Bio331 Fall 2019 Group Project Report

Emily Crook, Maddy Doak, Miffy Guo, Jonah Kohn, Tunc Kose, Hannah Mead, Gabe Preising, Tobias Rubel Jansson, Sol Taylor-Brill, Karl Young, and Julia Yuan

***Abstract*—This document describes the Reed College Bio331 group project to predict candidate regulators of non-muscle myosin II (NMII) for follow-up experimental validation in Reed's cell biology course.**

I. Introduction

A. Biological Background

1) Apical Constriction (Gabe):

Epithelial morphogenesis, or the rearrangement and specialization of epithelial cells, is one of the main processes that drive morphological changes during eukaryotic development. In their unspecialized form, epithelial cells make up the lining of the stomach, the skin, and secretory glands, where they mediate absorption and secretion while simultaneously serving as a protective barrier. Structurally, epithelial cells form sheets that are joined together by protein complexes at cell-cell junctions. The unique properties of these complexes drive whether or not an epithelial cell will specialize. Apical constriction, or cellular constriction of the apical side, facilitates epithelial morphogenesis. When a sheet of epithelial cells undergoes apical constriction, the inward forces exerted by neighboring cells causes the sheet to invaginate and form a protrusion [1]. This protrusion can undergo further specialization, resulting in the formation of organs or other biological structures.

2) Non-Muscle Myosin II (Karl):

In order to migrate or change shape, cells need to be able to exert some mechanical force on their environments. Proteins that are able to generate mechanical force via hydrolysis of ATP are known as motor proteins. Motor proteins can be broadly divided by the kinds of microscopic filaments they associate with; myosins utilize actin filaments while dyneins and kinesins use microtubules. Myosins are best known for their role in generating muscle movements, but all eukaryotic cells express myosin in some form. The basic myosin molecule has three citical domains that inform its function: a head, neck and tail. The head domain is the most conserved because it performs the critical action: binding actin and ATP and generating the actual force. The neck acts as a flexible connection between the head and the rest of the molecule, and interacts with specific regulators. The tail domain is the least conserved, and determines the myosins specific activity (think complexing with specific target molecules for different tasks) [2].

Non-muscle myosin II (NMII) is a well characterized molecule that plays a central role in cell morphogenesis. In *Drosophila melanogaster*, NMII is a hexamer consisting of 3 polypeptides that each appear twice: the heavy chain (Zipper/zip), essential light chain (myosin light chain/Mlc-c) and regulatory light chain (Spaghetti squash/sqh). A crucial method of activation/deactivation is by phosphorylation of the regulatory light chain sqh. In sqh's unphosphorylated form, the tail region forms a hairpin-like fold to interact with the head, preventing actin/ATP binding. Phosphorylation disrupts the ability for the tail to interact with the head, unfolding the molecule and placing in the active state [3].

3) Fog Signaling (Sol):

The Folded gastrulation (Fog) pathway results in apical constriction in specific tissues in *D. melanogaster*, and is an important driver of epithelial morphogenesis. Cells in epithelial tissues that are fated to undergo apical constriction-based shape changes express the large secreted protein, Fog, which interacts with cells primarily through autocrine, rather than paracrine, signaling [4]. The Fog ligand binds to Mist (mesoderm invagination signal transducer), a seven-pass, transmembrane, G-protein coupled receptor. Fog activates Mist, which is believed to stimulate the activation of the G-protein to which it is bound, Concertina (Cta). Cta, which exists as part of a trimer in its inactive state, exchanges its GDP for GTP, and is then able to dissociate from the other two trimer proteins, Gβ and Gγ. The activated Cta binds to RhoGEF2, a guanine nucleotide exchange factor (GEF). The Cta-RhoGEF2 complex activates Rho1 by stimulating its release of GDP and binding of GTP. GTP-bound Rho1 goes on to activate Rok (Rho-kinase), which phosphorylates the regulatory light chain of non-muscle myosin II (NMII) sqh [4]. The phosphorylation of sqh finally results in constriction of the actin network by NMII fibers in affected cells.

Transcription of Fog is thought to be regulated by the transcription factor (TF) Twist, whereas Mist transcription is regulated by the TF Snail [4]. The expression of Twist and Snail are thought to be controlled by dorsal-ventral patterning as well as by physical forces acting on tissues. In addition to the main pathway just discussed, other proteins are known to be involved in the regulation of Fog signaling as well.

Fog signaling-mediated apical constriction is involved in several morphogenic processes in *D. melanogaster* including invagination of the ventral furrow (VF) and posterior midgut (PMG), internalization of salivary glands, and the folding of the imaginal wing disc [4]. Mutant embryos that lack expression of the Fog protein are unviable and display aberrant constriction of the VF, demonstrating the importance of Fog signaling in development. Although the ligand and receptor involved in Fog signaling are specific to flies, protein signaling pathways that begin with the activation of a G$\alpha_{12/13}$ protein and result in actin rearrangement by Rho are conserved across

Template and outline provided by Anna Ritz (aritz@reed.edu).

many species, including humans [4], indicating that insight into Fog signaling may result in an enhanced understanding of signaling pathways related to human health and development. Although the Fog signaling pathway is highly studied, certain mechanisms remain unknown, such as how exactly physical forces affect Fog signaling, and how Fog signaling is terminated.

B. Motivation

1) Major Question (Julia):

Given our knowledge of apical constriction, NMII, and the fog signaling pathway, our question is as follows: Can we accurately predict previously unknown regulators of NMII and the Fog pathway? Furthermore, what computational methods can we use to make these predictions? Our ultimate goal is to produce a ranked list of candidate genes that can be tested by Derek's students in the wet lab environment.

2) Why We Care (Emily):

Predicting regulators of NMII using computational methods, while not always accurate, is a key part of apical constriction pathway research. Computational methods are vital for the comparison and consolidation of information in different databases. Due to the error margins of different research methods, data from two sources might be in conflict. Combining interactomes using computational biology allows researchers to estimate the validity of existing information. Resources for physical research into possible regulators are limited. Using existing information to model likely pathways or to find significant patterns enables physical research to be directed towards the most likely candidate genes. A targeted approach to interactome research using computational methods is both more efficient and more effective.

C. Computational Approach

1) Protein-Protein Interaction Networks (Tobias):

Protein-Protein Interaction Networks (PPIs) are combinatorial graphs which represent observed interactions between proteins. Given a collections of proteins, create a set V of their names. Then, create a set $E \subseteq V \times V$ which contains an edge (u, v) whenever u and v have been found to interact. Then, a PPI is just a graph $G = (V, E)$. For instance if we wanted to model the interactions of proteins A, B, ..., L we might end up with a graph which looks like Figure 1. A variety of experimental methods and experiments might go into the creation of a given PPI. We used an interactome made by Anna, which is a weighted combination of six previously established databases.

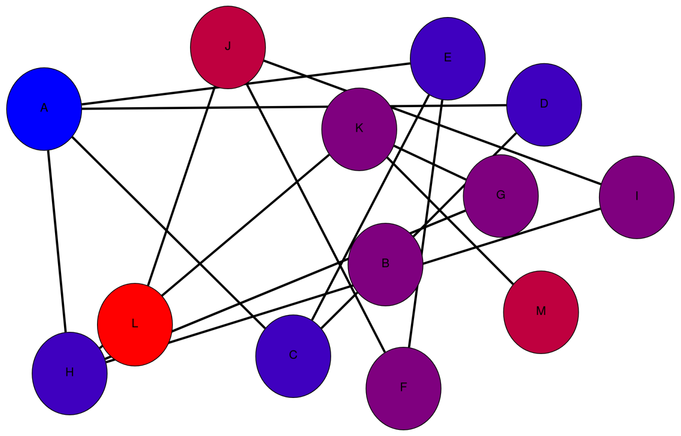

Fig. 1. A very small PPI might look like this.

2) Fly PPI Statistics (Tunc):

The fly interactome used contains 11473 proteins and 233054 interactions in between them. The average node degree of the interactome is 41 proteins, however the median degree is 16. The mean average neighbor degree across all degrees is 325 and the median is 229. This difference in order of magnitude of average node and neighbor degrees suggest that the interactome is disassortative to an extent, where low-degree nodes tend to be connected to high degree-nodes. This idea is supported by the interactomes low average clustering coefficient (0.16). In other words, there seem to be some proteins connected to many other proteins, the latter of which are not connected to many proteins. The interactome appears to be scale-free (Fig. 2). The interactome is composed of 9 connected components, 8 of which are made up of 2 or 3 nodes.

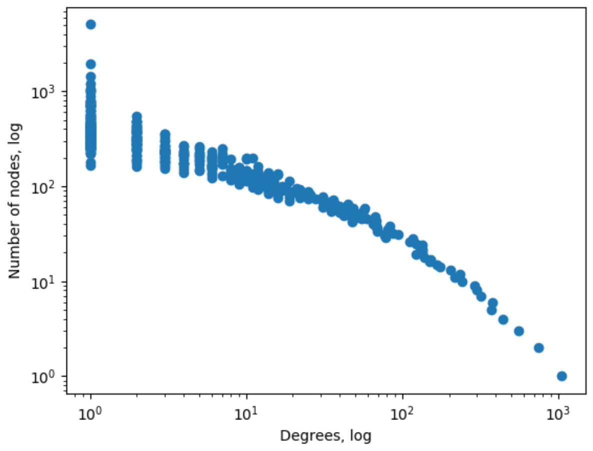

Fig. 2. Degree distribution of the interactome

3) Positive Examples (Miffy):

Anna Ritz selected a set of 104 proteins to serve as known positives. These proteins are known to be involved in Fog signaling (19 positives), apical constriction (22 positives), and folded gastrulation (89 positives) pathways. The proteins involved in Fog signaling are identified from a review paper based on wet-lab experiments [4]. The proteins involved in apical constriction and folded gastrulation are identified by gene ontology, which includes a computational representation of the biological functions of genes and gene products. All positively labelled proteins were already present in the network, so no new proteins were added.

4) Negative Examples (Maddy):

Anna Ritz selected a set of 470 proteins that are extremely unlikely to be involved in the Fog pathway due to their confirmed functions, and thus can be classified as confident negatives in the interactome. These proteins are known to be involved in autophagy (153 proteins), pigmentation (76), regulation of cell growth (72), regulation of circadian rhythm (69), and regulation of immune response (141), as determined from experimental evidence. These categories were chosen because they were not expected to overlap with apical constriction; for example, a protein with a specific function (one that is not necessary for general cell functioning and survival) will likely not be involved in both pigmentation, giving a cell color, and cell rearrangement and specialization, including apical constriction. The source of these proteins and their categories was the Gene Ontology. All negatively labeled proteins were already present in the interactome; no new proteins were added to the network based on these categories.

5) Semi-Supervised Learning (Karl):

The basic premise of machine learning is that we'd like the computer perform some task that we haven't explicitly programmed it to do. A classic example is if we pass the computer a picture, we might ask it to guess whether the picture is of a cat or a dog. In order to do this, we probably should give the computer some examples of cats and dogs and tell it which ones are which, so it can learn the patterns to look for. In general, if the input data (in this case, example pictures) all have associated output values (cat/dog labels), the process is called supervised learning. Alternatively, we might want to learn something about the intrinsic properties of the input data (for instance, if all dog pictures had red backgrounds and all cat pictures had blue backgrounds, they might separate just based on the mean color value of each photo). These kinds of problems don't necessarily require labels, and form the branch of unsupervised learning.

Semi-supervised learning falls in between supervised and unsupervised learning methods, when researchers have datasets that are only partially labeled. In our case, we have a small subset of proteins that we are confident are significant players in the Fog pathway. Unfortunately, we don't know the extent to which most proteins are involved, but we do have the PPI network. So, we're going to try to use both the unlabeled data (the interactome) and the labeled data (positive/negative regulators) to predict new regulators.

6) Useful Graph Algorithms (Hannah and Jonah):

a) Intro (Hannah): Graph algorithms are useful in discerning gene ontology because they allow us to make predictions based statistical analysis of complex biological structures. The following algorithms are a small section of those implemented in our body of work, but show common strategies used in interaction prediction.

b) Random Walks (Jonah): Random walk algorithms are used to establish the likelihood of nodes being visited, given stochastic travel between nodes via edges. There are a number of ways to generate random walks, including the famous PageRank algorithm developed by Google to characterize travel between websites. Random walks are useful in understanding the difficulty of travel between two nodes, and can provide an interesting metric for distance given enough random travel across a graph.

c) Breadth First Search (Hannah): Breadth First Search (BFS) is an algorithm which searches a graph. As input it takes a source node and a target, and outputs, dependant on the implementation, either a True/False value to represent is the target is/isn't there or a path from source to target. This method searches the adjacent nodes to the source, and stores those adjacent nodes to be searched after the first round is complete. This process is repeated, essentailly searching layer by layer, until the target is found.

d) Dijkstra's algorithm (Jonah): Djikstra's algorithm is a method for computing the shortest path from any source node in a graph to any other target node in a graph. At every step, it recomputes the cost of outgoing edges to reflect the total cost of the path from the source to the future node. Djiktra's algorithm's use is widespread, and many variations of the algorithm have been employed for specialized purposes, such as a variation that computes ALL shortest paths between a source and target node, as opposed to one.

e) Minimum Spanning Trees (Jonah): A minimum spanning tree is a subset of edges in a graph that create a connected graph, such that the cumulative weight of the edges in the tree is as low as possible. it is possible for multiple minimum spanning trees exist for any given graph, and many algorithms exist for computing minimum spanning trees. A classic greedy algorithm for computing minimum spanning trees is Primm's algorithm, which beings with an empty tree and incrementally adds the cheapest edge which connects a new node to the tree. Minimum spanning trees can be employed for various path analyses, including Steiner Trees.

f) Steiner Trees (Hannah): Given a graph, G and a set of nodes, called 'terminals', a Steiner tree is a minimum weight tree that connects all the terminal nodes, while only using edges and nodes present in G. A subtype of these is called Prize Winning Steiner Tree. This formulation of the problem is game-like, the terminals and the movements between elements are given positive and negative values, the goal is then to maximize the score. The terminals may be avoided if the cost of graph traversal outweighs the benefits of obtaining one of the 'prize'terminal nodes.

II. Methods and Results

A. Randomly Choosing Candidates (Anna)

Consider all the nodes from the protein-protein interactome. These nodes include a small number of labeled proteins (e.g. positive and negative examples); the majority of proteins in the interactome are unlabeled. We used `random.choice()` to randomly pick 10 unlabeled that as our candidates (Table I).

B. Common Neighbors of Known Positives (Sol)

Find the neighbors of all the known positives, and assign each a weight based on how many known positives they are

Rank	Protein Name
1	schlank
2	CG15172
3	CG3344
4	Sec15
5	CG5844
6	CG31206
7	HLH3B
8	CG15283
9	Set1
10	CG5334

TABLE I
RANDOMLY-SELECTED CANDIDATES (RANK DOESN'T MEAN MUCH HERE).

connected to. Next, find the neighbors of all of the nodes in the graph and score each node based on the sum of the weights of the nodes to which they connect which are the neighbors of known positives. The top 10 are chosen as candidates (Table II). Note: nodes were chosen by number of known-neighbors they were attached to *not* the percentage of their neighbors that were attached to known-positives, so this method is somewhat biased toward nodes with a higher overall degree.

C. Clustering Coefficients (Sol)

Given a binary matrix representation of the graph (A_{uw}) let d_v be the degree of node *v* and N_v be the neighbors of node *v*. Find the clustering coefficient:

$$C(v) = \frac{\Sigma_{u \in N_v} \Sigma_{w \in N_v} A_{uw}}{d_v(d_v - 1)}$$

of all the nodes in the graph. Then, select the nodes that have a clustering coefficient that is closest to the mean clustering coefficient of the known positives (0.05812) (Table II).

Rank	Clustering Proteins	Common-Neighbor Proteins
1	Blm	Ubi-p63E
2	Cchl	Ubi-p5E
3	unc-104	smt3
4	CG1688	Hsp83
5	Socs44A	drk
6	FoxP	Appl
7	Nmnat	14-3-3ζ
8	hh	Nab2
9	Rbf	Rbp9
10	nrv2	fne

TABLE II
CANDIDATES THAT HAVE SIMILAR CLUSTERING COEFFICIENTS TO KNOWN POSITIVES (LEFT) AND CANDIDATES THAT SHARE A HIGH NUMBER OF COMMON NEIGHBORS WITH KNOWN POSITIVES (RIGHT)

D. Random Walk on a Lazy BFS (Hannah)

First I modified the graph by removing the negatives and adding a Super Source, this connects multiple nodes on the graph together, I chose Fog, CTA and RHO1, as they are keystones to the gene pathway. I used a breadth first method of searching the graph from the Super Source to all known positives, this essentially truncated the graph at the same distance from each source. Next I conducted random walks on the graph from known positives back to the sources. Results in Table III

Rank	Protein Name
1	Spn
2	NetB
3	CG7164
4	CG10347
5	Khc
6	Ten-m
7	dally
8	Ino80
9	gro
10	Ubi-p5E

TABLE III
CANDIDATE TABLE (HANNAH, IN-PROGRESS, NEED TO FILTER OUT THE BIO-LAB POSITIVES).

E. Low-degree-First Search (LFS) (Tunc)

Starting at spaghetti squash (sqh), put all the previously unseen neighbors of the current node along with a priority to a queue, sort the queue by increasing priority and repeat. The priority of sqh is 0 and of all consequent nodes n is

$$P(n) = \omega P(n_{current}) + d_n$$

where ω is a user-defined parameter and d_n is the degree of the node. Having $\omega > 1$ results in a Breadth-first search (BFS)-like traversal and $\omega < 1$ results in a Depth-first search (DFS)-like traversal. Results obtained with $\omega = 0.5$ can be found on Table IV.

Rank	Protein Name
1	CG12310
2	Karl
3	CG4022
4	CG12516
5	CG10834
6	CG7567
7	CG44002
8	Npc2e
9	CG31176
10	CG31816

TABLE IV
TOP 10 CANDIDATES FOUND VIA LFS, $\omega = 0.5$.

F. Dijkstra-all on Pairs of Related Nodes (Julia)

Using a selected set of 13 proteins known to be involved in the Fog signaling pathway, find the shortest paths between all pairs of these proteins using Dijkstra's Algorithm [4]. Sum the frequencies of nodes involved in all these shortest paths, and nodes with the highest frequencies have the best rankings. Nodes that are known positives were filtered out, and the remaining nodes with the highest frequencies are listed as the top 10 candidates (Table V).

G. Dijkstra's Algorithm Variation (Maddy)

We found the five shortest paths in the protein-protein interactome from Fog to Sqh, the two primary proteins involved in the NMII pathway, using Yen's Algorithm (an expansion on Dijkstra's algorithm that finds the "k" (a specified number) shortest paths between two nodes in a graph) [5]. Fog is a protein described in section A.3 of the introduction, and Sqh is the light chain of NMII, also described in section A.3. Of these

Rank	Protein Name	Frequency
1	drk	36
2	dsh	24
3	Dab	22
4	N	22
5	Galphai	21
6	pins	19
7	H	11
8	CtBP	11
9	flw	11
10	Axn	8

TABLE V
CANDIDATES THAT APPEAR WITH THE HIGHEST FREQUENCY IN THE SHORTEST PATHS.

shortest paths, we picked the path with the greatest proportion of positively-labeled proteins (proteins known to be involved in the pathway), the ”best” shortest path, and then removed the unlabeled proteins in that path from the graph and added them to a list of candidates. If ever there were no unlabeled proteins in the path, the next-best path with unlabeled proteins was chosen. We repeated this until there were at least 10 candidates in the list. The top 12 may be seen in Table VI as well as Figure 3.

Rank	Path	Protein Name
1	1	Pp1-87B
2	2	flw
3	2	Roc1b
4	2	ci
5	2	CkIalpha
6	2	Axn
7	4	l(2)gl
8	4	numb
9	4	N
10	7	CycK
11	7	Cdk2
12	7	Dsor1

TABLE VI
CANDIDATES FROM THE “BEST” SHORTEST PATHS FROM FOG TO SQH WITH REMOVAL; RANK IS NOT VERY SIGNIFICANT, BUT IS RELATED TO PATH NUMBER. “PATH” IS NUMBERED BY THE ITERATION DURING WHICH THAT PATH WAS CHOSEN.

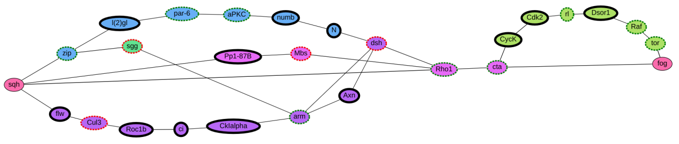

Fig. 3. Visualization of the “best” shortest paths from Fog to Sqh for the top 12 candidates. Candidates have bold black borders, and paths are colored from pink, to purple, to blue, to green, in order from first to last. Positively-labeled proteins have dashed green borders while negatively-labeled proteins have dashed red borders.

H. Dynamic Yen’s KSP Frequency Analysis (Jonah)

An implementation of Yen’s k shortest path algorithm was used to find shortest paths from fog to sqh. Yen’s k shortest paths is an algorithm that uses Djikstra’s shortest path algorithm to find the shortest path from a source to a target, and then removes an edge from the shortest path to find an alternative shortest path. This motif is repeated k times, to find k unique shortest paths. This algorithm was used to calculated the k shortest paths for all k from 1 to 100. For each value of k, every node’s frequency of appearance in the k shortest paths was recorded. These frequencies were then graphed in relation to their k, and a best-fit line was created for every node. Higher sloped lines as k increases were understood to correlate to more involved interactions in the pathway between these two nodes, and candidates were ranked based on the steepness of their best-fit line’s slope.

Rank	Protein Name	Slope
1	spn-A	0.000851
2	CG1774	0.000723
3	CG6178	0.000573
4	CycK	0.000505
5	Cul1	0.000416
6	Cdk7	0.000404
7	CG17841	0.000376
8	Cdk2	0.000358
9	Rev1	0.000354
10	CG8360	0.000347

TABLE VII
CANDIDATES WHICH MAINTAIN POSITIVE SLOPES FOR APPEARANCE IN THE K SHORTEST PATHS FROM 1 TO 100.

I. Frequently Occurring Nodes in Minimum Spanning Trees Generated from Metric Closures (Emily)

We began with two versions of the fly network, one weighted and one unweighted. These were created by converting the weights to costs (1-weight) for the weighted graph, and by setting all the costs equal to 1 for the unweighted graph. For each graph we found the metric closure, using positively labeled proteins as terminals.

The metric closure of a graph G_L is a graph in which the weight of each edge between terminals (u,v) is the total cost of the shortest path between u and v in the original graph G. The shortest path is found between each terminal using Dijkstra’s algorithm and by prioritizing the lowest edge cost. We took the shortest paths between each terminal node and the source node s and combined their edges. This created a set of edges connecting the source node s and every other terminal node through their shortest paths.

A minimum spanning tree was calculated between this set of edges and the terminals. A minimum spanning tree was collected for every terminal node as source node s. Every unlabeled node in each minimum spanning tree was combined for the weighted graph, and every unlabeled node in each minimum spanning tree was combined for the unweighted graph. Edges that occurred in both weighted and unweighted edge collections were found by taking the intersection of the two data sets. Unlabeled nodes were ranked by how often they occurred in the intersecting edges. The top ten candidate are listed below (Table X).

J. Random Walks on Minimum Steiner Tree with Known Positives as Terminals (Gabe)

We approximated the minimum Steiner tree for the fly network using every known positive as a terminal node. This

Rank	Protein Name
1	Drep2
2	Ten-m
3	CanB2
4	CG7164
5	CG9572
6	CG2199
7	NetB
8	prd
9	CycG
10	pll

TABLE VIII
CANDIDATES THAT OCCURRED MOST FREQUENTLY IN THE INTERSECTING EDGES.

produced a graph with 2 connected components: one larger component (200 nodes, 199 edges) and one smaller component (7 nodes, 6 edges). Next, we implemented a random-walk-with-teleports algorithm for undirected graphs. This algorithm will simulate a random walker from a source node s and do the following for k timesteps and probability q:

1) Visit Neighbor: With probability q, travel to a neighbor of s and update s to be the node the walker is on currently.

2) Teleport: With probability 1 - q, teleport back to the original s (i.e. reset the random walk).

We used the parameters k = 1000 and q = 0.75, to capture local network topology. We then ran the algorithm for each terminal node such that each terminal node was s exactly once and kept track of walker visits for every node in the network for all iterations. We report the top 10 candidates for this method in Table IX, ranked by the total number of walker visits over all iterations of the algorithm.

The smaller connected component of the graph was composed of 6 known positives and 1 unknown node (Fas2). While this may artificially increase the number of walker visits for nodes within this component, the fact that the one unknown node was connected exclusively to known positives suggests a high likelihood that Fas2 is involved in apical constriction; therefore, we included Fas2 in our results (Minimum Steiner Tree can be viewed here: http://graphspace.org/graphs/27855?user_layout=11980).

Rank	Protein Name	Walker Visits
1	gro	4041
2	drk	2620
3	ci	2232
4	Fas2*	2116
5	flw	1622
6	CG9572	1601
7	CtBP	1454
8	His3	1374
9	N	1347
10	H	1270

TABLE IX
MOST FREQUENTLY VISITED UNLABELED NODES IN STEINER TREE WITH POSITIVES AS TERMINALS (* FAS2 WAS THE ONLY UNKNOWN NODE IN A SMALL CONNECTED COMPONENT COMPRISED EXCLUSIVELY OF POSITIVE-LABELED NODES; THEREFORE, WE INCLUDED IT IN OUR ANALYSIS).

K. Random Walks on Steiner Tree that built from filtered Network (Miffy)

All negatively labeled proteins and their connective edges were removed from the original interactome network. Using all positively labeled proteins as terminals, we extracted a Steiner tree as a sub-network. The Steiner tree was fragmented into two large tree (n $>$ 30), two medium tree (n $>$ 10) and seven small tree (n $<$ 10), due to the fragmented nature of the filtered network (Figure 4). When two nodes were not connected in the filtered network, this edge in matrix closure was assigned with a large number and this edge was not expanded in Steiner tree, resulting in fragmented pieces of the Steiner tree. To rank the proteins in likeliness of being positives, the same random walk algorithm from Gabe was implemented on these Steiner for the purpose of cross-comparison. Top ten candidate are listed below (Table X).

Rank	Protein Name
1	gro
2	drk
3	flw
4	ci
5	N
6	ptc
7	snk
8	Fas2
9	HDAC1
10	Dsor1

TABLE X
CANDIDATES THAT WERE VISITED MOST FREQUENTLY IN THE FILTERED STEINER TREE.

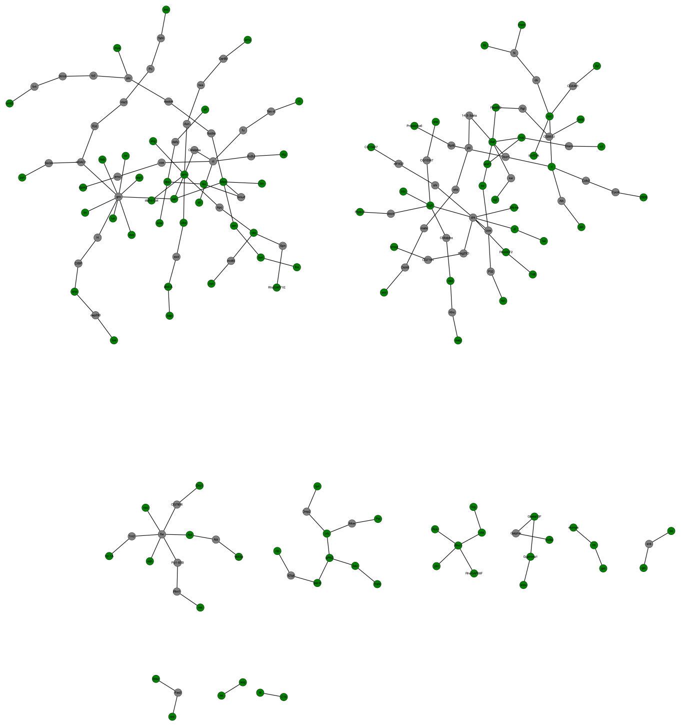

Fig. 4. Visualization of Fog Steiner trees built from the filtered network

L. Gaussian Smoothing (Karl)

The basic idea behind this smoothing technique is that we'd like to generate a function $f(v) : V \mapsto [0,1]$, where the values of the labeled nodes are all equal to 1, and the graph is "topologically smooth", i.e. the difference between two nodes is smaller if the two nodes are closer together. In technical terms, we're attempting to minimize the function:

$$\min_f \frac{1}{2} \sum_{(u,v)\in E} w_{uv}(f(u) - f(v))^2$$

where for an edge $(u, v) \in E$, its weight is given by w_{uv}. Ultimately, this will converge to a solution in which the value of every node is simply the average of its neighbors. So, we can start with some arbitrary distribution on the unlabeled nodes, and iteratively calculate the average of each node's neighbors until the system converges:

$$f_t(v) = \frac{1}{d_v} \sum_{u\in N_v} w_{uv} f_{t-1}(u)$$

where the weighted degree $d_v = \sum_{u\in N_v} w_{uv}$ and the set N_v denotes the neighbors of node v. To define convergence, we can simply check the inequality

$$\sum_{v\in V} |f_t(v) - f_{t-1}(v)| \le \epsilon$$

for whatever arbitrarily small ϵ we choose. A final note, I chose to ignore negative labels because this method heavily depends on how those negative values are distributed. Instead, I chose to attach a single negative node to every node in the graph, more evenly applying the negative pressure. For the specifics of implementation I'll refer to [6], but we can let the variable λ represent the strength of this negative connection, which along with ϵ and the graph G are enough to generate our list of candidates ($\lambda = 0.1$ and $\epsilon = 0.001$) in Table XI:

Rank	Protein Name
1	snk
2	pon
3	CG15172
4	CG12730
5	Spn77Bb
6	CG5964
7	CG3592
8	CR10102
9	CG14657
10	CG6974

TABLE XI
CANDITATES WITH THE HIGHEST GAUSSIAN SMOOTHING SCORES

M. Node2Vec Similarity to X (Tobias)

Node2Vec is an algorithm for embedding graphs into a feature space such that feature representations of vertices preserve local neighborhoods in a d-dimensional space. In particular, node2vec attempts to maximize the log-probability of observing a network neighborhood with respect to sampling strategy S $N_S(u)$ for a vertex u conditioned on its vector representation $f(u)$ [7, 3]:

$$\max_f \sum_{u\in V} \log Pr(N_S(u) \mid f(u))$$

This is similar to feature learning by way of Skip-gram architectures which are used in natural language processing, but characterizing the neighborhood of a node is not as straightforward as characterizing the neighborhood of a word [7, 3]. In the case of a word in a sentence, we can make use of a sliding window which captures the w words before and after the word we are creating a feature representation of. For a graph, node2vec uses biased random walks of length l as the sampling method in order to capture the benefits of both breadth-first and depth-first search in a computationally efficient way.

Using node2vec we create a 128 dimensional model of the interactome. Given this model we consider two means of predicting candidate proteins.

1) Similarity to Known Regulators: For each known regulator find the k most similar vectors — using cosine similarity as the operational metric of similarity. Then, combine the lists of potential regulators into one by ranking the vectors by the number of known regulators they are in the k-most similar vectors list of. For k=10, the predictions can be seen in table XII

Rank	Protein Name
1	lmd
2	Scr
3	CG33458
4	Antp
5	Abd-B
6	CG9813
7	knk
8	abd-A
9	mid
10	eve

TABLE XII
CANDIDATES WHICH ARE AMONGST THE 10 MOST SIMILAR PROTEINS TO THE HIGHEST NUMBER OF KNOWN REGULATORS. NOTE THAT THE ORDERING IS PARTIAL RATHER THAN STRICT.

2) Similarity to sqh: Given the same node2vec embedding of the PPI, find the k most (cosine) similar vectors to sqh. The results for k=10 can be seen in table XIII.

Rank	Protein Name
1	CG42554
2	CG15544
3	alpha-Spec
4	Mlc-c
5	RpS19a
6	dgo
7	TAF1C-like
8	CCT2
9	SIFaR
10	CG8939

TABLE XIII
CANDIDATES RANKED BY SIMILARITY TO SQH. THIS IS A STRICT ORDERING IN PRACTICE, THOUGH NOT IN PRINCIPLE.

N. Exploring Sub-Graphs of Predicted Nodes (Sol)

Assemble a sub-graph from the top 10 ranked nodes from all of the other methods except randomly-chosen. Only nodes that were chosen as candidates and edges that go between candidates in the original interactome are included. Nodes are chosen from the subgraph in one of two ways:

1) Dijkstra with Edge Rankings on the Sub-Graph: Assign each node a nodecost based on its average ranking across different methods. For each method in *m* total methods, node *v* has a rank, m_j. The nodecost is 1 minus the average ranking across all methods (if node not ranked in a method, rank = 0) normalized to a value between 0-1 :

$$nodecost(v) = 1 - \frac{\sum\limits_{m_j \in m} (11 - rank(v, m_j))}{10m}$$

Assign each edge a cost equivalent to the average cost of the two nodes it connects. Then, run dijkstra-all, which finds all of the shortest paths between any two nodes, on all pairs of nodes in the subgraph. The nodes that are included in the highest number of shortest paths (ie have the largest betweeness centrality) are chosen as the top 10 candidates (Table XIV, Left).

2) Random Walks From Known-Positives on the Sub-Graph: Add known-positives and any edges between known-positives or between known-positives and candidate nodes to the graph. Add a super source node and edges that connect from the super source to every known positive. Run a random walk with restarts simulation starting from the super source for 100,000 time steps with a teleportation probability of 25%. The nodes that are visited most frequently (that are not known-positives) are chosen as the top 10 candidates (Table XIV, Right; 5).

Rank	Dijkstra Proteins	Random-Walks Proteins
1	Ubi-p63E	Ubi-p63E
2	gro	Hsp83
3	N	Appl
4	Hsp83	drk
5	flw	gro
6	Ten-m	NetB
7	CG4198	Ubi-p5E
8	drk	fne
9	Spn	Rbp9
10	l(2)gl	14-3-3ζ

TABLE XIV
SUBNETWORK CANDIDATES. CANDIDATES THAT HAVE THE HIGHEST BETWEENNESS CENTRALITY IN SUBNETWORK WITH EDGE-COSTS BASED ON NODE RANKING (LEFT) AND CANDIDATES THAT APPEAR MOST FREQUENTLY IN RANDOM WALKS FROM KNOWN-POSITIVES (RIGHT)

O. Intersections Between Methods (Sol)

Input lists of ranked nodes from all of the other methods used. Then categorize the proteins based on how many methods selected them as potential candidates. The top candidates are those which were predicted by the highest number of methods. (Table XV).

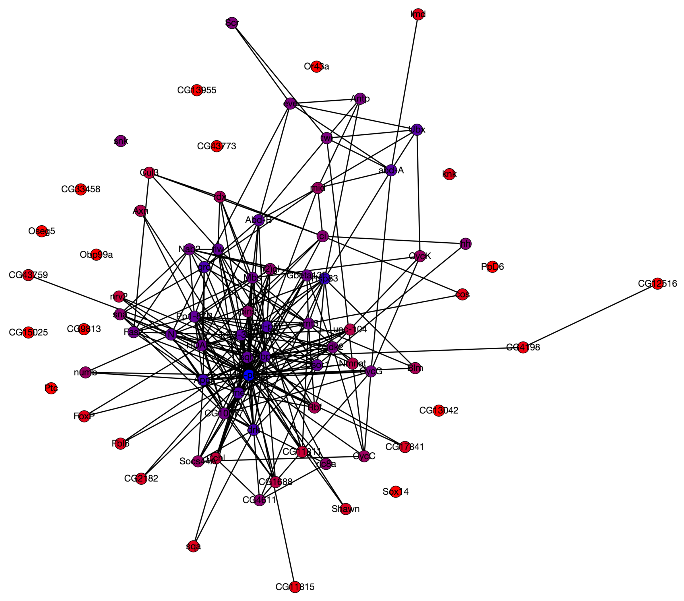

Fig. 5. Subnetwork of all candidates suggested by other methods. Nodes are colored by how frequently they are visited in random walks (red = low frequency; blue = high frequency). Known-positives not included.

Rank	#methods	Protein Name
1	5	gro
2	5	drk
3	5	N
4	4	flw
5	3	ci
6	3	Ubi-p63E
7	3	Spn
8	3	Hsp-83

TABLE XV
CANDIDATES THAT ARE RANKED WITHIN THE TOP 10 CANDIDATES IN THE HIGHEST NUMBER METHODS (INCLUDING SUBNETWORK CANDIDATES). PROTEINS LISTED IN THE SAME NUMBER OF METHODS ARE TIED WITH EACH OTHER. ONLY 8 PROTEINS ARE SHOWN SINCE THERE WERE 16 NODES THAT WERE SUGGESTED BY 2 METHODS.

III. DISCUSSION

A. Challenges and Considerations

1) Disconnected components in either filtering the interactome or in the output graphs (e.g. Steiner trees). In the case of the Steiner tree approximation, we suspect it is an artifact of the path expansion part of the algorithm.
2) RWs on a subnetwork of predicted nodes with known positives; some positives are not connected to any candidates.
3) Working on a big graph.
    a) Hard to visualize
    b) Running time concerns
4) Challenges in debugging / making sure algorithm is doing the right thing.
5) Shortest paths seem to be really long – Anna suspects that it is an artifact of the edge weighting. This hypothesis was confirmed by Tunc (data not shown).
6) The Interactome has edge weights; many algorithms require an edge costs. Weights need to be transformed

to costs.

B. Final Recommendations

These final recommendations are based on the number of times each candidate was suggested by various methods. Proteins that were listed by the same number of methods should be considered tied with each other. Proteins that were suggested by only two methods are not shown given the high quantity (16) (Table XV).

1) Suggested 5x: gro, drk, N
2) Suggested 4x: flw
3) Suggested 3x: ci, Ubi-p63E, Spn, Hsp-83

Fig. 6. Group photo from IEEE BIBM 2019 in San Diego, CA

Other Author Contributions

Final Editing/Fomatting: KY

References

[1] Jacob M Sawyer, Jessica R Harrell, Gidi Shemer, Jessica Sullivan-Brown, Minna Roh-Johnson, and Bob Goldstein. Apical constriction: a cell shape change that can drive morphogenesis. *Developmental biology*, 341(1):5–19, 2010.

[2] Miguel Vicente-Manzanares, Xuefei Ma, Robert S. Adelstein, and Alan Rick Horwitz. Non-muscle myosin ii takes centre stage in cell adhesion and migration. *Nature Reviews Molecular Cell Biology*, 10(11):778–790, Nov 2009.

[3] Rachel E. Farrow, Jeremy C. Fielden, Alex E. Knight, and Justin E. Molloy. *Single Molecule Studies of Myosins*, page 1–33. Elsevier, 2009.

[4] Alyssa J Manning and Stephen L Rogers. The fog signaling pathway: insights into signaling in morphogenesis. *Developmental biology*, 394(1):6–14, 2014.

[5] Yen's algorithm. *Wikipedia*, Mar 2019.

[6] Miriam Bern, Alexander King, Derek A. Applewhite, and Anna Ritz. Network-based prediction of polygenic disease genes involved in cell motility. *BMC Bioinformatics*, 20(S12):313, Jun 2019.

[7] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.