2121
21
21
Jake Crawford[1 2] Henri Schmidt[2] University, Medford, MA Junyuan Lin[3 4] Benjamin Hescott[5]
21
Donna Slonim[2] Xiaozhe Hu[3] Lenore J. Cowen[2 6]

A Double Spectral Method for Uncovering Disease Modules in Heterogeneous Biological Networks

Abstract

We present details of our top-scoring method for the DREAM 2016 Disease Module identification challenge. The method uses spectral techniques twice, first to re-embed the networks using the DSD distance metric, so that proximity in the embedded space corresponds to meaningful neighborhood similarity in the original networks, and second, when standard spectral clustering is called on the embedded nodes to produce the clusters themselves. We explore whether combining an additional method of finding clusters based on bipartite structure (which we also explored and incorporated into our challenge contest submission) improves performance, and find that on balance, on the collection of networks and data types used to score the challenge, it does not. We also explore, for the single challenge network that was provided with directions on the edges, whether a version of our double-spectral method that can respect edge directions performs better than the same method when edge directions are discarded and all edges are treated as undirected. Our double-spectral method is entirely deterministic, accepts edge weights without any need to modify or pre-process them and only has a single parameter to tune, namely m, the number of target clusters initially passed to the spectral clustering algorithm. We explore the dependence on m for the different DREAM challenge networks, and find that a larger m is more meaningful in the classical PPI networks, while in the DREAM network constructed from co-expression data, a smaller number of larger clusters displayed more significant functional enrichment. Similar to other groups, we find that modular structure captures more meaningful information content in DREAM networks 121 -4 than in DREAM networks 5 and 6. We release all the modules we find on the DREAM networks using our methods, spotlighting a few that seem particularly biologically interesting. We release a new code package that efficiently allows both the exact and approximate computation of our algorithms.

1 Introduction

When performing inference on biological networks, there is an important place for both supervised and unsupervised methods. Supervised methods are able to leverage existing biological knowledge, whereas unsupervised methods rely instead on the intrinsic connectivity structure of the network(s) alone, and therefore can often discover things further from known genes and pathways. A core unsupervised problem studied on many networks is the *community detection* problem, where the network is to be split into non-overlapping communities, or *modules,* that should correspond to sets of genes that cooperate in the network.

When community detection algorithms are benchmarked in other domains, such as social networks, the performance of such algorithms are usually tested in two extreme cases: in the first case, it is assumed that the ground truth "true" communities are known (for example, they are *planted cliques* [8]) and performance is measured in terms of how well these true communities are recovered. In the second

Subchallenge 1, such a score was computed for each network separately, and then summed to produce the total score for that team's entry.

This paper describes our winning "Double Spectral" method for Subchallenge 1 of the DREAM Challenge in depth. We explain exactly how our winning entry was constructed in Section 2, and discuss some of the interesting modules we find with literature support in Section 3. Based on a post-facto analysis, we analyze the robustness of the method to different parameter settings, and make recommendations on how those parameters should be set. A main contribution of this paper is also an approximate "Double Spectral" method that greatly speeds up the original method without any major sacrifice in performance.

1.1 Method overview

The heart of our Double Spectral method is the "Diffusion State Distance" metric defined in Cao et al. in 2013 [5] and generalized to deal with confidence weights on the edges (cDSD) in 2014 [4].

Our general strategy was as follows: 1) Compute the cDSD matrix 2) Use this as input to a generic off-the shelf well-studied clustering method (in the case of our submission, spectral clustering [20]).

Since Spectral clustering can return clusters that are both smaller and bigger than the target size of 3-100 nodes, we ignored any cluster that was of less than 3 nodes, and recursively split clusters with more than 100 nodes. For half the networks, we also tried to find modules with dense bipartite subgraphs using the Genecentric program [12]. This did not end up impacting the results greatly, and in fact few such modules were found in these networks. We discuss this further below.

The structure of the remainder of the paper is as follows. In Section 1.2 we review and offer intuition about the cDSD metric. In Section 2.2 we discuss different ways to speed up computation of an approximate cDSD matrix. There were very few parameters that are needed to be set to run our method: edge weights were used unchanged, and the main parameter which needs to be set is the number, m, of clusters initially passed as input to spectral clustering. In Section 3.1, we explore after the fact, how the performance of our method changes based on different settings of m, the number of clusters passed to spectral clustering. We also look in depth at the contribution adding Genecentric clusters makes to performance. In Section 3.2 we highlight some of the biologically interesting modules found by our winning entry.

For networks 2, 3, and 5, we also tried to find clusters with the Genecentric program [12]. Genecentric was designed for a very different use case, in networks where the edges represent positive and negative *genetic* interactions (and the edge weights are both positive and negative.) When Genecentric is run with all edge weights set to -1, it becomes equivalent to the algorithm of Brady et al [3] and will find modules that contain dense bipartite subgraphs. Dense bipartite subgraphs were found to be signatures of possible instances of the "Between Pathway Model" [15] type of functional modules in networks whose edges indicated *negative genetic interactions* [3]. Since the *type* of network data used to construct the anonymous networks was also blinded for contest participants, we looked at searching for this type of structure in constructing our modules as well: the method we used to search for dense bipartite structure is essentially the algorithm in [3]: the implementation we used was the Genecentric software package [12] with the network passed to the algorithm with all edge weights set to -1. Based on performance in the training rounds, we did this for networks 2, 3, and 5 (and not for networks 1, 4, and 6). We explain further how we merged the Genecentric clusters with the spectral clusters below.

We note that Genecentric could alternatively be run with all edge weights set to 1. In this case, it would return pairs of clusters of high modularity, making its output more similar to that of other

methods.

1.2 Some Intuition about DSD

PPI networks are known to be "small world" networks in the sense that they are small-diameter, and most nodes are close to all other nodes. Thus any method that infers similarity based on proximity will find that a large fraction of the network is proximate to any typical node. In fact, this issue has already been termed the "ties in proximity" problem in the computational biology literature [1].

In Cao et al. [5, 4], we argued that the typical shortest-path measure of proximity missed a lot of informative structure that was encoded in the network: not all short paths gave equal indications of similarity; paths through low-degree nodes were stronger indications of functional similarity than paths that went through high-degree nodes, or hubs. For intuition, we could take an analogy with social networks. If two people who do not know each other, have several Facebook mutual "friends" each with a low to moderate number of total number of friends themselves, the claim is that this is a stronger vote that they might have something in common than if they are both Facebook "friends" with a "famous" node with millions of friends such as Oprah Winfrey. Because everyone is Facebook friends with Oprah, a length-two path through Oprah is a much weaker indication of similarity than a length-two path through a low-degree node would be. Going back to biological networks, we find a similar phenomenon, where hub nodes tend to be involved in the general machinery of the cell, interacting with many proteins that are not functionally related. Proteins that are both connected to such a high-degree hub are less likely to have similar function, than proteins that both interact with a protein of much lower degree.

In order to come up with a more fine-grained measure of similarity that downweights hubs, [5] defined a new spectral graph metric called Diffusion State Distance, or DSD. Consider the undirected graph $G(V, E)$ on the vertex set $V = \{v_1, v_2, v_3, ..., v_n\}$ and $|V| = n$. Recall that $He^{\{k\}}(A, B)$ is defined as the expected number of times that a simple symmetric random walk starting at node A and proceeding for k steps, will visit node B (we always include the origin of the walk as a visit in step 0). In what follows, assume k is fixed, and when there is no ambiguity in the value of k, we will denote $He^{\{k\}}(A, B)$ by $He(A, B)$. Notice that $He(A, B)$ begins to get at the notion of the relative importance of paths through low-degree rather than high-degree nodes; if A and B are connected through a low-degree neighbor, the random walk starting at A is likely to reach B; if however, a neighbor that connects them is of very high degree, the walk through that neighbor is likely to diffuse away and reaches B with much lower probability. However, we are not done, because $He(A, B)$ is not a metric; in particular, it is not even symmetric (for example, if A is at the center of a star graph with many petals, and B is at one of the petals, a walk started at B will always reach A, whereas a walk started at A is unlikely to reach B).

Thus we further define a $n - dimensional$ vector $He(v_i), \forall v_i \in V$, where

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

This vector can be thought of as the global view of the $He(A, B)$ measure from each vertex to all the other vertices of the network.

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

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

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

[5] showed for any fixed k, that DSD is a 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, 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. DSD generalizes easily to weighted edges; the random walk defined in the He measure is instead biased according to the edge weights[4]. We remark also that convergence occurs very quickly in these small-world interaction networks, and that if k is set to 7, instead of ∞, the resulting distances are very empirically close to the converged DSD.

We note that in the main challenge paper [6], DSD is defined as an example of a "kernel method", since we are re-embedding nodes in space according to their DSD distance. However, at its heart, DSD is also a special type of random-walk method that internally uses network propagation [10] to define its embedding kernel.

1.3 Some Intuition about Genecentric

A *maximal bipartite subgraph* of $G(L)$ is a partition of the vertex set of $G(L)$ into two disjoint subsets of vertices A and B as follows. Let $|C|$ denote the number of edges with one endpoint in A and one endpoint in B. Now, for any vertex $v \in A$, define A' to be $A - v$ and B' to be $B \cup v$. Then (A, B) is maximal in $G(L)$ implies that the number of edges of $G(L)$ that have one endpoint in A' and one endpoint in B' is at most $|C|$. And similarly for $v \in B$. In other words, moving a single vertex from A to B or vice versa cannot increase the number of edges that go across from A to B.

The randomized algorithm of Brady et al [3] searches for *stable* bipartite subgraphs. These are bipartite subgraphs that appear in a large proportion of the *maximal bipartite subgraphs* of G. They tend to be small, dense bipartite subgraphs. When the Genecentric software package [12] is passed a network where edges are assigned constant negative edge weight, it implements precisely the algorithm of Brady et al [3].

Dense bipartite substructure is likely to have meaning in *genetic* interaction networks, because it is likely to represent redundant functional pathways [15, 3]. When the sources of the anonymized networks for the DREAM contest challenge were revealed, it turned out that none of the test networks were derived from genetic interaction networks, and indeed, as we show below, Genecentric does not find a large number of clusters in any of the DREAM networks because they do not have many dense bipartite subgraphs. However, Genecentric will also find some instances where a set A of many independent proteins interact with a small clique B (because even though the edges between vertices of B make this graph non-bipartite, this is also a case where the number of edges between A and B strictly dominates the number of edges in A or B alone). Our method finds some instances of meaningful modules using Genecentric that we discuss below; in absolute numbers it did not end up affecting our performance very much. Therefore, we don't recommend adding Genecentric when analyzing network data of the types considered by the DREAM challenge, but include it in this paper to produce a historically accurate and complete description of the methods we ran for the DREAM contest challenge.

2 Methods

2.1 Computation of the DSD matrix: Overview

The DSD matrix can be computed for any weighted network using the "cDSD" method from software available at: http://dsd.cs.tufts.edu/capdsd.

Converged cDSD can be computed in time proportional to inverting a matrix; it is completely feasible to compute it directly on networks of the size of this DREAM challenge, even on ordinary desktop computers. Furthermore, it only has to be done once, for each of the six networks. However, because we were running on ordinary desktop computers, we decided to instead produce very close approximations for the cDSD matrix instead of computing it exactly in one of two ways. 1) Either running cDSD with a 7-step random walk only (i.e. setting $k = 7$ instead of $k = \infty$; note that when k is set greater than the diameter of the network, cDSD converges very quickly, so $k = 7$ presents a good approximation on the DREAM challenge networks) or 2) a new method to apply algebraic multigrid (AMG) methods to speed up the computation of the discrete Green's function. We describe 2) in more detail in the next section.

For Subchallenge 1, we used the AMG methods (described next) to produce the approximate cDSD distance matrices for networks 1, 2, 4, 5, and 6. For network 3 (the directed signaling network), we wanted to keep the edge directions for subchallenge 1. However, the edge directions could have caused the network to be (weakly) disconnected, which would result in some nodes having infinite cDSD distance. So, we kept the directions of the original edges and added low-weight backedges, having $\frac{1}{100}$ of the minimum edge weight in the network, to prevent the network from being disconnected. We then ran the original version of cDSD, with a random walk of length 7, on the resulting largest connected component.

On subchallenge 2, we could have used the faster AMG methods on our combined networks (see below) as well, but we ended up using the original version of cDSD with a random walk of length 7 instead, because it was easier than coordinating between different members of the team.

2.2 AMG methods

Consider a connected graph $G = (V, E)$. The *Diffusion State Distance* (DSD) between two nodes u and v is defined as:

$$\mathrm{DSD}(u, v) = ||(\boldsymbol{b}_u^T - \boldsymbol{b}_v^T)(\boldsymbol{I} - \boldsymbol{P} + \boldsymbol{W})^{-1}||_1 \tag{1}$$

where $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{P}$ is the transition matrix, and $\boldsymbol{W}$ is a matrix whose each row is equal to the transpose of the steady state distribution $\boldsymbol{\pi}$, i.e., $\boldsymbol{W} = \boldsymbol{e}\boldsymbol{\pi}^T$, where $\boldsymbol{e}$ is the constant vector of all ones. Since we need to compute the distances between all pairs of nodes, the most time expensive part is computing the matrix inverse in (1), which is the discrete Green's function of the graph G, i.e.,

$$\mathbb{G} = (\boldsymbol{I} - \boldsymbol{P} + \boldsymbol{W})^{-1}. \tag{2}$$

Our efficient algorithm for computing the discrete Green's function is based on the nearly-optimal solver for the graph Laplacian [17]. To use this, we need to rewrite the discrete Green's function a little bit. Let $\boldsymbol{A}$ be the adjacency matrix and $\boldsymbol{D}$ be the diagonal degree matrix. The graph Laplacian is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ and the normalized graph Laplacian can be obtained as $\boldsymbol{N} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$. It is well-known that the eigenvector corresponding to the the zero eigenvalue of $\boldsymbol{N}$ is given by

Table 1: CPU time for computing the distance of one pair of nodes (Sub-challenge 1 Networks)

	Information		CPU time (second)		
	#Edges	#Nodes	Old method (C)	New method (Matlab)	Speedup
Network 1	2,232,405	17,397	17.959	1.596	11.25
Network 2	397,309	12,420	7.558	0.277	33.30
Network 3	21,826	5,254	0.953	0.048	19.85
Network 4	1,000,000	12,588	7.713	0.524	14.20
Network 5	1,000,000	14,679	11.060	0.687	16.10
Network 6	4,223,606	10,405	5.233	3.068	1.71

$\boldsymbol{d} := \frac{1}{\sqrt{d_{total}}} \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{e}$. Using those definition and some algebraic calculations, we arrive at

$$\begin{aligned} \mathbb{G} = (\boldsymbol{I} - \boldsymbol{P} + \boldsymbol{W})^{-1} &= \left(\boldsymbol{D}^{\frac{1}{2}} (\boldsymbol{N} + \boldsymbol{d}\boldsymbol{d}^T) \boldsymbol{D}^{-\frac{1}{2}} \right)^{-1} \\ &= \boldsymbol{D}^{\frac{1}{2}} (\boldsymbol{N} + \boldsymbol{d}\boldsymbol{d}^T)^{-1} \boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{D}^{\frac{1}{2}} (\boldsymbol{N}^{\dagger} + \boldsymbol{d}\boldsymbol{d}^T) \boldsymbol{D}^{-\frac{1}{2}}, \end{aligned}$$

where $\boldsymbol{N}^{\dagger}$ is the pseudo-inverse of $\boldsymbol{N}$ which can be computed by fast linear solvers. In our case, we use algebraic multigrid (AMG) method to compute it. The following algorithm describes our method for computing the discrete Green's function.

Algorithm 2.1. *Compute* $\mathbb{G} = (\boldsymbol{I} - \boldsymbol{P}^T + \boldsymbol{\pi}\boldsymbol{e}^T)^{-1}$ *[1]* $i = 1 : N$ *Compute* $\boldsymbol{b} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{e}_i$ *where* $\boldsymbol{e}_i$ *is the* i*-th column of the identity matrix* $\boldsymbol{b}_2 \leftarrow \left(\boldsymbol{d}^T \boldsymbol{b}\right) \boldsymbol{d}$ *and* $\boldsymbol{b}_1 \leftarrow \boldsymbol{b} - \boldsymbol{b}_2$ *Compute* $\boldsymbol{x}_1$ *by solving* $\boldsymbol{N}\boldsymbol{x}_1 = \boldsymbol{b}_1$ *by the AMG method* $\mathbb{G}(:, i) \leftarrow \boldsymbol{D}^{\frac{1}{2}} (\boldsymbol{x}_1 + \boldsymbol{b}_2)$

Since we use the AMG method at Step 4 in Algorithm 2.1, which has computational complexity $\mathcal{O}(N \log N)$, $N = |V|$, the overall computational complexity of our algorithm is $\mathcal{O}(N^2 \log N)$. Note that, if the standard direct method is used, the overall computational cost is $\mathcal{O}(N^4)$ and the best computational cost we can expect for computing the inverse of a matrix is $\mathcal{O}(N^2)$. Therefore, our approach is nearly optimal.

Table 1 presents the amortized cost for computing the distance between one pair of nodes (to estimate the total time to compute the entire matrix, multiply by the square of the number of nodes in the network). This is actual time over the networks from subchallenge 1, where the CPU we used for the tests was an Intel Xeon CPU E5-2637 v3 @ 3.50GHz. One can see that our new method using the AMG algorithm can achieve about 16 times speed up on average comparing with old method using direct solvers. We can expect bigger speed up when we compute the whole distance matrix of the network.

2.3 Edge Weights

For subchallenge 1 and our final submission to subchallenge 2, we passed all of the edge weights unchanged to our cDSD algorithm. In each case, if the network was disconnected we ran cDSD on each connected component individually, and used the largest for clustering.

For subchallenge 2, we tried different ways of rescaling the edge weights on networks 3 and 6 in order to combine them with the other networks, since the edge weights in networks 3 and 6 appeared to be on a different scale. However, we ended up not including any information from networks 3 and 6 in our final submission to subchallenge 2.

2.4 Clustering method

Originally, we tried applying two different clustering methods to the resulting cDSD matrix: edge removal (similar to the Girvan-Newman algorithm [19], using DSD distance rather than edge betweenness to choose edges to remove), and spectral clustering [20].

In the training rounds, we observed that spectral clustering performed much better across all of the input networks. We used the spectral clustering package provided in scikit-learn [22], documented at http://scikit-learn.org/stable/modules/clustering.html .

Note that the spectral clustering algorithm we used operates on a similarity matrix (i.e. entries that are most alike have higher values in the matrix, and entries that are more disparate have lower values in the matrix). However, the cDSD matrix is a distance matrix; that is, similar entries have low cDSD values, and vice-versa. To convert the cDSD matrix to a similarity matrix, we applied the RBF kernel [23] to each entry in the cDSD matrix, which maps low distances to high similarity scores and vice-versa. We then clustered the similarity matrix.

Since the spectral clustering algorithm provided by scikit-learn uses k-means as the underlying clustering mechanism, it takes a parameter m specifying the number of cluster centers. In the training rounds, we varied m to change the number of clusters that were output.

We tried several different values of m for each network ($m = 100, 200, 500, 800, 1000$, and 1200 for each network). In the training rounds, fewer number of large clusters ($m = 100$) appeared to perform the best on networks 3, 4, and 6, and larger numbers of smaller clusters ($m = 1000$ and $m = 1200$) performed the best on network 1. Networks 2 and 5 performed the best when large clusters were combined with smaller Genecentric clusters, as described below.

Also note that spectral clustering will produce clusters of size less than 3, and clusters of size more than 100. Whenever we produced a cluster of size less than 3, we ignored those vertices and did not include them in a cluster. Whenever we produced a cluster of size more than 100, however, we recursively called spectral clustering again, with a cluster size of 2, and continued the recursion until all clusters were of size < 100. We compared this method with submissions in which large clusters were thrown out. With only one exception (network 3 with cluster size 200), splitting clusters resulted in performance that was unchanged or improved, usually improved.

2.5 Merging the Genecentric and Spectral Clustering Sets

On networks 2, 3, and 5, we also ran Genecentric with all the edge weights in the network set to -1. In the interest of running Genecentric reasonably quickly, we split the graph into large spectral clusters (using $m = 10$ and $m = 20$ for networks 2 and 5) based on cDSD as before, and ran Genecentric on the resulting spectral clusters. Network 3 was small enough to run Genecentric on directly, without splitting it into spectral clusters first.

We then ran Genecentric using a minimum partition size of 3 and a maximum partition size of 100, to generate clusters that fit within the size limits for the challenge, and edge squaring, as is recommended in the Genecentric documentation, to speed up the computation slightly. Otherwise, we used all of the default parameters.

Genecentric outputs clusters (which it calls modules) split into precisely two individual subclusters (which it calls pathways). We considered each individual pathway to be a separate Genecentric cluster, unless either of the partitions was smaller than 5 nodes. In that case, we combined the partitions into one large cluster. Note that Genecentric only puts a small percentage of the network into clusters;

most of the network remains unclustered by Genecentric.

We then had two sets of clusters: spectral clustering clusters and Genecentric clusters, and we had to merge them into a single set of non-overlapping clusters. We preferentially selected Genecentric clusters when we merged them. To merge Genecentric clusters with spectral clusters, we tried 2 approaches. The first was a greedy approach, simply removing all of the spectral clusters that overlapped at all with any Genecentric cluster. This meant that some (potentially informative) clustering information was thrown out, but in practice adding the Genecentric clusters and removing the corresponding spectral clusters seemed to help our scores overall on networks 2, 3, and 5.

In addition, we tried a different strategy for merging Genecentric clusters based on their overlap with spectral clusters. Consider the Genecentric clustering with all nodes that are not placed in a Genecentric cluster as a new, single additional large cluster. Now we return clusters of nodes that are placed in the same cluster by *both* the spectral and Genecentric clustering. In practice, this overlap method seemed to result in more informative final clusters on networks 2 and 5 during the training rounds, as long as the starting clusters are sufficiently large to allow for a high degree of overlap.

We tried this overlap strategy on network 3 as well, but it did not show appreciable improvement over the greedy merge strategy described above, at any cluster size we tried. In our final submission, we used the greedy merge approach for network 3, and the overlap approach for networks 2 and 5.

2.6 Subchallenge 2

For our contest submission, to Subchallenge 2, we instead created a *combined* network where we took the union of all the edges from a subcollection of networks 1-6, (with network 3 treated as undirected and networks 3 and 6 reweighted as follows: all edges on network 3 were given a constant weight of 0.95, and low-weight edges on network 6 were thrown out, with the remaining edge weights normalized between 0 and 1).

We then performed exactly the same spectral clustering method as we did for subchallenge 1 on the combined network. We did not add any Genecentric clusters. We computed cDSD on the combined network, and then performed spectral clustering, recursively splitting all clusters of size > 100 as before.

In the training rounds, we tried including different subsets of networks 1-6. We tried including all 6 networks, we tried including networks 1-4, we tried networks 1-3 and 1-2, and in the end, decided to include edges from network 1, 2, and 4 only. At a m-value of 500, this gave the best performance across all FDR values during the training rounds.

3 Results

3.1 Summary results and parameter settings

The performance of our winning entry for Subchallenge 1 of the DREAM challenge was extensively benchmarked in the main challenge paper [6]. In order to produce that collection of modules, we had only a few parameters to tune: most importantly, the number of clusters m that would be initially input to the Spectral clustering algorithm, but also, whether or not to run Genecentric, and if so, which of our two proposed methods to use to merge the Genecentric and Spectral clusters. We made some parameter choices based on performance in Leaderboard training rounds, but these were sufficiently limited that we could not make a full study of how these parameters affected our method's performance

network	$m = 100$	$m = 200$	$m = 500$	$m = 1000$
1	14	8	13	7
2	5	6	6	8
4	14	11	12	14
5	0	5	5	4
6	10	5	5	5

Table 2: The number of significant clusters scored by the Pascal tool when DSD/Spectral clustering is run with initial input m to the spectral clustering algorithm, on all DREAM networks except network 3 (where edges were directed and so the algorithm we employed on network 3 was modified to handle edge directions)

network	$m = 100$	$m = 200$	$m = 500$	$m = 1000$
1	8	7	7	6
2	2	4	6	5
4	9	10	6	7
5	0	5	5	3
6	5	7	4	6

Table 3: The number of significant clusters scored by the Pascal tool when DSD/Spectral clustering and Genecentric are both run with initial input m to the spectral clustering algorithm (on all DREAM networks except network 3), and the resulting clusters from the two methods are greedily merged. In 19/20 cases, performance stays the same or degrades from running just the DSD/Spectral clustering algorithm alone.

during the contest itself. We rectify this in our post-facto contest analysis. In particular, Table 1 gives the number of significant modules (as scored by the contest method) as the value m passed to the spectral clustering method is varied for five of the six DREAM networks (network 3, which was the only directed network, we consider separately below). We consider the following two collections of clusters for all networks A) using DSD/Spectral clustering only B) using DSD/Spectral clustering with Genecentric clusters, combined with the greedy method As can be seen from the table, including genecentric clusters did not have a significant impact on number of significant clusters found by Pascal: for a very few values of network and m it was occasionally better; for most settings of network and m, adding Genecentric slightly degraded performance.

This is not to say that some of the significant clusters that Genecentric finds aren't novel and interesting; just that since the contest rules required that the collection of clusters be non-overlapping, each enriched Genecentric cluster crowded out other spectral clusters instead, some of which tended also to be enriched and interesting. In many regions of the networks, Genecentric doesn't find any clusters. Figure 2 shows the smallest significant bipartite subgraph that Genecentric found in network 2: it is also the closest to truly bipartite, being only one edge away from a bipartite graph. This subgraph is found to be enriched for extreme height and BMI in males in the European Giant study [**?**], with the mouse phenotype enrichment terms displayed in Table 4.

When Genecentric is run by itself on network 2 (all edges being given the weight of -1). The rest of the bipartite subgraphs that Genecentric finds are large enough that the modules returned are one of the two pathways rather than both pathways together: Figure **??** is a typical example. Note that this module is not part of our final contest submission, since it spans multiple of the DSD spectral clusters, and thus we break it apart into multiple clusters using the overlap method in our contest

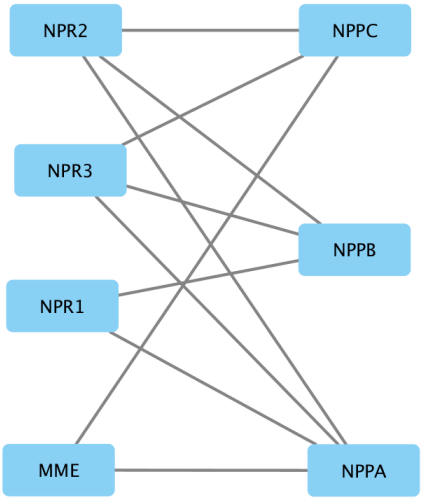


Figure 1: Team Tusk/Genecentric bipartite module 100, enriched for extreme height and BMI in Dream Network 2

NPR3, NPPC	abnormal vertebral epiphyseal plate morphology
NPR2, NPPC	disorganized long bone epiphysial plate
NPR2, NPPC	decreased long bone epiphyseal plate size
NPR2, NPR3	abnormal bone trabecula morphology
NPR2, PRRC, NPR3	abnormal long bone epiphyseal plate proliferative zone

Table 4: Associated phenotypes for genes in Network 2 Module 100 from the Mouse Phenotype Ontology

Gene	BMI:Men (EUR.GIANT 2015)	HT:Men (EUR.GIANT 2013)	HT(EUR.GIANT 2010)
NPR3	1.78742738E-6	1.12705572E-7	4.88764584E-12
NPPC	2.1292252E-5	2.91375657E-9	1.58095759E-13
NPR2	1.33682038E-1	9.37176087E-4	1.06468681E-5
NPPB	5.90374817E-2	4.5585421E-1	4.33398764E-1
NPPA	1.31599667E-1	3.67667874E-1	4.31629029E-1

Table 5: Pascal p-values give significance for 5 of the 7 genes in Network 2 Module 100, for GWAS studies involving height and BMI of Males in European population studies included in the DREAM GWAS collection.

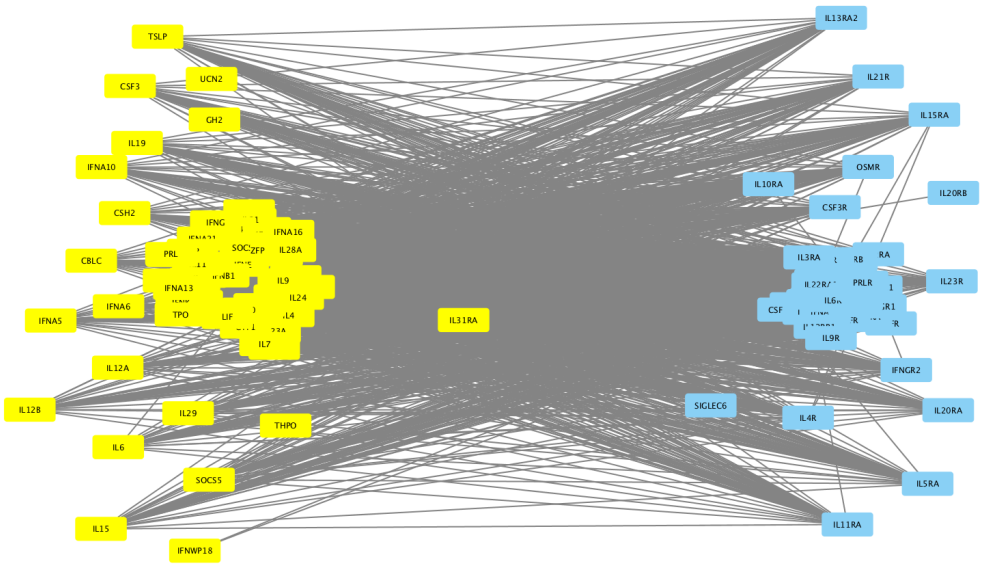


Figure 2: Large Genecentric module enriched for Neuroblastoma genes in Dream Network 2

submission.

We also remark that regardless of whether Genecentric is included or not, the clusters returned by our method are not necessarily even connected: sometimes genes of closest DSD distance have no direct connection in the original network.

We next wanted to explore whether keeping information about edge directions improved our performance in Network 3. We suspected that it did, because we found more significant modules in network 3 than any other team in our submitted contest entry. We note that all the new results below were run *without* including Genecentric.

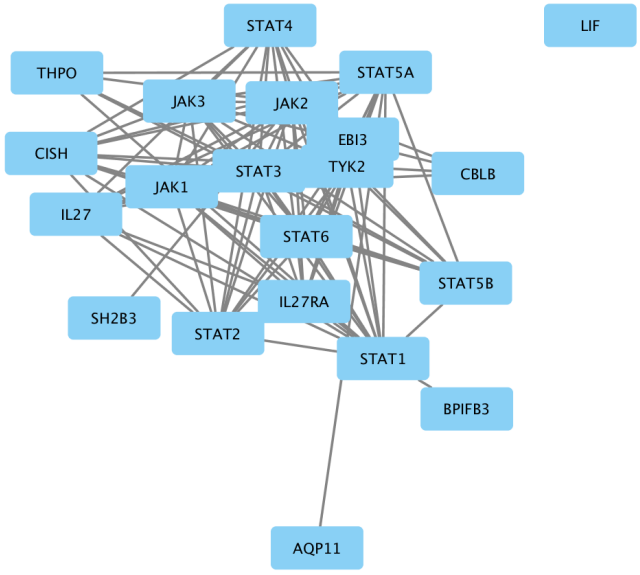
Recall that, as described above, our submitted entry for DREAM3 was constructed by running DSD (step 7) on the directed network, with weights unchanged; where if an edge only existed from u to v, but there was no edge from v to u, we added a back-edge from v to u of very low weight to preserve connectivity properties of the network. We compare this to taking DSD (step 7) on two different undirected networks: in both undirected versions, where there is a directed edge (u, v), but no directed edge (v, u), we replace directed edge (u, v) with an undirected edge (u, v) of the same weight. In the case where there exist directed edges (u, v) of weight e_1 and directed edge (v, u) of weight e_2, UndirectedA creates an undirected edge of weight that's the average of the weight of the two directed edges, and UndirectedM creates an undirected edge of weight that's the maximum of the weight of the two directed edges. Table 6 shows the summary results for different values of m (the initial input to Spectral Clustering); in most cases, it seems that preserving the directions can help, but the improvement is only minor.

3.2 Biological Results

After the contest was over, the de-anonymized node names were revealed, and it became possible to look back at the disease module "communities" that we produced and see if looking at the literature, we had found any biologically interesting modules. We spotlight a handful of the interesting modules we find; note that the entire collection of modules that we produced is available here (FILL IN). We

	UndirectedA	UndirectedM	Directed
$m = 100$	4	7	8
$m = 200$	7	7	7
$m = 500$	6	7	4
$m = 1000$	2	2	5

Table 6: Comparison of number of significant modules in network 3 (using DSD step 7), preserving the directed edges as compared to two weight variants for treating network 3 as an undirected network.



highlight below an interesting modules that contain individual genes enriched in a GWAS study of Irritable Bowel Disease and Crohn's disease in a European study population, and look at the literature support for Crohn's disease involvement for the other genes in the modules.

3.2.1 An interesting module related to Crohn's disease

Figure 3: Team Tusk module enriched for Crohn's disease genes in Dream Network 1

This module is enriched in a GWAS study of Irritable Bowel Disease and Crohn's disease in a European study population. The raw gene by gene p-values for the genes in this module appear in Table 7 for all genes with a non-negligible p-value.

The genes in this module that have intermediate GWAS support for Crohns disease in a European population are IL-27, STAT3, TYK2, and JAK2. STAT1, LIF and STAT5 have weak GWAS support (see Table 7). The rest of the genes in the module do not have any significant GWAS support as singleton genes in the study, but some have a known association with Crohns disease in the literature, as we now describe. For the GWAS supported genes, we find that, beyond the European population, IL-27 has three gene polymorphisms associated with Crohns disease in a Chinese Han population [30] whereas TYK2 and STAT3 have joint association for susceptibility to Crohns disease in the Japanese population [26], and a STAT6 gene polymorphism association has been found in Malaysian patients with Crohns disease [7]. All the STAT genes in the module below have some support for an association with Crohns disease in published literature, but STAT3 has perhaps been most studied. For example,

k height	
IL27	4.60007588e-10
JAK1	0.00061118583
STAT3	5.15645304e-09
STAT4	0.000364206324
IL27RA	NA
STAT2	NA
AQP11	NA
TYK2	1.6808388e-10
BPIFB3	NA
CISH	NA
STAT1	2.37516791e-05
LIF	1.77498137e-06
SH2B3	0.002654
CBLB	NA
THPO	NA
JAK2	4.33307967e-09
EBI3	NA
STAT5A	1.02941522e-06
STAT6	0.0001455
JAK3	NA
STAT5B	4.30089833e-06

Table 7: Gene by gene significance values for the EUR.IBD.gwas.ichip.meta.release.CD GWAS for the genes in the Team Tusk Network 1 module

it has been studied for its role in colon leukocyte recruitment in pediatric Crohns disease [32]; it is also known that dendritic cells from Crohns disease patients show aberrant STAT1 and STAT3 signalling [21]. There is evidence for STAT4 association with colonic Crohns disease and early disease onset [14], and a SNP in the STAT5 gene that favors colonic as opposed to small-bowel inflammation in Crohns disease was also identified [9]. The JAK genes also show connections to Crohns disease. Along with STAT3 [2], which has also been a target of drug development, the Phase 2 FITZROY Study showed that an Oral JAK1 Selective Inhibitor induced clinical remission in patients with moderate-to-severe Crohns disease [29]. A JAK2 variant that alters the intestinal barrier as one mechanism of action has been shown to be active in Crohns disease [24], and there has been a patent application for using genetic variants of JAK3 to diagnose and predict Crohns disease (Taylor et al., 2009) . Several other genes in this module were known to be associated more generally with Inflamatory Bowel Disease; based on their presence in this module, we now predict that they associate more specifically with Crohns disease. For example, EBI3 was known to be active in IBD, perhaps activated by Epstein-Barr virus infection [13]. Also, Zahn et al. [33] looking primarily at ulcerative colitis, note reduced AQP11 expression in patients with Crohns disease. Some of the genes in this module do not have a known association with Crohns disease or IBD more generally: we make new predictions that CISH, CBLB, and THPO are important to Crohns disease pathology on the strength of this disease module. There are rare familial mutations in THPO that are associated with elevated platelet count [31, 18], and both for both Crohns disease and ulcerative colitis, unexplained abnormalities of platelet number and function have been noted [11]. Polymorphisms in the CISH gene has known associations with increased susceptibility to TB and Leprosy; but Schurr et al. [27] notes also a possible connection to Crohns disease. CBLB has known associations with Type 1 Diabetes [**?**] and Multiple Sclerosis [25] but prediction of a Crohns disease association appears to be new.

4 Code Release and Availability

To assist with replicating our contest results, we have provided 1) Pre-computed DSD matrices for Networks 1-6 (both undirected and our directed with low-weight back-edges version for the directed Network 3), as well as for combined networks... ?? We provide four versions of these matrices: 1) Exact converged DSD 2) DSD step 7 3) the AMG-approximated converged DSD matrices that were used for our contest submission 4) improved AMG-approximated converged DSD matrics, using the post-contest results in [17].

The winning collection of modules that we submitted for the challenge is available here.

To employ our method on new networks that were not part of the DREAM challenge, you will have to compute DSD distance matrices. Optimized code for computing exact DSD matrices is available from [**?**]. Improved AMG-approximated code is newly released with this paper and is available here. Discuss!

A robust and Dockerized implementation of the top 3 methods from the DREAM challenge (including ours) is available from [28]. Depending on the size of your networks, our method might be slow because it computes the DSD distance matrices, so you might want instead to use the AMG-approximated code.

5 Author Contributions

Designed the winning DREAM method: JC,JL,BH,XH,DKS,LJC. Ran experiments during the DREAM challenge: JC. Ran experiments after the DREAM Challenge concluded: JC and HS. Worked on the AMG speedup: JL KD and XH, Analyzed results: JC, HS, DKS and LJC. Wrote the paper: JC, HS, XU, DKS and LJC.

6 Acknowledgments

We thank the Tufts BCB group and the DREAM conference organizers for helpful discussions. This research was partially supported by NSF grant DMS-1812503 (to LC and XH) and the Tufts T-Tripods Institute (NSF HDR grant 1934553).

References

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

[2] Florian Beigel, Matthias Friedrich, Corina Probst, Karl Sotlar, Burkhard Göke, Julia Diegelmann, and Stephan Brand. Oncostatin M mediates STAT3-dependent intestinal epithelial restitution via increased cell proliferation, decreased apoptosis and upregulation of SERPIN family members. *PloS one*, 9(4):e93498, 2014.

[3] Arthur Brady, Kyle Maxwell, Noah Daniels, and Lenore J Cowen. Fault tolerance in protein interaction networks: stable bipartite subgraphs and redundant pathways. *PloS one*, 4(4):e5364, 2009.

[4] Menfei 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.

[5] 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.

[6] Sarvenaz Choobdar, Mehmet E Ahsen, Jake Crawford, Mattia Tomasoni, Tao Fang, David Lamparter, Junyuan Lin, Benjamin Hescott, Xiaozhe Hu, Johnathan Mercer, et al. Assessment of network module identification across complex diseases. *Nature methods*, 16(9):843–852, 2019.

[7] Kek Heng Chua, Jin Guan Ng, Ching Ching Ng, Ida Hilmi, Khean Lee Goh, and Boon Pin Kee. Association of NOD1, CXCL16, STAT6 and TLR4 gene polymorphisms with Malaysian patients with Crohns disease. *PeerJ*, 4:e1843, 2016.

[8] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[9] Tara M Connelly, Walter A Koltun, Arthur S Berg, John P Hegarty, David Brinton, Sue Deiling, Lisa S Poritz, and David B Stewart. A single nucleotide polymorphism in the STAT5 gene favors colonic as opposed to small-bowel inflammation in Crohns disease. *Diseases of the Colon & Rectum*, 56(9):1068–1074, 2013.

[10] Lenore Cowen, Trey Ideker, Benjamin J Raphael, and Roded Sharan. Network propagation: a universal amplifier of genetic associations. *Nature Reviews Genetics*, 18(9):551, 2017.

[11] Silvio Danese, Carol De La Motte, and Claudio Fiocchi. Platelets in inflammatory bowel disease: clinical, pathogenic, and therapeutic implications. *The American journal of gastroenterology*, 99(5):938, 2004.

[12] Andrew Gallant, Mark DM Leiserson, Maxim Kachalov, Lenore J Cowen, and Benjamin J Hescott. Genecentric: a package to uncover graph-theoretic structure in high-throughput epistasis data. *BMC bioinformatics*, 14(1):23, 2013.

[13] Thomas Gehlert, Odile Devergne, and Gerald Niedobitek. Epstein–Barr virus (EBV) infection and expression of the interleukin-12 family member EBV-induced gene 3 (EBI3) in chronic inflammatory bowel disease. *Journal of medical virology*, 73(3):432–438, 2004.

[14] Jürgen Glas, Julia Seiderer, Melinda Nagy, Christoph Fries, Florian Beigel, Maria Weidinger, Simone Pfennig, Wolfram Klein, Jörg T Epplen, Peter Lohse, et al. Evidence for STAT4 as a common autoimmune gene: rs7574865 is associated with colonic Crohn's disease and early disease onset. *PloS one*, 5(4):e10373, 2010.

[15] R. Kelley and T. Ideker. Systematic interpretation of genetic interactions using protein networks. *Nature Biotechnology*, 23(5):561–566, 2005. doi: 10.1038/nbt1096 PMID: 15877074.

[16] David Lamparter, Daniel Marbach, Rico Rueedi, Zoltán Kutalik, and Sven Bergmann. Fast and rigorous computation of gene and pathway scores from snp-based summary statistics. *PLoS computational biology*, 12(1):e1004714, 2016.

[17] Junyuan Lin, Lenore J Cowen, Benjamin Hescott, and Xiaozhe Hu. Computing the diffusion state distance on graphs via algebraic multigrid and random projections. *Numerical Linear Algebra with Applications*, 25(3):e2156, 2018.

[18] Kun Liu, Robert Kralovics, Zbigniew Rudzki, Barbara Grabowska, Andreas S Buser, Damla Olcaydu, Heinz Gisslinger, Ralph Tiedt, Patricia Frank, Krzysztof Okon, et al. A de novo splice donor mutation in the thrombopoietin gene causes hereditary thrombocythemia in a Polish family. *Haematologica*, 93(5):706–714, 2008.

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

[20] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[21] Janne K Nieminen, Mirja Niemi, Taina Sipponen, Harri M Salo, Paula Klemetti, Martti Färkkilä, Jukka Vakkila, and Outi Vaarala. Dendritic cells from Crohns disease patients show aberrant STAT1 and STAT3 signaling. *PloS one*, 8(8):e70738, 2013.

[22] 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.

[23] Michael James David Powell. Restart procedures for the conjugate gradient method. *Mathematical programming*, 12(1):241–254, 1977.

[24] Matthias Prager, Janine Büttner, Verena Haas, Daniel C Baumgart, Andreas Sturm, Martin Zeitz, and Carsten Büning. The JAK2 variant rs10758669 in Crohns disease: altering the intestinal barrier as one mechanism of action. *International journal of colorectal disease*, 27(5):565–573, 2012.

[25] Serena Sanna, Maristella Pitzalis, Magdalena Zoledziewska, Ilenia Zara, Carlo Sidore, Raffaele Murru, Michael B Whalen, Fabio Busonero, Andrea Maschio, Gianna Costa, et al. Variants within the immunoregulatory cblb gene are associated with multiple sclerosis. *Nature genetics*, 42(6):495, 2010.

[26] Kayoko Sato, Mizuho Shiota, Sayaka Fukuda, Eiko Iwamoto, Haruhisa Machida, Tatsuo Inamine, Shinji Kondo, Katsunori Yanagihara, Hajime Isomoto, Yohei Mizuta, et al. Strong evidence of a combination polymorphism of the tyrosine kinase 2 gene and the signal transducer and activator of transcription 3 gene as a DNA-based biomarker for susceptibility to Crohns disease in the Japanese population. *Journal of clinical immunology*, 29(6):815–825, 2009.

[27] Erwin Schurr and Philippe Gros. A common genetic fingerprint in leprosy and Crohn's disease?, 2009.

[28] Mattia Tomasoni, Sergio Gómez, Jake Crawford, Weijia Zhang, Sarvenaz Choobdar, Daniel Marbach, and Sven Bergmann. Monet: a toolbox integrating top-performing methods for network modularisation. *bioRxiv*, page 611418, 2019.

[29] Severine Vermeire, Stefan Schreiber, Robert Petryka, Tanja Kuehbacher, Xavier Hebuterne, Xavier Roblin, Maria Klopocka, Adrian Goldis, Maria A Wisniewska-Jarosinska, Andrei Baranovsky, et al. Filgotinib (GLPG0634), an oral JAK1 selective inhibitor, induces clinical remission in patients with moderate-to-severe Crohn's disease: results from the phase 2 FITZROY study interim analysis. *Gastroenterology*, 150(4):S1267–S1267, 2016.

[30] Zhengting Wang, Lei Wang, Rong Fan, Jie Zhou, and Jie Zhong. Association of IL-27 gene three polymorphisms with Crohns disease susceptibility in a Chinese Han population. *International journal of clinical and experimental pathology*, 7(12):8952, 2014.

[31] Adrian Wiestner, Ronald J Schlemper, Anthonie PC van der Maas, and Radek C Skoda. An activating splice donor mutation in the thrombopoietin gene causes hereditary thrombocythaemia. *Nature genetics*, 18(1):49, 1998.

[32] Tara A Willson, Benjamin R Kuhn, Ingrid Jurickova, Shaina Gerad, David Moon, Erin Bonkowski, Rebecca Carey, Margaret Collins, Huan Xu, Anil G Jegga, et al. STAT3 genotypic variation and cellular STAT3 activation and colon leukocyte recruitment in pediatric Crohn disease. *Journal of pediatric gastroenterology and nutrition*, 55(1):32, 2012.

[33] Alexandra Zahn, Christoph Moehle, Thomas Langmann, Robert Ehehalt, Frank Autschbach, Wolfgang Stremmel, and Gerd Schmitz. Aquaporin-8 expression is reduced in ileum and induced in colon of patients with ulcerative colitis. *World Journal of Gastroenterology: WJG*, 13(11):1687, 2007.