.synapse.org/modulechallege., 2016.

[CPF+14] Mengfei Cao, C. M. Pietras, X. Feng, K. J. Doroschak, T. Schaffner, J. Park, H. Zhang, L. J. Cowen, and B. Hescott. New directions for diffusion-based prediction of protein function: incorporating pathways with confidence. *Bioinformatics*, 30:i219–i227, 2014.

[CZP$^+$13] Mengfei Cao, Hao Zhang, Jisoo Park, Noah M. Daniels, Mark E. Crovella, Lenore J. Cowen, and Benjamin Hescott. Going the distance for protein function prediction. *PLOS One*, 8:e76339, 2013.

[For10] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[GN02] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences USA*, 99(12):7821–7826, 2002.

[HBG$^+$14] Steve Harenberg, Gonzalo Bello, L Gjeltema, Stephen Ranshous, Jitendra Harlalka, Ramona Seay, Kanchana Padmanabhan, and Nagiza Samatova. Community detection in large-scale networks: a survey and empirical evaluation. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):426–439, 2014.

[KVV04] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

[LLM10] Jure Leskovec, Kevin J Lang, and Michael Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World Wide Web*, pages 631–640. ACM, 2010.

[LRRF11] Andrea Lancichinetti, Filippo Radicchi, José Ramasco, and Santo Forunato. Finding statistically significant communities in networks. *PLoS ONE*, 6(4):e18961, 2011.

[NJW$^+$01] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. In *NIPS*, volume 14, pages 849–856, 2001.

[NYP12] Tamás Nepusz, Haiyuan Yu, and Alberto Paccanaro. Detecting overlapping protein complexes in protein-protein interaction networks. *Nature Methods*, 9:471–472, 2012.

[PL06] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *J. Graph Algorithms Appl.*, 10(2):191–218, 2006.

[PVG$^+$11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[SS09] Jimin Song and Mona Singh. How and when should interactome-derived clusters be used to predict functional modules and protein function? *Bioinformatics*, 25(23):3143–3150, 2009.

[Szk15] et al. Szklarczyk, Damian. String v10: protein–protein interaction networks, integrated over the tree of life. *Nucleic Acids Research*, 43(D1):D447–D452, 2015.

[VM03] Deepak Verma and Marina Meila. A comparison of spectral clustering algorithms. *University of Washington Tech Rep UWCSE030501*, 1:1–18, 2003.

[YL13] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pages 587–596. ACM, 2013.