# Prize-Rich Hypershrubs in Directeded Signaling Hypergraphs

A Thesis
Presented to
The Division of Mathematics & Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Barney Isaksen Potter

May 2016

Approved for the Committee
(Interdisciplinary Committee for Mathematics & Biology)

Dr. Anna Ritz

Dr. James Fix

# Acknowledgments

I want to thank a few people:

- Mom and Dad
- Mari
- Far Far
- Sam, Will, Rich, and ext. family
- Helen and Melvin
- Michael and Caleb
- Anna and Jim
- Nicole, heather, and the Ritz Group
- Sarah, Jeremy, Leigh, and the Schaack Lab
- My benefactor
- A.T & A.L
- Riot Games & Valve
- The Paradox, Gigantic, Maru, Tea Chai Te, Tom Yum, The Lutz, Pambiche.

# Preface

*We can only see*
*a short distance ahead,*
*but we can see plenty there*
*that needs to be done.*

ALAN TURING

# Table of Contents

# List of Tables

# List of Figures

# Abstract

Cell signaling pathways are an important tool used by biologists to model how signals are transduced through cells. Though they can be (and often are) modeled using graphs, we instead use a generaliation known as a hypergraphs. Using these, we draw from the literature on hypergraphs and their traversal to develop the notion of a hypershrub, a multi-source, multi-target generalization of a hyperpath. We then draw on the literature on Prize-Collecting Steiner Trees and their applications to biological systems to create a formulation that finds specific, prize-dense hypershrubs that may be representative of biological phenomena that went previously unnoticed in standard graph representations of the same signaling pathways. Finally, we apply this formulation to the human Hedgehog signaling pathway, and the *Arabidopsis thaliana* interactome, to analyze the effectiveness of our algorithm in real data.

# Dedication

You can have a dedication here if you wish.

# Introduction

Since the discovery of cells by Robert Hooke in 1665, biologists have worked tirelessly to unlock the mysteries of cell function. Over the last half-century, our knowledge of how cells function has grown tremendously. An important and growing part of this field has been the growing use of computational methods in helping researchers to both test pre-existing hypotheses, and generate novel hypotheses. Computer-based methods have a wide range of applications, from DNA and RNA sequence analysis (Lander et al. (2001)) to high-throughput modeling of population growth (Anderson et al. (2005)), to the growing fields concerned with artificial neural networks (Chon et al. (1996)). Importantly, in recent years, biologists[1] have begun to employ some of the computational tools that computer scientists have been developing for decades. This recent synthesis of fields (see Figure 1) has created a new area of research, where existing algorithms and computational methodologies are being adapted and tweaked to help elucidate new meaning from biological data.

## 0.1 Signaling Pathways

One of the most interesting areas where computational methods are now being applied is in the study of signaling pathways within cells. Generally, signaling pathways are representations of what we currently know about how proteins and small molecules interact within cells in order to propagate signals throughout the cell, and ultimately change the cell's behavior. We find cell signaling pathways interesting because understanding how signaling information spreads throughout a cell helps us to better understand what disregulations can result in changes in cell dynamics, which often manifest themselves in heterogenous diseases such as cancer[2] (Taylor et al. (2009)). Signaling pathways have been studied for years, and they have often been represented visually as flow diagrams (Figure 2), but recently, researchers have begun to "convert"

[1] And other scientists, for that matter.

[2] As well as many other diseases in which signaling pathway disregulations have been implicated.

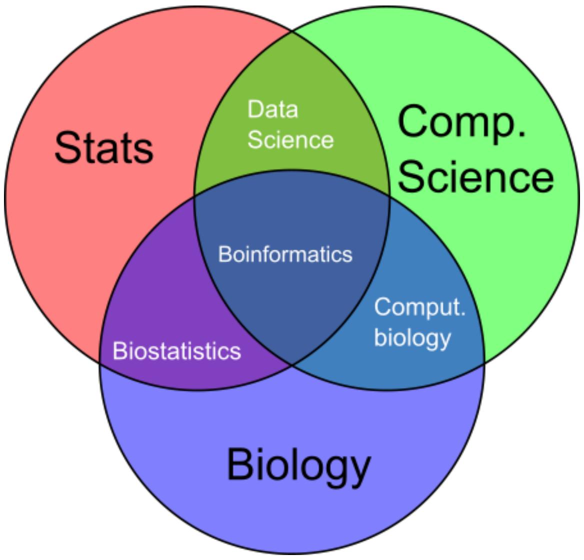


Figure 1: A venn diagram of the intersection between the fields of biology, computer science, and statistics. This thesis is primarily concerned with the field of computational biology. Image from `https://genomejigsaw.wordpress.com/2015/09/27/faq/`, Marek Cmero.

the traditionally drawn digrams of signaling pathways into mathematically intersting structures, for which known algorithms can be employed to learn more about the signaling pathways.

## 0.2 Network Representations

To build a mathematical model of cell signaling pathways, we often use a graph as a data structure that is representative of the structure of the pathways. Graphs are a data structure first described by Euler in 1736 (Shields (2012)) to help explain the complex relationships between objects[3] in a concise, well defined way. For the purpose of this thesis, the word "graph[4]" will refer to the data structure. A graph, $G = (V, E)$ is a structure that represents a set of objects, $V$ referred to as *nodes* or *vertices* and pairwise connections between them, $E$ referred to as *edges*. Edges in a graph can either be *undirected* or *directed* (shown in Figure 3), meaning that they go from one node (the *tail*) to another node (the *head*). We define a *subgraph*, $G'$, of $G$ as $G' = ((V', E'|V' \subseteq V \text{ and } E' \subseteq E))$. Furthermore, we define an *induced subgraph* as the subgraph that arises from a subset of edges, $E''$, and their respective heads

[3]In Euler's case, the seven bridges of Königsberg.
[4]A term coined by James Joseph Sylvester in 1878 (Biggs et al. (1986)).

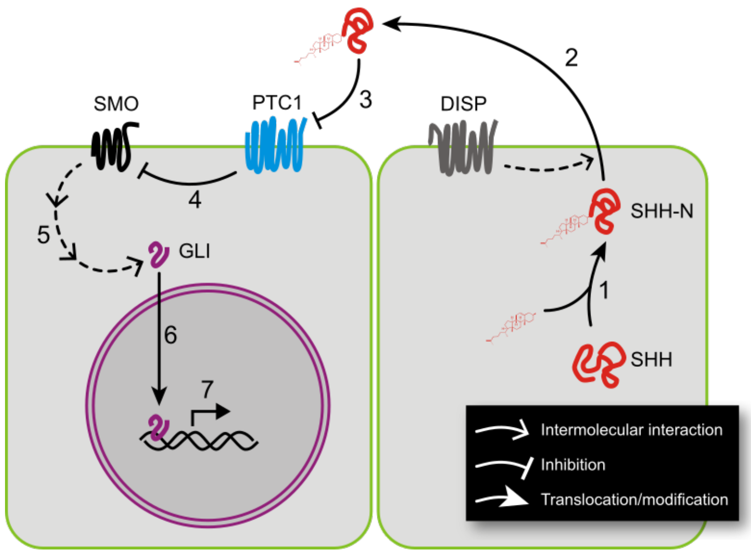


Figure 2: A traditional representation of the Sonic hedgehog signaling pathway. By Fred the Oysteri. The source code of this SVG is valid. This vector graphics image was created with Adobe Illustrator., GFDL, `https://commons.wikimedia.org/w/index.php?curid=36313869`

and tails.

One of the greatest advantages of using graphs as modeling tools is the wealth of graph algorithms that have been invented since Euler first proposed problems on graphs in 1736. These algorithms make it simple for graphs that are representative of real biological data to be queried for informative (i.e. shortest path between two nodes, modularity of a graph, or subgraphs that satisfy certain parameters) which can themselves be analyzed to determine whether they represent a biologically meaningful phenomena.

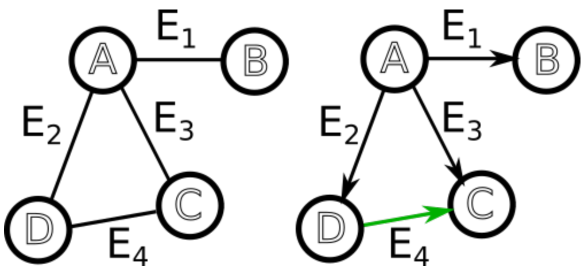


Figure 3: A simple undirected graph (left) and it's directed counterpart (right). The green edge has node $D$ as its tail and $C$ as its head.

One way that we often reason with graphs is through weighting. When we weight a graph, we generally assign values to the edges (though sometimes the nodes as well) to indicate differences between the edges, other than which nodes they connect. Weights can represent many different conditions, and often help apply real-world meaning to a graph that is being used as a modeling tool. For instance, if one had a graph where the nodes were representative of locations, then the weights of the edges could be used to show the distance between the nodes. One could also use a graph to represent a group of lab-mates who play streetball together during the summer, with weighted edges representing the likelihood that any two people will play on the same team. In the case of this thesis, we will use a combination of edge and node weighting to build subgraphs that may be biologically interesting using gene expression data.

### 0.2.1 Walks, Paths, Connectivity, and Trees

In graphs, it is useful to define a way to traverse the graph, through the nodes and edges. We define a *walk* in a graph as an alternating series of nodes and edges, such that each edge included in the walk is adjacent to the two nodes incident upon it. An example of one such walk, $W$ in the undirected graph from Figure 3 is:

$$W = (A, E_2, D, E_2, A, E_1, B)$$

Furthermore, we can define a *directed walk* in a directed graph as a walk that respects the direction of the edges in the graph. Additionally, we define a *path* (or a *directed path*, in directed graphs) as a walk for which there are no repeated vertices or edges. If we want to talk about a path $P$ that goes from one node, $s$ to another $t$, we use the notation $P(s, t)$ to describe the start and endpoints of the path. Furthermore, we say that a graph is *connected* if, for any pair of nodes in the graph, there exists a path between those two nodes. It is important to note that the endpoints of both paths and walks are always nodes, not edges. For example, in the directed graph shown in Figure 3, one possible path from $A$ to $C$, is:

$$P(A, C) = (A, E_2, D, E_4, C)$$

Using our definition of a path, we can also define a *cycle* as a path for which the start and end nodes are the same. Using this definition, we can also say that any path or graph that does not contain any cycles as an *acyclic* path or graph. Finally, we can define a *tree* as any graph that is connected and acyclic. This means that for any pair of nodes in a tree there exists exactly one path between them.

### 0.2.2 Spanning Trees and Steiner Trees

It is often useful to construct subgraphs from a given graph that fulfill specific parameters. These subgraphs are often formulated as a decision question: "Given a graph $G$ and a set of parameters, does subgraph $G'$ of $G$ exists that meets all of the specified parameters." An example of one such graph is a complete graph, shown in Figure 4. A complete graph of $n$ nodes ($k$-$n$) is defined as an undirected graph for which each node is connected to each other node by exactly one edge.

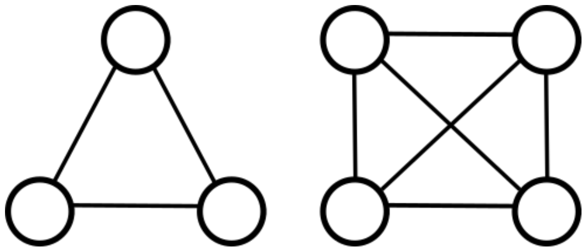


Figure 4: The complete graphs with three and four nodes.

**Spanning Trees.** One useful structure that we can look for in a graph is a *spanning tree*. Given a graph $G = (V, E)$, we say that there exists a spannning tree if there exits some tree $\mathcal{T} = (V', E')$ for which $V' = V$. More plainly, given a graph, a spanning tree is one that "hits" every node in the graph while still conforming to all the constraints that define a tree.

**Steiner Trees.** While spanning trees are useful substructures, sometimes we want to find a subtree that only spans a subset of the nodes in a graph. If this is the case, we can construct a slightly altered version of a spanning tree called a *Steiner Tree*[5] We say that a Steiner Tree $S$ exists in a graph $G$, if there is a subtree within the graph that connects all of a set of nodes $T$, which we call target nodes. Furthermore, we call any node that is included in the Steiner Tree, but that is not a member of $T$ a Steiner Node. Figure 5 gives an example of both spanning and Steiner Trees.

## 0.3 Optimization

It is often the case that, instead of just knowing *if* a certain Steiner Tree exists in a graph, we want to know what that Steiner Tree is. Furthermore, if there are multiple

[5]Named after Swiss mathematician Jakob Steiner.

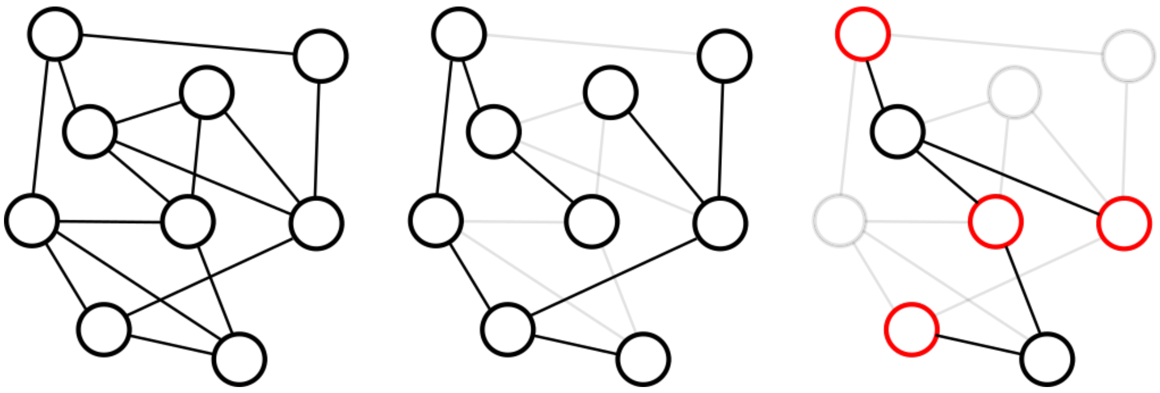
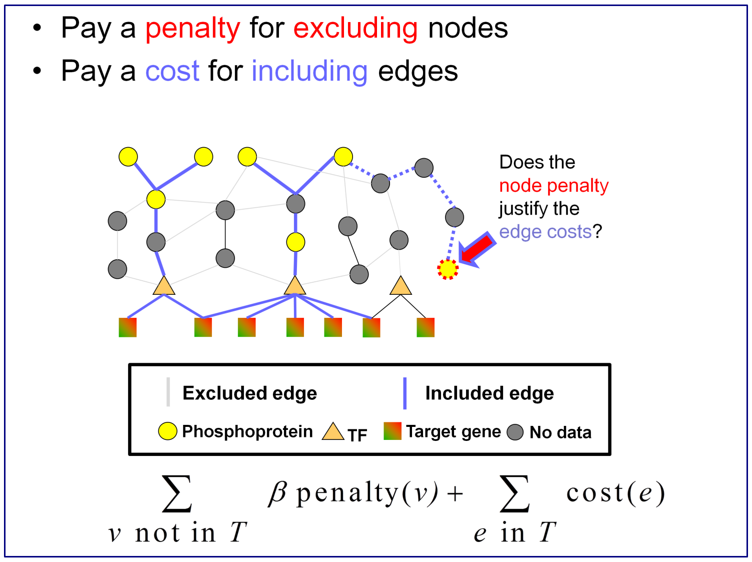


Figure 5: A graph *(left)* along with a spanning tree *(center)* and a Steiner Tree *(right)* within. In the Steiner Tree, the red nodes are target nodes and the black nodes are the Steiner Nodes.

possible Steiner Trees, we may want to find one (or more) that fulfills some notion of optimality. Most often, we want to find the Steiner Tree from a set of candidate Steiner Trees that will be minimal. In the case where the parent graph is unweighted, this will mean that the minimal Steiner Tree is the one that contains the fewest number of edges. Similarly, in a weighted graph, we say that the minimal Steiner Tree is the for which the sum of it's edge weights is smaller than the sum of the edge weights of all other candidate Steiner Trees.

### 0.3.1 Prize Collecting Steiner Trees

We often do not just want to find a Steiner Tree that explains a fixed target set, but rather want to loosen the constraints to let the exact nodes that are included in a subnetwork be determined algorithmically. This is particularly useful when we are examining biological signaling pathways, where we often have continuous data for a large subset of nodes because of high-throughput transcriptomics and proteomics. In this case we can assign values to some of the nodes in a graph based on available data. These so called "prizes" are used in lieu of a target set, to quantify how valuable a given node is, if it is included in the subnetwork. Using these prizes, and some scheme of edge weighting, we can find a subnetwork that captures the maximum amount of prize nodes, while minimizing the total edge costs (Figure 6). We call this subgraph a *Prize-Collecting Steiner Tree.* We use this structure as motivation for how to formulate a generalization, which we will call a *Prize-Rich Hypershrub* (PRHS), later on.



- Pay a penalty for excluding nodes
- Pay a cost for including edges

$$\sum_{v \text{ not in } T} \beta \, \text{penalty}(v) + \sum_{e \text{ in } T} \text{cost}(e)$$

Figure 6: In PPI networks, we want to find the smallest possible subgraph that captures as much prize as possible. One way to find this is by searching for a Prize-Collecting Steiner Tree. Image (c) Copyright by Ernest Fraenkel 2016, reprinted by permission.

## 0.4 Integer Linear Programming

For this thesis, we use a type of programming known as *integer linear programming.* Integer linear programs (ILPs) solve for the optimal solution(s) to an objective expression, subject to a set of linear constraints (Papadimitriou & Steiglitz (1998)). In our case, the set of variables, $\mathbf{x}$, are all binary variables, which will represent whether a given part of the graph is included in the solution. If variable $\mathbf{x}_i$ is equal to 0 in the solution, the part of the graph with which it is associated will not be present in the optimal solution. Similarly, if $\mathbf{x}_i$ is equal to 1, its associated graph component is present in the optimal solution.

The canonical form of an ILP is:

$$\begin{aligned} \text{Maximize:} \quad & \mathbf{c}^{\mathbf{T}}\mathbf{x} && (1) \\ \text{Subject to:} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, && (2) \\ & \mathbf{x} \geq 0, && (3) \\ \text{and} \quad & \mathbf{x} \in \mathbb{Z}^n && (4) \end{aligned}$$

In this form $\mathbf{x}$ is the vector of binary variables for which the program is solving,

**c** and **b** are vectors and **A** is a matrix, all of which have integers as values. For this thesis IBM ILOG CPLEX Optimization Studio (CPLEX) was used to optimize input ILPs which were input in the form of LP files (`.lp`, see Appendix B).

## 0.5 Goals of this Thesis

The goal of this thesis is to generate an algorithm that will create subnetworks in a generalization of a graph called a *hypergraph*[6]. The sub-hypergraph generated by this algorithm should use weights on the edges and nodes of the hypergraph to find a subnetwork that will minimize the total cost of edges while maximizing the value of nodes. Furthermore, the output should not contain disconnected subgraphs, and it should have some sort of "flow" going from an undetermined (i.e. not pre-specified) set of hypernodes, $S$ to another set of hypernodes, $T$. In addition to formally defining and solving for this target subnetwork, this thesis will find a way to build a network from publically available signaling pathway data, and weight that network in a meaningful way using publically available data.

Ultimately, this should create a software package that anyone could use to build hypergraphs from available data, and to query that hypergraph for potential hypotheses about signaling pathway disregulations, based on a weighting scheme that the user can apply easily.

[6]This will be defined both informally and formally later.

# Chapter 1

# Cell Signaling Networks

Cell function is governed by countless interactions between proteins, nucleic acids, lipids, carbohydrates, and many other small molecules. The interactions between all of these components form what we call *protein-protein interaction* (PPI) networks, which are responsible for almost every process within a cell (Taylor et al. (2009)). Furthermore, we often think about specific sub-networks that explain important biological phenomena (hormone secretion, transcriptional regulation of a gene, or response to a particular stimuli are all examples). We refer to these subsets of the larger PPI network as *cell signaling pathways.* We use cell signaling pathways to describe how many of the most basic reactions within cells cause propagations of information, and the results of thse signals. Some of the most important types of interactions that we see in cell signaling networks include the assembly and destruction of protein complexes, how small molecules such as ATP interact with proteins, the cascade of events that can occur after a membrane-bound protein is bound by a ligand, or where negative feedback loops exist that can have an effect on cell function. Historically, PPI networks have been a useful tool for compiling knowledge about individual interactions that have been studied *in situ*, so that larger scale patterns of interaction can be examined, and both communicated easily and analyzed algorithmically. In recent years, there has been a push to find ways to accurately model these networks so that they can be used to predict potential areas of future research (Haverty et al. (2004)), in particular, by using graph-based methods (Aittokallio & Schwikowski (2006)).

There are a multitude of different forms of cell signaling pathways[1], responsible for many types of cell functions, but we will use the example of a cell changing its gene expression in response to an external stimuli as a general model for signaling

[1]As well as extracellular signaling networks and metabolic cell functions that are not triggered externally.

pathways. In these cases, there are a few key steps that result in signal transduction throughout the cell. First, a trans-membrane protein will undergo a conformational change in response to a particular ligand[2]. Typically, this signal will be some sort of messenger molecule to which the cell needs to respond. The change in conformation results in a cascade of interactions between proteins and other small molecules within the cell. Some of the changes that can result from this cascade of signaling are protein complex assembly, complex degradation, conformational changes to other proteins, phosphorylation, or dephosphorylation (among many other reactions). The ultimate result of this signaling cascade is a change in one or more transcription factors, proteins that bind to DNA, and regulate the rate of transcription of DNA to mRNA. The ultimate result of this pathway will be a change in the expression of one or more genes within the cell, and possibly the start of new signaling pathways.

**Computational Methods.** Signaling pathways are of particular importance because they are often disregulated in heterogenous diseases such as cancer (Taylor et al. (2009)). Because of this, researchers' ability to obtain accurate information about signaling pathways, and their ability to interpret that information may help them to deduce interactions whose disregulation may be implicated in a particular disease. As these networks grow and become more accurate, we will become increasingly able to develop computational methods to mine these networks for novel areas to research, that may help explain or treat heterogenous diseases. Additionally, as the databases become more complete, cross-pathway interactions may begin to be documented that would not otherwise be apparent upon studying individual pathways. The development of new computational techniques to elucidate regulatory changes in areas where different pathways, once modeled as discrete, crosstalk. If areas like these, could be identified, they could lead to new hypotheses that researchers could investigate *in situ*. Furthermore, while past research has found many individual proteins and pathways that are implicated in particular diseases, our ability to observe or quantify certain elements that could be playing an important role in pathway disregulation is limited by our current sampling methods (RNA-Seq, microarray, etc). The development of more nuanced algorithms could increase our ability to implicate proteins that we cannot currently observe being disregulated because of thow protein expression in cels is currenly quantified.

Beyond individual signal transduction pathways, it is useful to think of the set of

[2]A small molecule that typically will bind to a trans-membrant, initiating a cascade of events (i.e. the signal that starts the process).

all known interactions, together. We refer to this object as an *interactome*, and our ability to analyze it computationally could lead to discovery (or at least postulation) of completely novel signaling networks. By examining the interactome for a species, and weighting it appropriately, it may be the case that we are able to elucidate sets of interactions that had been observed separately, but actually to be correlated with each other. A simple case of this would be if we find that one signaling pathway was to lead directly into another, that is, the outputs of the first pathway were the inputs to the other. Though interactions between multiple networks as simple as that are likely to have already been observed, it could be the case that an algorithm could find that there is some "chain" of connections between one network and another, or that there was some form of crosstalk between some of the components of two networks. If we could find this type of interaction, it would help researchers understand the extent to which a change in one network may affect another.

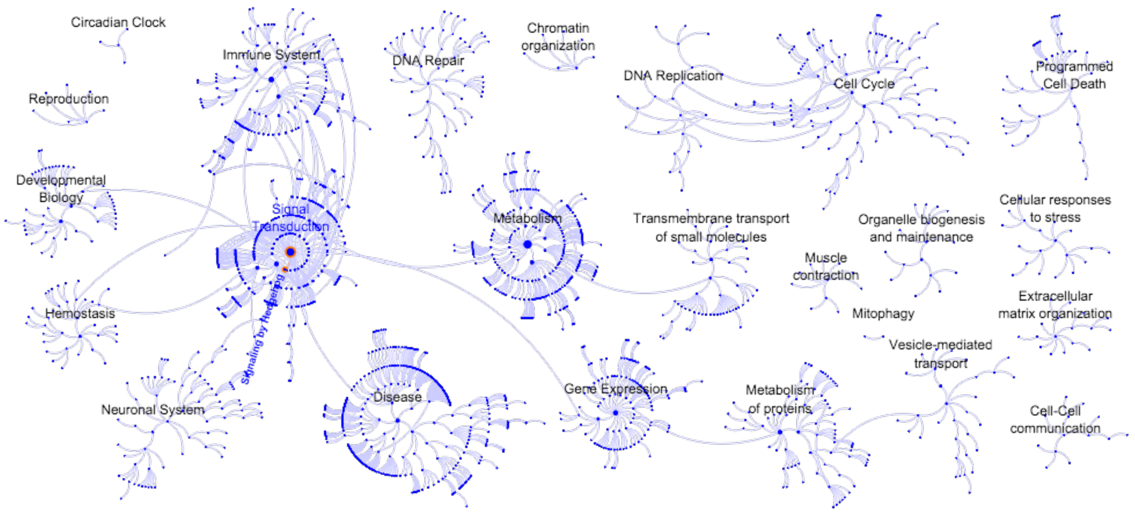


Figure 1.1: The entire human interactome, as displayed by the Reacome interactive pathway browser.

## 1.1 PPI Databases

In recent years, there has been a push to start curating what is already known about cell signaling, and publishing these networks online (Bauer-Mehren et al. (2009), Cusick et al. (2009)). Many of these databases have been made available to the public, so that the data that they contain can be used collaboratively by anyone.

Some of these networks are the Reactome database (`reactome.org`; Matthews et al. (2009)), NetPath (`netpath.org`; Kandasamy et al. (2010)), and SPIKE (`http://spike.cs.tau.ac.il/spike2/`; Paz et al. (2011)).

The purpose of signaling pathway databases is twofold: to create repositories of known interactions, so that they can be easily referred to and viewed in a zoomable, searchable manner (Hu et al. (2007)), and so that researchers can take advantage of the computational tools that alerady exist to find novel areas of research from these manually-curated networks (Karlebach & Shamir (2008), Battle et al. (2010)). These more modern pathway interaction databases are much smaller than their earlier, non-curated counterpart, but contain much higher confidence interactions, making them better candidates for applying computational methods.

## 1.2 Signaling by *Hedgehog*: A Case Study

One example of a cell signaling pathway that has a variety of biological consequences is the *Hedgehog* (Hh) signal transduction network. Hedgehog is a protein that helps regulate limb formation during early development, cell development, differentiation, and the development of neural tubes (Hui & Angers (2011)). Hedgehog has been implicated in the development of basal cell carcinoma[3] (BCC) when it is overexpressed (**?**). Additionally, it has been shown that Hh has powerful effects on the proper layout and development of tissues in mammals, and it has been proposed that Hh is responsible for assisting with stem cell assisted tissue regeneration in adult mammals (**?**).

Put in 2-3 ¶here about how Hh signaling works in more detail, and why it is an interesting pathway to study.

[To do.]

The Hedgehog signaling pathway makes a useful example for applying computational methods and graph-based modeling, since the network is small enough to look at manually, but also complex enough that its analysis is nontrivial. Furthermore, because it has been implicated with BCC, along with many other forms of cancer, data for weighting signaling networks are available through public databases such as The Cancer Genome Atlas (`http://cancergenome.nih.gov/`). These factors make it an excellent choice as an example network for testing new hypergraph algorithms.

[3]Basal cell carcinoma is thought to be the most common form of cancer in humans**?**. Indeed the author of this thesis was diagnosed with and treated for BCC.

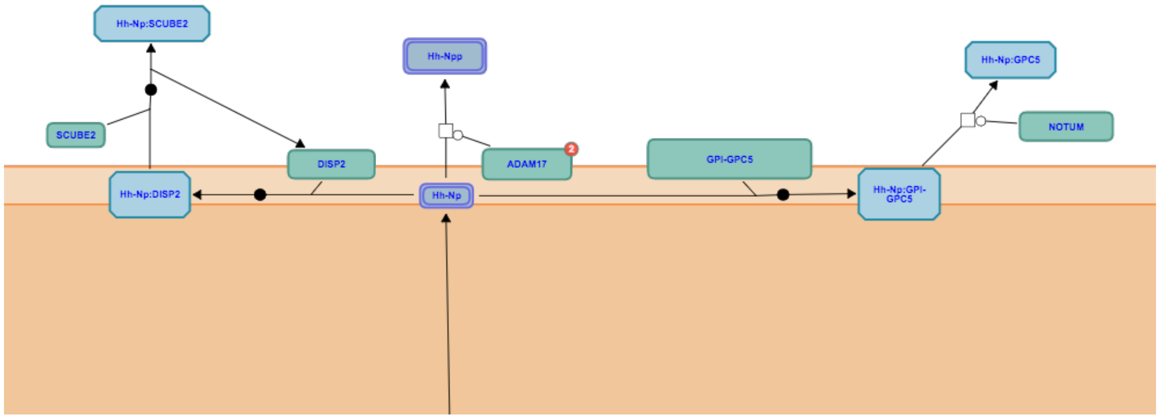


Figure 1.2: Hh-Np secretion, as shown by `reactome.org` in *Homo sapiens*.

## 1.3 Signaling Networks in *Arabidopsis thaliana*

One of the greatest strengths of computational methods is that they are generally agnostic to where their inputs come from, making them good tools for studying more than just human systems. In addition to humans, there is a wealth of data available for other species that can also be studied using PPI database mining. One such species is *Arabidopsis thaliana* (mouse-ear cress), a plant in the mustard family that is commonly used as a model for plant genetics. Researchers often use *A. thaliana* to examine the effects of genetic manipulation on plants, since *A. thaliana* has one of the best known genomes of any plant[4]. Because of this, there is a huge quantity of data available that make implementing the same kind of algorithms that are possible in human signaling pathways in *A. thaliana*.

Unfortunately, for many small-scale pathways (such as the Hedgehog pathway), there is not as much pathway data available in pathway databases, since much more research has been done on humans than on any other organism. Fortunately, however, there is enough pathway data available that the overall interactome can be searched and may yield useful outcomes.

Our ability to look at the entire interactome of species such as *A. thaliana* allows us

[4]It is like the *Drosophila melanogaster* of the plant world.

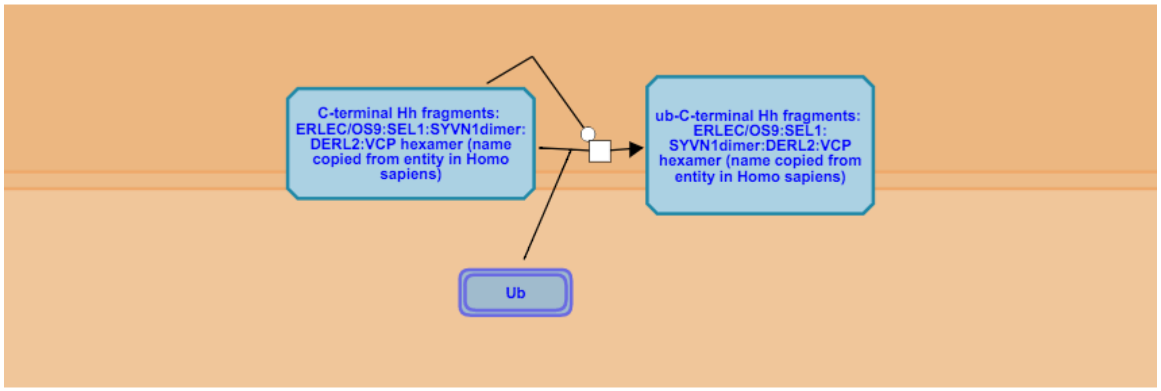

C-terminal Hh fragments: ERLEC/OS9:SEL1:SYVN1dimer: DERL2:VCP hexamer (name copied from entity in Homo sapiens)
ub-C-terminal Hh fragments: ERLEC/OS9:SEL1: SYVN1dimer:DERL2:VCP hexamer (name copied from entity in Homo sapiens)
Ub

Figure 1.3: The Reactome Hedgehog signaling pathway in *A. thaliana*. For this pathway, and many others, the Reacome database is very sparse relative to the orthologous pathways in humans, and is therefore not very useful for implementing network algorithms, at this point.

to investigate some of the global effects of genetic manipulation on signal transduction. This is especially useful in model organisms like *A. thaliana*, since we can actually perform genetic manipulations on them quickly, safely, and ethically, unlike with human beings. Additionally, our ability to breed model organisms in the lab means that we can use specific datasets with large sample sizes. This is very useful compared to human datasets, which are generally based on small sample sizes, since human gene expression data is generally collected from pseudo-random, sparse cases when a person has a degenerate tissue type, such as a tumor, rather than under direct experimental manipulation.

Now that we have an understanding of what signaling pathways and protein interaction networks are, we need to develop an appropriate tool with which we can model them. To do so, we will now describe a generalization of a graph, known as a hypergraph. We will then describe a new way to query hypergraphs that will find important subnetworks, which we call hypershrubs, that may reveal biologically interesting phenomena that current computational methods fail to capture.

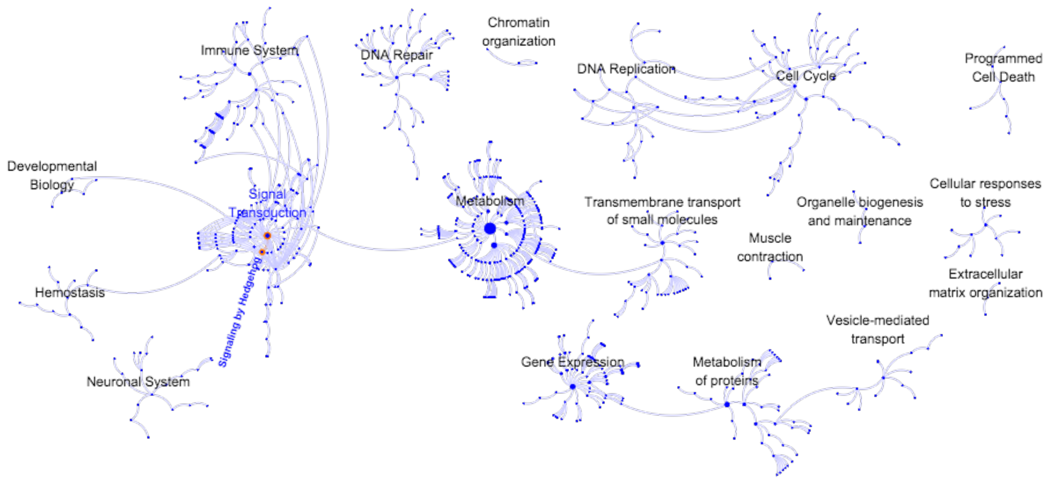


Figure 1.4: The entire *A. thaliana* interactome, as displayed by the Reacome interactive pathway browser. This interactome is comparably dense to the human interactome seen in 1.1, which makes it a better candidate for running network algorithms.

# Chapter 2

# Hypergraphs & Hyperpaths

In order to model cell signaling pathways computationally, it is first necessary to choose the correct computational object to represent our data. For this, we can use graphs, but there are some ways in which graphs are not the ideal structure, and leave some things to be desired (see Figure 2.1). Instead of using graphs, we will define a generalization of a directed graph, called a directed hypergraph, and use it to model cell signaling pathways and PPI networks.

## 2.1 Graphs and their Limitations

While standard[1] graphs are useful for many applications, they are severely limited in their ability to represent cell-signaling interactions. Since they can only show pairwise interactions between nodes, whenever there is an interaction that requires more than two connections, the visualization of the graph becomes confusing, and biologically meaningless. Furthermore, interactions involving multiple molecules require an enormous amount of different edges to represent all of the sub-interactions that take place. Determining whether two edges are part of the same biological event is a non-trivial problem, and requires manual curation to solve, at this time.

One area of cell-signaling that becomes particularly problematic in standard graphs is the formation, interaction, and destruction of protein complexes. The easiest way that complexes can be represented in standard graphs is by creating a complete subgraph of all of the elements of the protein complex. On their own, these complete subgraphs can yield useful information about the make-up of a protein complex, but once they begin interacting with other elements of the graph, the graph becomes

[1]The term "standard" is used to describe traditional graphs, since the terms "regular" and "normal," both refer to specific types of graphs.

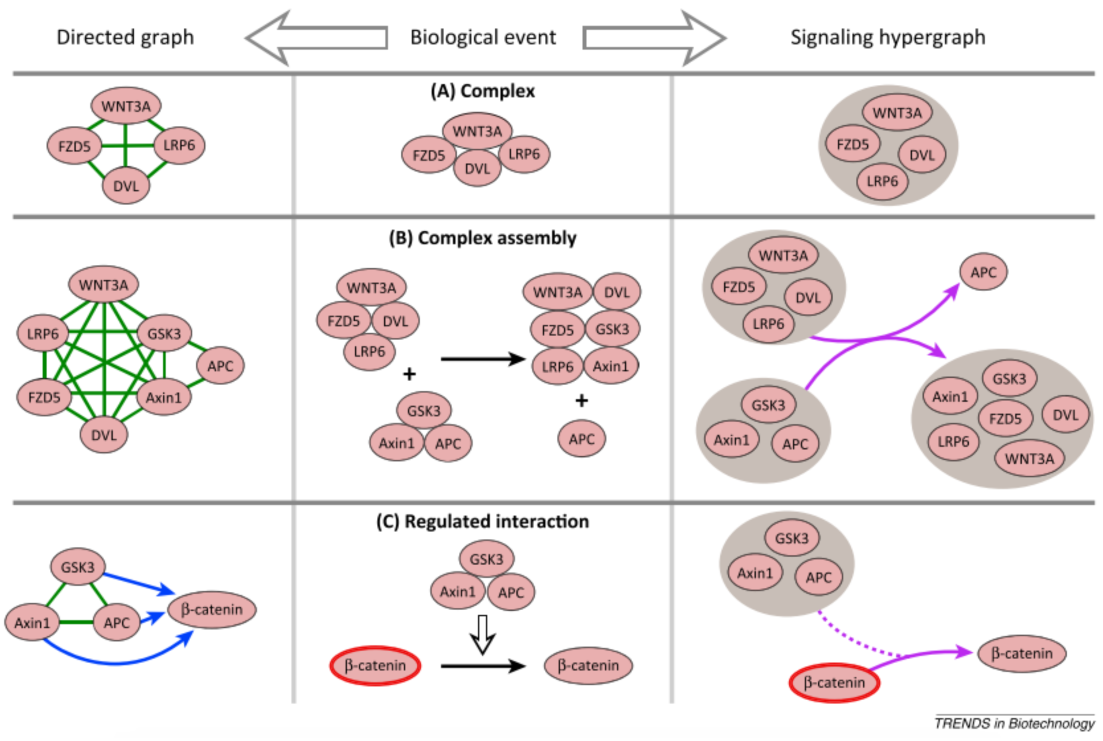


Figure 2.1: Standard graphs are not always the best tool for modeling biological events. Instead, we use hypergraphs, a generalization of standard graphs that allow for more nuanced modeling.

much more complex, as all of the proteins in the complex must be represented independently. In the case of interactions between multiple large complexes, it becomes the case that the standard graph representation of this interaction is a large complete subgraph that contains nodes for all of the proteins involved in either complex. It is then computationally impossible to distinguish whether the entity being described by the subgraph is the interaction between multiple protein complexes, or simply one large complex that contains all of the components of both complexes. Furthermore, since the complete subgraphs which represent complexes are undirected, it is extremely difficult to tell what the inputs or outputs of a biochemical reaction may be.

Another shortcoming of standard graphs in representing complex biological interactions is that there is no way in which to represent positive or negative regulation of interactions. Since there is a standardized way in which edges interact with nodes, there is no way to differentiate types of interactions between nodes. This poses a challenge when there are regulators or catalysts present that are necessary for a reaction, but are not part of the inputs or outputs of the reaction. If regulators are

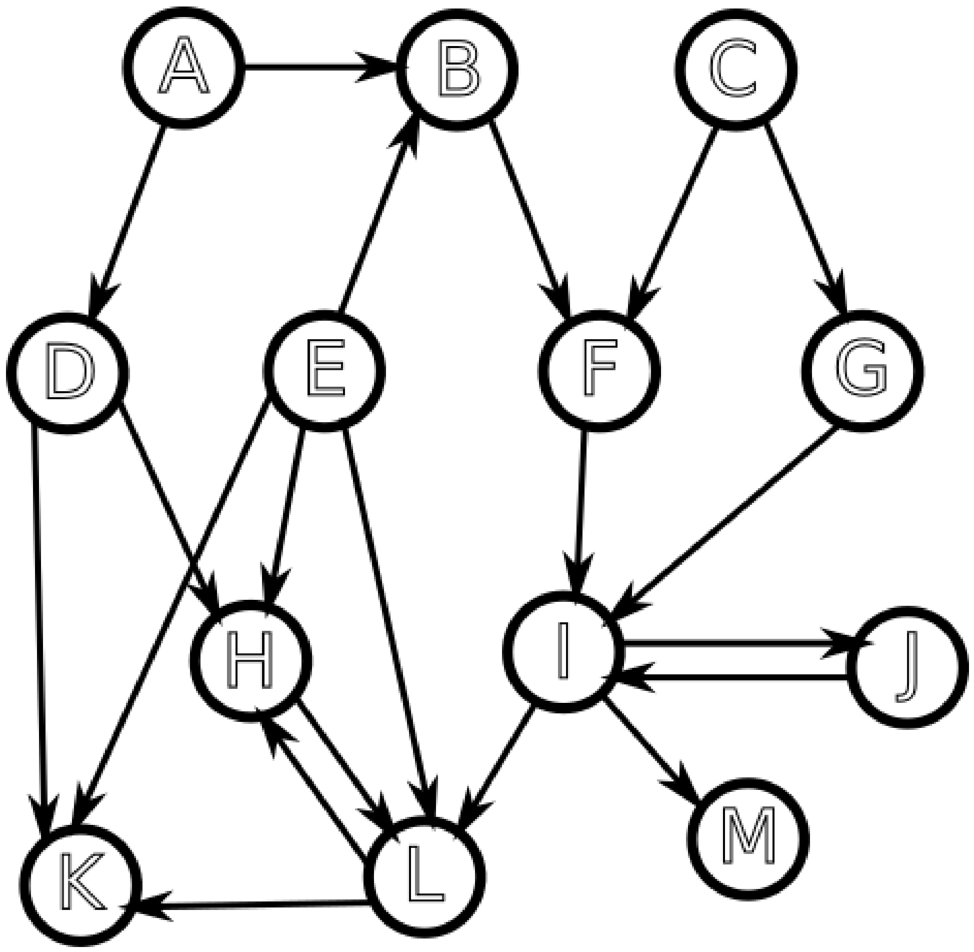


Figure 2.2: An example of a standard graph. All edges represent directed, pairwise interactions between two nodes. Biologically, it is difficult to extract meaningful signaling pathway information from this graph, since protein complex formation and degradation is ambiguous.

to be included in a standard graph representation of a cell-signaling network, they become indistinguishable from any other types of interactions that are taking place. This lack of specificity is problematic, as it treats all interactions as equal, and hides potentially useful information from the graph[2].

To resolve the issues presented by standard graphs, we instead use a generalization called a *hypergraph* that allows for the addition of more detail and specificity within the data structure than standard graphs allow. In particular, hypergraphs allow for both the representation of protein complexes in the form of *hypernodes*, which we think of simply as a set of one or more nodes, and for the representation of complex, directed interactions that can have multiple inputs and outputs. We represent these interactions with the use of *hyperedges*, which define a set of one or more hypernodes. Since a hyperedge may include more than two hypernodes, we gain the ability to represent both multi-protein interactions, as well as to define the notions of regulation on reactions.

## 2.2 Hypergraphs

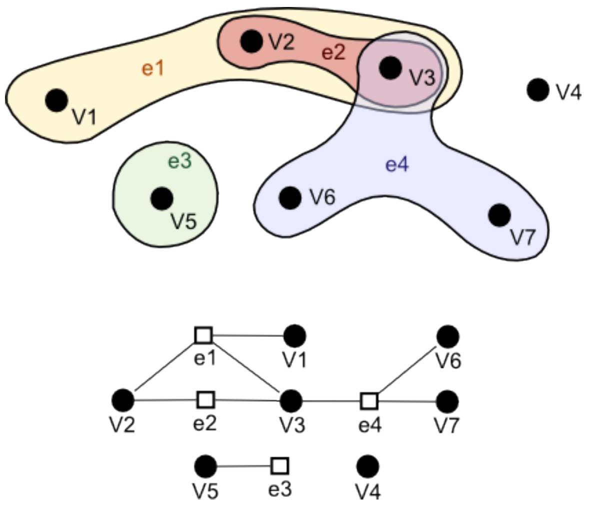


Figure 2.3: An example of an undirected hypergraph (SHARPEN (2012)), where each edge is shown by a colored region, as well as the conversion of the hypergraph to a standard graph.

Where a directed graph represents directed, pairwise interactions between only two

[2]The Reacome database is available for download as a standard graph, but it is not available as a directed graph. This means that we could not use their graphs as a standard graph comparison to ours.

vertices, we can use *directed hypergraph* to represent directed interactions between sets of vertices (nodes). We formally define a directed hypergraph, $\mathcal{H}$, as a pair $(V, E)$, where $V$ is a finite set of vertices and $E \subseteq 2^V \times 2^V$ is a finite set of *directed hyperedges* connecting members of $V$ such that, for every $e = (T(e), H(e)) \in E$, $T(e) \cap H(e) = \emptyset$, and $T(e)$, $H(e) \neq \emptyset$ (definition adapted from Gallo et al. (1993)). We refer to $T(e)$ as the *tail* of the hyperedge, and to $H(e)$ as the *head* of the hyperedge.

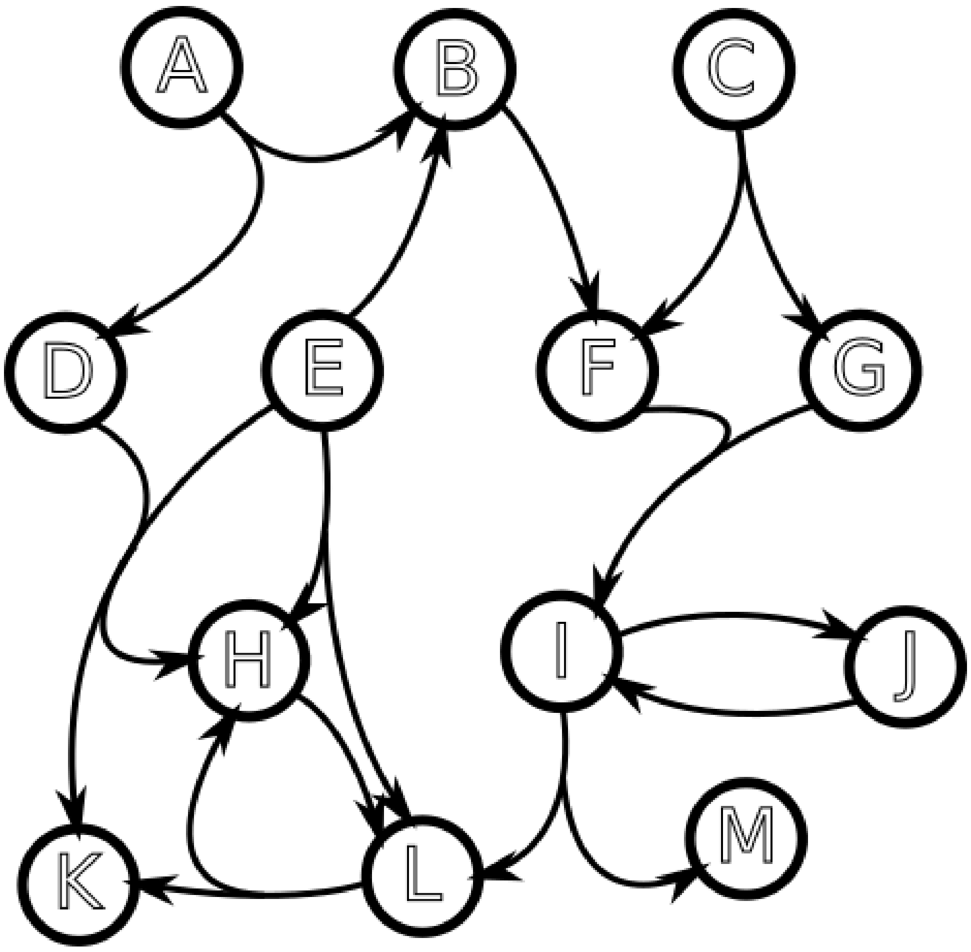


Figure 2.4: An example of the hypergraph, $\mathcal{H} = (V, E)$ that corresponds to the standard graph shown in Figure 2.2. Note that each edge now contains more specific information about the propogation of a signal through the network. For this example, $V = A, B, ..., M$ and each hyperedge is shown by a compund arrow, for example the edge whose tail is hypernodenode $A$ and whose tail is hypernodes $B$ and $D$.

It is important to note that a standard directed graph is a special case of a directed hypergraph. This is the case if every hypernode in the graph contains only one element, and if each edge has exactly one head element, and one tail element. This has two important implications for algorithms that run on directed hypergraphs.

First, this means that there is no loss in functionality caused by using a hypergraph representation of a cell network, since anything that could be computed on a standard directed graph can be recreated exactly using the special case of the hypergraph. Secondly, this is important, because it means that anything that can be computed on a standard graph will be at least as computationally difficult to compute when generalized to a hypergraph. In fact, we find that many tasks that are computationally easy on standard graphs become very difficult when generalized to hypergraphs (Ritz et al. (2014b)).

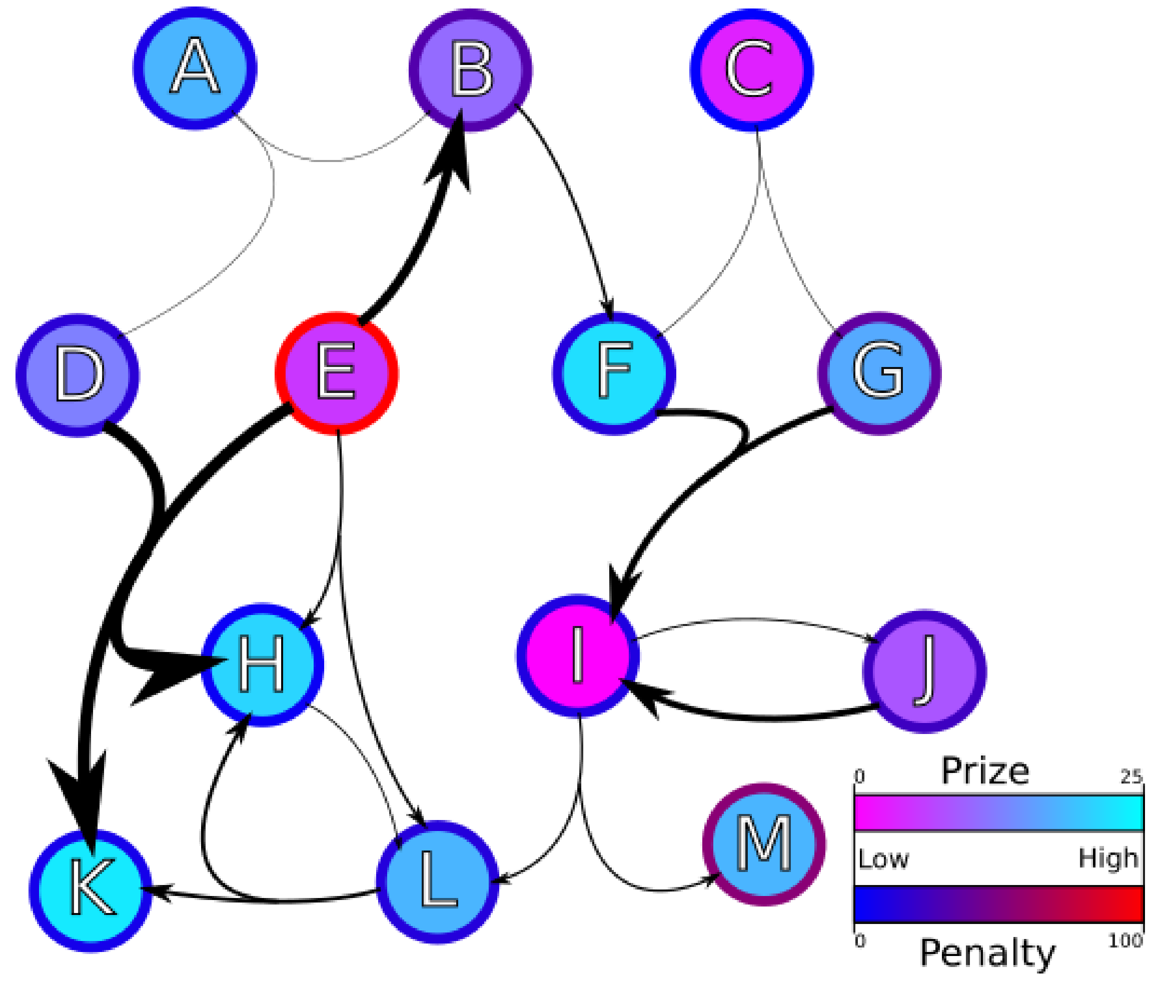


Figure 2.5: The same hypergraph as in 2.4, now weighted with node prizes (inner color) and dangling penalties (outer color). Edge thickness corresponds with edge weight. Intuitively, to find the PRHS for this hypergraph, we want to find the subnetwork that maximizes the amount of blue, and minimizes the amount of both red nodes and thick lines.

## 2.3 Hyperpaths & Connectivity

In order to find the shortest route between two vertices, in a directed hypergraph, we define the notion of a *hyperpath*, $\mathcal{P}$, on the directed hypergraph $\mathcal{H}$. We think of a path, $\mathcal{P}$ from vertex $s$ to vertex $t$ as the list of vertices and hyperedges that one must pass through in order to traverse through the directed hypergraph from vertex $s$ to $t$. The existence of a hyperpath between two nodes encodes the notion of "connectivity" between those nodes. If a hyperpath exists between nodes $s$ and $t$, we say that they are *connected*. This is a useful definition, because it allows us the notion of a *connected hypergraph*, a hypergraph in which every node is connected by some hyperpath to every other node in the hypergraph.

### 2.3.1 Directed Hyperpaths

There are many ways in which we can define hyperpaths, but for the purposes of finding Steiner Trees in directed hypergraphs, we must define a simple *directed hyperpath* between nodes $s$ and $t$. We can think of a directed hyperpath $\mathcal{P}(s,t) = \{s, e_2, v_3, ..., e_{n-1}, t\}$ as an ordered list of nodes and edges, beginning with $s$ and ending with $t$ such that for any edge, $e_i$, in the path, $v_{i-1}$ is in the tail of $e_i$, and $v_{i+1}$ is in the head of $e_i$.

There are two types of hyperpaths that we find particularly interesting, as a result of the branching nature of hyperedges. The first of these is a path the branches out from a single back-end node (Figure **??**). We call this type of hyperpath that is defined in terms of it's back end a B-hyperpath (Gallo et al. (1993)). B-hyperpaths are a particularly interesting to biologists modeling signaling pathways, since a B-hyperpath represents all the products that can be reached from a given reactant (Ritz et al. (2014a)).

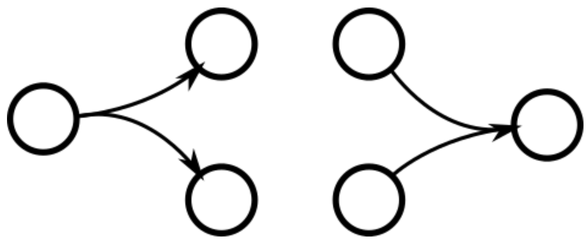


Figure 2.6: *left* A very simple example of a B-hyperpath. *right* A simple example of an F-hyperpath.

The second type of hyperpath that we are interested in, F-hyperpaths, are very similar to B-hyperpaths, except that they are defined by a node on their front end (Figure **??**), and are made of hyperedges that lead into that edge. Biologically, this represents all of the reactions that can lead to a given product. Both of these types of hyperpaths will be formalized later, and they will be generalized as part of our definition of a hypershrub.

# Chapter 3

# Hypershrubs

Recall that we are interested in finding interesting signaling pathways in protein-protein interaction networks using hypergraph representations of the PPI networks. Remember as well that we want to use biological data to find subnetworks that capture as much of the data in as small as a subnetwork as possible (*this should probably be reworded.*), much like in the Prize-Collecting Steiner problem. Finally, recall that with problems such as this, we see two formulations, as an unweighted decision problem, and as a weighted optimization formulation.

Now, we will review the formulations of B and F hyperpaths, which are both concerned with single source-target pairs. From there, we will generalize these hyperpaths to include multiple targets. Our ultimate goal is to present two new formulations, one a decision problem for hypergraphs seeking a set of targets from multiple sources, and the second an optimization problem for hypergraphs that relaxes the decision formulation. This will allow us to seek worthy targets and sources. Finally, we will show an ILP formulation that will produce this optimal subnetwork, which we will call a *hypershrub*.

**Hyperwalks.** In order build our formalization of B and F hyperpaths, we begin by defining *hyperwalks*, an analog to a walk in a standard graph. We define a hyperwalk from $s$ to $t$ as a sequence of edges $\mathcal{W} \subseteq E$

$$(e_1, e_2, ..., e_\ell)$$

of length $\ell$, where $T(e_i) \cap H(e_{i+1}) \neq \emptyset$[1] and $s \in H(e_1)$ and $t \in T(e_\ell)$. In other

[1] This ensures that there is at least one shared vertex between $e_i$ and $e_{i+1}$

words, a hyperwalk is a series of “hops” that one can make along directed hyperedges that take you from $s$ to $t$.

**B-connectivity.** For B-connectivity relative to a source, $s$, we consider a tree[2] of hyperedges emanating forwards from $s$, with the condition that there be no “loose tails,” such as in Figure 3.1. More precisely, we define the set of *outs* from $s$ in terms of a series of $\ell$ outward hops $\mathcal{O}_\ell(s)$ where:

$$\begin{aligned} \mathcal{O}_0(s) &= \{s\} \\ \mathcal{O}_{i+1}(s) &= \{u \mid \forall\, e, v \text{ where } u \in T(e) \text{ and } v \in \mathcal{O}_i(s) \,\forall\, v \in H(e)\} \end{aligned}$$

We can now say that $s$ is *B-connected* to some node $t$ if $t \in \mathcal{O}_h(s)$ for some $h \geq 0$. In other words, this means that $t$ is $h$ hops forward from $s$, and there is some walk from $s$ to $t$ for which there are no loose tails.

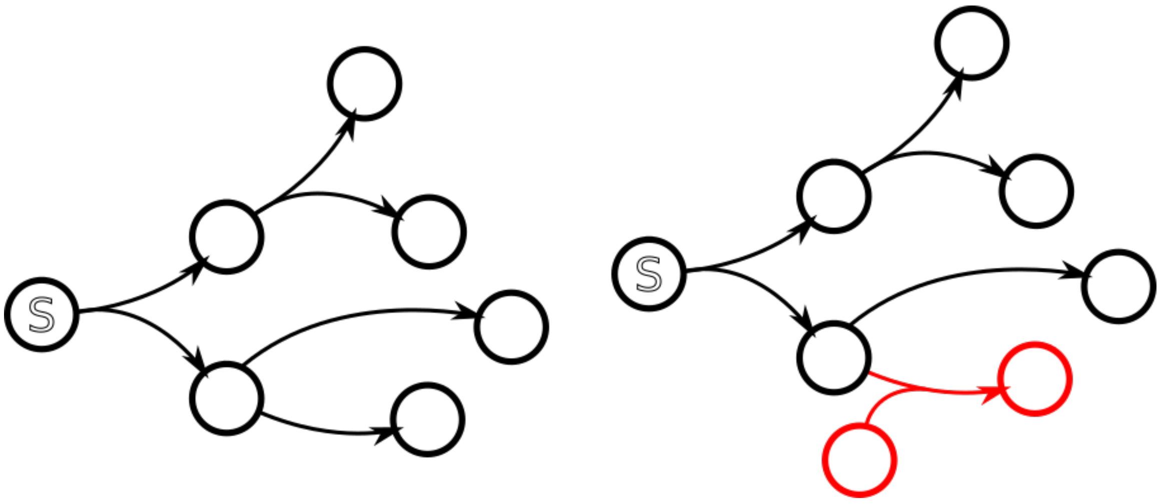


Figure 3.1: *left* A hypergraph $s$ is B-connected to all nodes, when a loose head is added *right* $s$ is no longer B-connected to the head of that edge (shown in red).

**F-connectedness.** Similarly to B-connectedness, we can consider the set of nodes that “funnel” into a given target node, $t$ without any loose heads. We therefore can use the same logic to define the *ins* to $t$ as a series of $\ell$ inward hops $\mathcal{I}_\ell(t)$. Similarly, we can say that a node $s$ is F-connected to $t$ if $s \in \mathcal{I}_h(t)$ for some $h \geq 0$.

[2]Note that this is not to be confused with the mathematical structure.

**B-Hypershrubs, F-Hypershrubs, & BF-Hypershrubs.** In our formulation of B and F-connectedness, we were focusing on the vertices that reach from or to a target. Instead, we might consider the edges that participate in $\mathcal{O}(s)$ or $\mathcal{I}(t)$. Consider the following, where $A \subseteq E$:

$$H(A) = \bigcup_{e \in A} H(e) \tag{3.1}$$

and

$$T(A) = \bigcup_{e \in A} T(e) \tag{3.2}$$

From this, we can also define the induced subset of nodes $V(A) = H(A) \cup T(A)$. From this, we also define the set of everything that is in the tail of some edge, but not in the heads of any hyperedges in $A$ as $D(A) = V(A) \setminus H(A)$. The set $D(A)$ represents the set of the induced roots from $A$. Similarly, we find the induced leaves from $A$ by defining $C(A) = V(A) \setminus T(A)$. From these definitions, we can construct three definitions of Hypershrubs, *B-Hypershrubs*, *F-Hypershrubs*, and *BF-Hypershrubs*. Intuitively, we think of a B-Hypershrub which strictly respects the backward set of nodes, an F-Hypershrub as one that strictly respects the forward set of nodes, and a BF-Hypershrub as one that strictly respects both sets of nodes. The names for these three types of hypershrubs is adapted from the naming convention for different types of hyperpaths (Gallo et al. (1993)). For all three of the hypershrubs, we begin with a hypergraph, $\mathcal{S}$, and two sets of nodes: a source set, $S$, and a sink set, $T$, where $S, T \subseteq V$. We now formulate three decision problems:

**B-Hypershrub.** Given a hypergraph $\mathcal{H}$ and node sets $S$ and $T$ we say that there is a B-Hypershrub, $\mathcal{B}$ if there exists some subset of hyperedges $A$ such that:

$$C(A) \subseteq T$$

and

$$D(A) = S$$

We call the induced hypergraph of $A$ the B-Hypershrub, $\mathcal{B}$.

**F-Hypershrub.** Given a hypergraph $\mathcal{H}$ and node sets $S$ and $T$ we say that there is a F-Hypershrub, $\mathcal{F}$ if there exists some subset of hyperedges $A$ such that:

$$C(A) = T$$

and

$$D(A) \subseteq S$$

We call the induced hypergraph of $A$ the F-Hypershrub, $\mathcal{F}$.

**BF-Hypershrub.** Given a hypergraph $\mathcal{H}$ and node sets $S$ and $T$ we say that there is a BF-Hypershrub, $\mathcal{BF}$ if there exists some subset of hyperedges $A$ such that:

$$C(A) = T$$

and

$$D(A) = S$$

We call the induced hypergraph of $A$ the BF-Hypershrub, $\mathcal{BF}$.

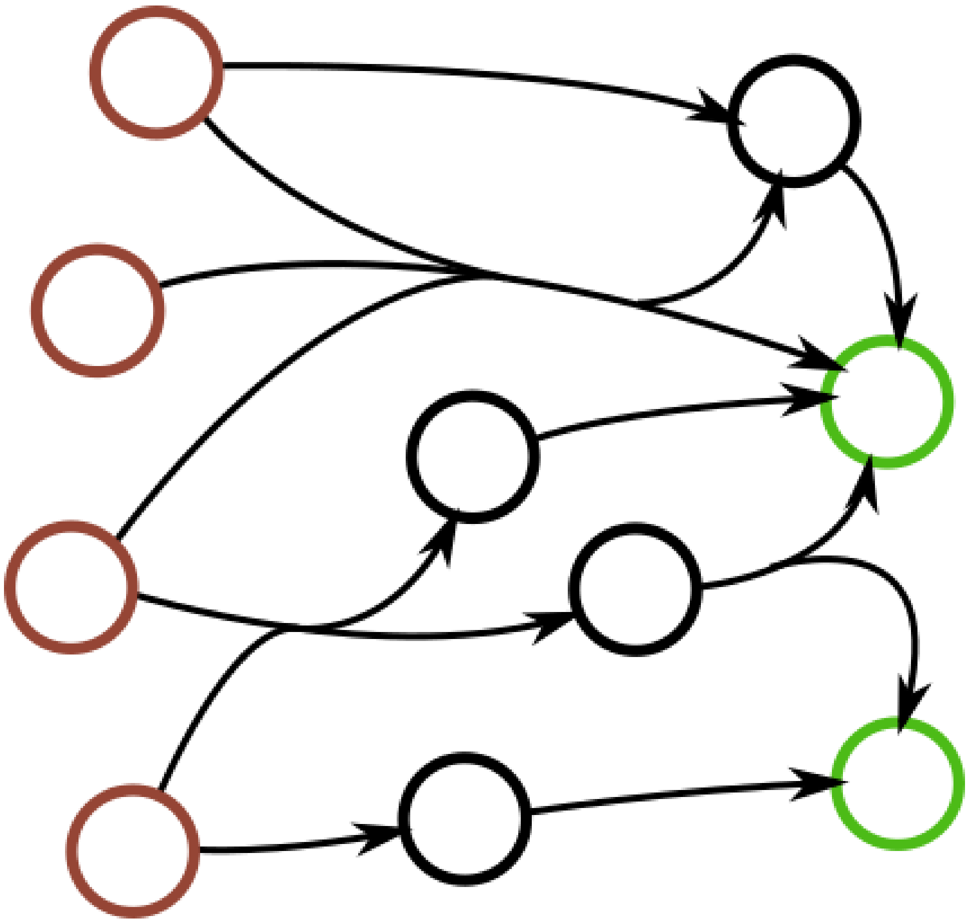


Figure 3.2: A Hypershrub, with edge set $A$. The roots, $D(A)$ are shown in chestnut and the leaves, $C(A)$ in kelly green. All of the black nodes represent $H(A) \cap T(A)$.

One important piece to note is that we can easily reconstruct B, F, or BF-hyperpaths from $s$ to $t$ from their respective hypershrub generalizations by making $S = \{s\}$ and $T = \{t\}$. Furthermore, because we encode hypershrubs through the heads and tails of the included hyperedges, we implicitly know that there is always at least one hyperwalk from each $s \in S$ to some $t \in H(A)$, and from some $s \in T(A)$ to each $t \in T$ (*Do I need to prove this formally?*). Finally, though this formulation does encode a hypershrub that is connected from its roots to its leaves, it does not necessarily encode a completely connected sub-hypernetwork, since these constraints could create two mutually exclusive hypershrubs from only one $S$ and one $T$.

**Optimization.** While all three formulations of the decision problem for a Prize-Rich Hypershrub may have useful applications, we will focus on the optimization and weighted versions of the BF-Hypershrub. From the decision question, "Does a BF-Hypershrub $\mathcal{S}$ exist in $\mathcal{H}$ given some $S$ and $T$?" we can extend to looking for a minimal BF-Hypershrub. We say that a BF-Hypershrub $\hat{\mathcal{S}}$ is minimal in a hypergraph $\mathcal{H}$ given $S$ and $T$ if there are no other possible BF-Hypershrubs in $H$, for $S$ and $T$ that have fewer hyperedges than $\hat{\mathcal{S}}$.

## 3.1 Definition of PRHS

When we are looking for a minimal BF-Hypershrub using empirical data, we often do not know what $S$ and $T$ are *a priori*. To get around this, we instead try to determine a prize-collecting variant of the decision and optimization problem described above, so that we can determine a plausible set of roots and leaves that will define some BF-Hypershrub in a weighted hypergraph. We now can define the Prize-Rich Hypershrub, $\mathcal{S} = (V', E')$, to be the set of hypernodes and hyperedges for which total vertex prizes are maximized, and total edge weights are minimized. We also try to minimize the cost that we give to each node for being present without any incoming hyperedge[3].

To make our Prize-Rich Hypershrub, $\mathcal{S}$, we begin with a parent hypergraph, $\mathcal{H} = (V, E)$, and a set of target nodes, $T$, such that $T \subseteq V$. We refer to each node as some $v$ such that $v \in V$, and each edge is some $e$ such that $e \in E$. When building $\mathcal{H}$, each hyperedge is assigned a weight, that is, a cost associated with including that edge in the solution hypergraph, $\mathcal{S}$. We then assign each hypernode in $\mathcal{H}$ two values, a prize for being included in $\mathcal{S}$, as well as a penalty associated with being a *root* hypernode.

[3]In signaling pathways, this cost quantifies how likely or unlikely we consider it to be that a given reactant is present in the cell, agnostic to the reactions that caused it to be there.

## 3.2 Implementation as an Integer Linear Program

To start our implementation, we begin by looking backwards through a hypergraph from a set of target nodes, $T$. We begin in this way because, in PPI data, downstream nodes are often more easily inferred using empirical data such as gene expression, since changes in the expression of some genes is the ultimate output of cell signaling pathways.

Given an input hypergraph $\mathcal{H} = (V, E)$, and a set of target nodes, $T$, we construct a hypershrub, $\mathcal{S} = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$. We build $\mathcal{S}$ using an ILP which encodes the definition of a hypershrub. In order to accomplish this, we define three indicator variables, $\alpha_v$, $\alpha_e$, and $\delta_v$, where $v$ and $e$ are hypernodes or hyperedges in $\mathcal{H}$. If hypernode $x$ is in the solution of the ILP, $\alpha_v$ will have a value of 1, otherwise it will be equal to 0. Similarly, if edge $e$ is present in the solution, $\alpha_e$ will take a value of 1. The value of $\delta_v$ will be determined by whether a hypernode is dangling.

We find the hypershrub $\mathcal{S}$ by optimizing the function:

$$\underset{\alpha,\delta}{\operatorname{argmax}} \sum_{v \in V} g_v \alpha_v - \sum_{e \in E} c_e \alpha_e - \sum_{v \in V} h_v \delta_v \tag{3.3}$$

Subject to the set of linear constraints:

$$\alpha_v \geq 1 \qquad \forall\, v \in T \tag{3.4}$$

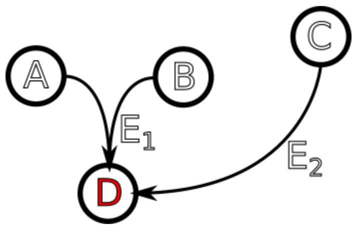
$$\sum_{v \in H(e)} \alpha_v \geq |H(e)| \alpha_e \qquad \forall\, e \in E \tag{3.5}$$

$$\sum_{v \in T(e)} \alpha_v \geq |T(e)| \alpha_e \qquad \forall\, e \in E \tag{3.6}$$

$$\delta_v \leq \alpha_v \qquad \forall\, v \in V \tag{3.7}$$

$$\delta_v \geq \alpha_v - \sum_{e : v \in H(e)} \alpha_e \qquad \forall\, v \in V \tag{3.8}$$

Here, the objective function (3.3) tries combinations of nodes ($\alpha_v$), edges ($\alpha_e$), and dangling nodes ($\delta_v$) that maximize the sums of node prizes ($g_v$), minimize the sums of edge costs ($c_e$), and minimize the sums of dangling penalties ($h_v$). Constraint (3.4) encodes that every target node in $T$ is in the solution, $\mathcal{S}$. Constraints (3.5) and (3.6) ensure that if an edge is in $\mathcal{S}$, any nodes incident on that edge (i.e. in the head or tail of that edge) will also be in $\mathcal{S}$. Finally, constraints (3.7) and (3.8) encode the ability for nodes to dangle if they do not have an incoming edge. To illustrate what each of these constraints does in more detail, we will construct a simple example, and examine each constraint within that example.



| Nodes | | |
|---|---|---|
| Node | $g_v$ | $h_v$ |
| $A$ | 10 | 1 |
| $B$ | 10 | 1 |
| $C$ | 1 | 10 |
| $D$ | 10 | 10 |

| Edges | |
|---|---|
| Edge | $c_e$ |
| $E_1$ | 1 |
| $E_2$ | 10 |

Figure 3.3: A simple hypergraph, and its associated node and edge weights.

**Example.** Consider the hypergraph shown in Figure 3.3. For this hypergraph, the PRHS that comes from this graph should consist of hypernodes $A$, $B$, and $D$, as well as hyperedge $E_1$. This is obvious, since nodes $A$ and $B$, which will be dangling in the solution, both have low dangling penalties and high prizes, while node $C$ has a very large dangling penalty relative to its prize. Similarly, $E_1$ has a relatively low cost, whereas $E_2$ has a very high cost. Finally, $D$ is considered a target, in this case, and is therefore automatically included in the solution. Knowing what the solution should be, we can use this hypergraph to demonstrate the effect of each linear constraint in our ILP.

Table 3.1: All of the possible solutions that satisfy all necessary linear constraints, and their corresponding objective values. We see that the induced subgraph from $E_1$ yields the optimal solution.

| Subgraphs | | |
|---|---|---|
| $V$ | $E$ | Objective Value |
| $A, B, D$ | $E_1$ | 27 |
| $C, D$ | $E_2$ | -9 |
| $A, B, C, D$ | $E_1, E_2$ | -2 |

We now know that, for the optimal PRHS, $\alpha_A$, $\alpha_B$, $\alpha_D$, and $\alpha_{E_1}$ are all equal to 1 (that is, they are included in the solution), and $\alpha_C$ and $\alpha_{E_2}$ are equal 0. Additionally, we know that in this solution $A$ and $B$ are dangling nodes, therefore $\delta_A$ and $\delta_B$ are both 1, and $\delta_C$ and $\delta_D$ are both 0. Finally, we are given that $D$ is a target node, hence $T = \{D\}$.

First let us look at the objective function, Formula (3.3). We can call each of the three sums included the equation which calculate the total node prizes, edge costs, and dangling penalties $\Xi$, $\Phi$, and $\Psi$, respectively. Given the values of $\alpha_v$, $\alpha_e$, and $\delta_v$

that we know should yield the PRHS, we get find:

$$\begin{gathered}\Xi = 30 \\ \Phi = 1 \\ \Psi = 2 \\ \implies \Xi - \Phi - \Psi = 27\end{gathered}$$

If $C$ or $E_2$ were to be added, or any node or edge were removed from the PRHS, the total value of the objective function would decrease, therefore yielding a suboptimal solution to the ILP.

Now, we can begin to look at how each linear constraint governs the behavior of the ILP. Let's begin with constraint (3.4). For our hypergraph, $T$ only has one element, $D$, constraint (3.4) only needs to be checked for one node. We know that $\alpha_D = 1$, therefore constraint (3.4) simplifies to:

$$\alpha_D \geq 1$$

We see that constraint (3.4) holds for all $v$ in $T$, therefore we know that all members of the target set are included in the PRHS.

Next, we can look at constraints (3.5) and (3.6). Since these constraints are the same, except for whether they are concerned with the head or tail of a hyperedge, we can evaluate the effect of only constraint (3.5), and assume that constraint (3.6) works in the same way. To assess this constraint, we must see how if the inequality holds for all edges in the hypergraph. We begin by looking at $E_1$. We know that $H(E_1) = D$, and that $|H(e)| = 1$, therefore we can check:

$$\alpha_D \geq 1 \bullet \alpha_{E_1}$$

So, we see that constraint (3.5) holds for $E_1$. Now, we can check for $E_2$ (whose head is also only $D$):

$$\alpha_D \geq 1 \bullet \alpha E_2$$

We see that this inequality also holds. This means that, since the inequality holds for all hyperedges, that for every edge in the hypergraph, its head is also included in the hypergraph. We can easily extend this to see that (3.6) enforces the same for the tails of every hyperedge.

Finally, we can look at the three constraints that allow hypernodes to be dangling: constraints (3.7) and (3.8). First, we look at constraint (3.7):

$$\delta_A \leq \alpha_A \qquad \delta_B \leq \alpha_B \qquad \delta_C \leq \alpha_C \qquad \delta_D \leq \alpha_D$$

Now that we see that constraint (3.7) holds for all nodes in the PRHS, we can look at 3.8. This constraint is trivial for every node in the hypergraph other than $D$, since it is the only node in the head of any hyperedges:

$$\delta_D \geq \alpha_D - \alpha_{E_1} - \alpha_{E_2}$$

## 3.3 Roots and Leaves Problem

When we implement the ILP in a more complex hypergraph, such as the hypergraph shown in Figure 2.5, we find that it creates a hypergraph that consists of multiple smaller hypergraphs (Figure 3.4), rather than a continuous subnetwork. Our original implementation ends up returning a group of disconnected hypergraphs that represent regions with high prize to edge/dangling cost ratios, separated by gaps where the parent hypergraph had regions of low prize to edge/dangling cost ratios. While this result is intersting in itself, it will not necessarily create a soltion that is biologically interesting, once we begin looking for PRHSs in pathway signaling data.

Our new goal is to find a way to connect regions that would be disconnected in the original formulation of the PRHS. In doing so, we have two cases that may exist: the region between sub-hypernetworks is prize-rich or it is prize-poor. If the region is prize-rich, we would want to connect the two regions throuh that region to construct a larger PRHS that explains the lowest cost connecting region. If the region is prize-poor, on the other hand, we would rather have the less valuable sub-hypernetwork to be left out of the final PRHS, since the cost of reaching it would not justify the prize that the sub-hypernetwork contained. Additionally, we need to consider the case where there is no way to get from one sub-hypernetwork to another

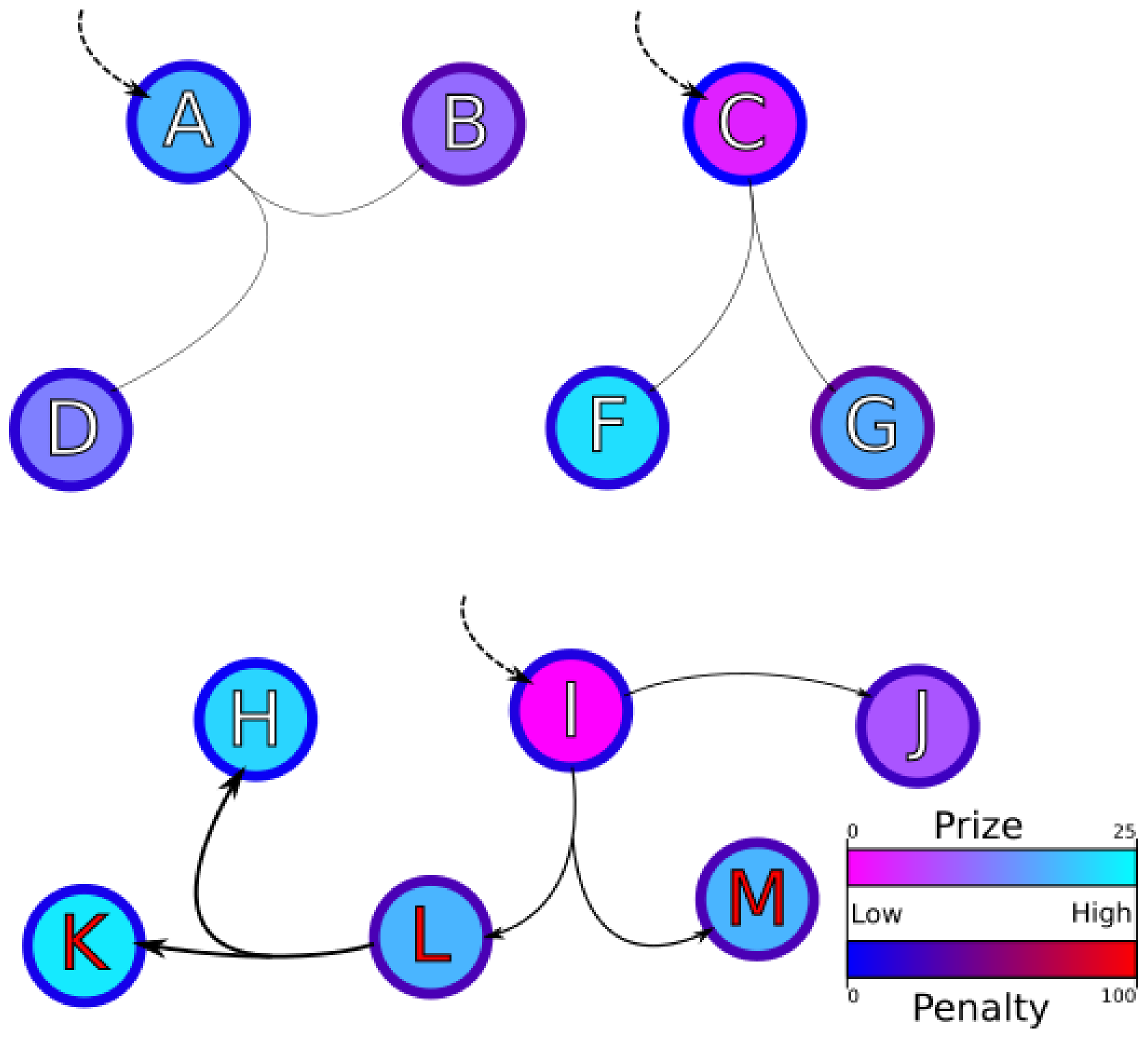


Figure 3.4: The output of our ILP, when it is run on the hypergraph shown in Figure 2.5. The dashed arrows indicate a dangling hypernode.

sub-hypernetwork, we will allow both to be present in the final solution.

To accomplish these goals, we use the set of nodes that either have no incoming or outgoing hyperedges in the solution: the roots and leaves. We are only concerned with these dangly nodes, and not with the inner nodes because we want to keep the same topology that is generated by the original formulation, and not be concerned with any hypereges that may connect the middle parts of sub-hypernetworks, but whose prices are too high to justify being included in the solution. When we think of this biologically, it means that we do not want to force including a reaction in the solution just because it's reactants and products are present elsewhere in the solution, if there are already lower cost explanations for why they are present.

To fix this new problem, we must incorporate the notion of hyperpaths and connectedness to our implementation of the PRHS, as well as define the class of leaf variables in our formulation. Additionally, we must construct a matrix of whether or not there exists a hyperpath between every pair of nodes in $\mathcal{H}$. As such, we adopt the notation $\iota_v$, as a binary variable that indicates whether hypernode $v$ is a leaf in the solution. Furthermore, we introduce the binary variable $\mathcal{P}(u, v)$ to indicate whether or not a hyperpath exists from $u$ to $v$. From this, we simply say that if there exists a hyperpath from node $s$ to $t$, then either $s$ cannot be a leaf or $t$ cannot be a root (i.e. $\mathcal{P}(u, v) \implies \neg(\delta_u \wedge \iota_v)$).

**Implementation in the ILP.** To add this change to the ILP for the PRHS it is necessary to implement three new linear constraints, as well as the two new sets of variables described already, *iota* and $\mathcal{P}$. It is important to note that, in a hypergraph with $n$ vertices, this will add $n$ new *iota* variables, and $n^2$ new $\mathcal{P}$ variables, since every pairwise combination of vertices $(s, t)$ must be checked to see if there is a hyperpath from $s$ to $t$.

We introduce linear constraints:

$$\iota_v \leq \alpha_v \qquad \forall\, v \in V \tag{3.9}$$

$$\iota_v \geq \alpha_v - \sum_{e: v \in T(e)} \alpha_e \qquad \forall\, v \in V \tag{3.10}$$

$$2 - \delta_u - \iota_v \leq \mathcal{P}(u, v) \qquad \forall\, u, v \in V \tag{3.11}$$

Similarly to in the original constraint (3.7), constraint (3.9) ensures that all nodes which are considered leaves must be present in the solution. Additionally, constraint

(3.10) follows the same form as constraint (3.8), only it now applies to edges for which the node of interest is in the tail, rather than the head. These two constraints enforce the labeling of nodes that do not have an outgoing hyperedge as leaves. Constraint (3.11) is where we encode that that if a path exits from $u$ to $v$, then either $u$ cannot be a leaf, or $v$ cannot be a root, in the PRHS. When we implement these three linear constraints, along with the existing set in the hypergraph from Figure 2.5, we find a new solution, as shown in Figure 3.5.

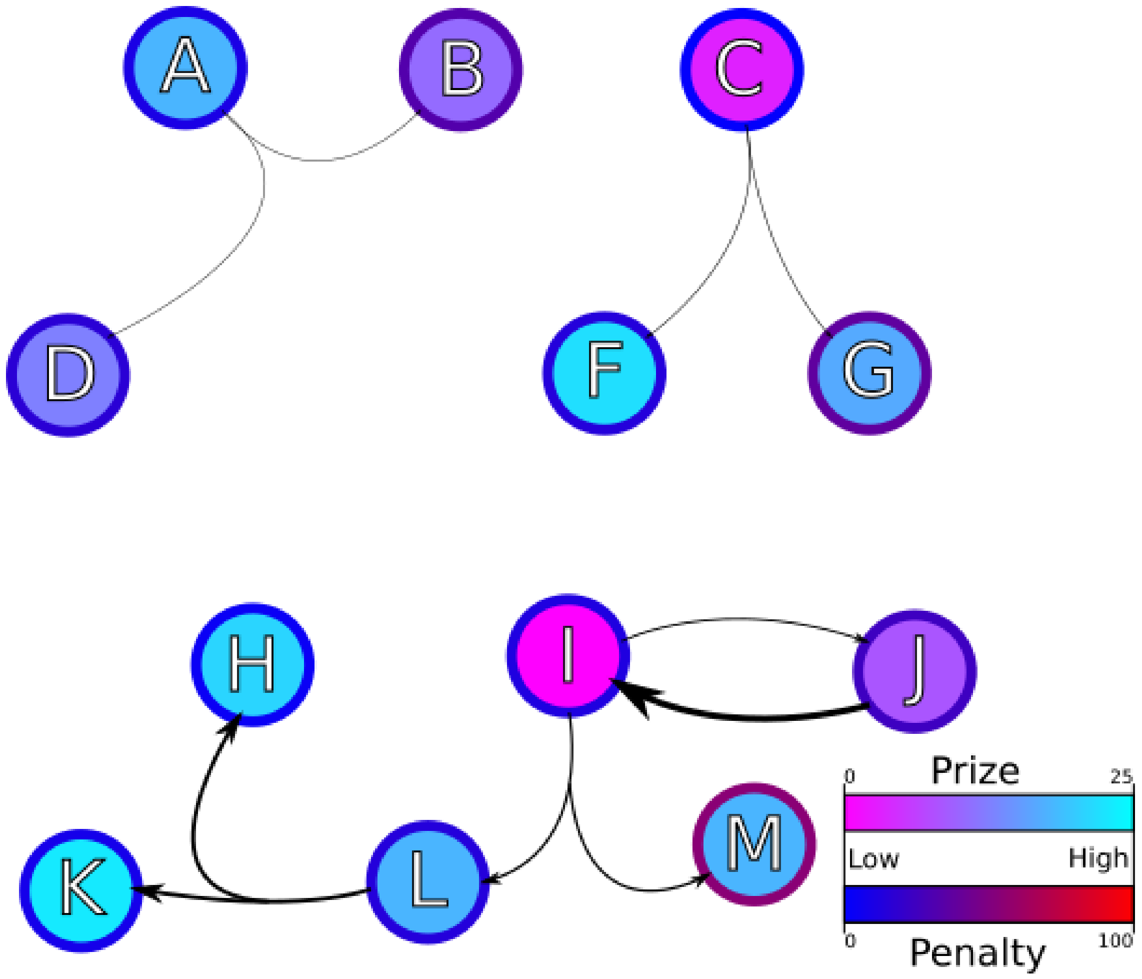


Figure 3.5: The output of our ILP, when it is run on the hypergraph shown in Figure 2.5, also taking the Roots and Leaves problem into account.

**Running Time.** It is important to note at this point that, because the number of variables and constraints in the ILP now quadratic with respect to the number of nodes in the original hypergraph, the amount of time that the ILP will take to solve will increase rapidly with the number of nodes. To this end, even the time to build the `.lp` files that are the input of the ILP solver is non-trivial on large datasets. This is illustrated in the table below.

Table 3.2: Four datasets of increasing size, and how long they take to be converted into *.lp* files using both the original formulation (*blp*) and the formulation that takes into account the roots and leaves problem (*blprl*). Input files are the dummy hypergraph used in 3.3, the example hypergraph from 2.5, the Hedgehog signaling pathway, and the entire Reactome database for *A. thaliana*. All times are in seconds.

| Hypergraph | Nodes | Edges | Time(*blp*) | Time(*blprl*) | Lines(*blp*) | Lines(*blprl*) |
|---|---|---|---|---|---|---|
| dummy | 4 | 2 | 0.00066 | 0.00095 | 30 | 86 |
| ex | 13 | 12 | 0.00115 | 0.00274 | 100 | 633 |
| Hh | 413 | 82 | 0.0183 | 1.464 | 1,904 | 514,437 |
| Reactome | do | do | do | do | do | do |

Figure out how to time CPLEX and make an analagous table here. [To do.]

# Chapter 4

# Methods and Results

## 4.1 Network Construction and Weighting

In order to construct directed signaling hypergraphs of biological pathways, we use pathway data from preexisting, curated protein-protein interaction networks, available publically. For this, pathway signaling information was downloaded from the curated Reactome pathway database (Croft et al. (2014), Milacic et al. (2012)), and parsed using a Java script (Ritz (2016)) into two flat files containing node and edge information necessary to construct the hypergraph (See Appendix B).

The networks generated from the Reactome database specify each hypernode as a combination of one or more proteins or small molecules. For simplicity, each hypernode was "converted" into a regular node for the purpose of algorithm implementation, and the components' information was saved so that they could be reconstructed *post hoc*. Additionally, each hyperedge contained information about whether the reaction that it represented was regulated either positively or negatively by one or more nodes. Positive regulators were added to the tail of their respective hypernodes, since they are assumed to be necessary reactants for the edge, and should therefore be considered part of the signaling network. Negative regulators, on the other hand, were excluded from the flat files, since the way that they act on reactions poses an interesting problem in modeling, since the current definition of PRHS assumes that all node prizes and edge weights are positive (see Future Directions).

From the edge files, hypergraphs were constructed using the Hypergraph Algorithms Package (HALP, Murali-group (2015)). Edge files gave enough information to create induced hypergraphs, which were then weighted based on information stored in node files. For the implementation PRHS to create meaninful subnetworks, it is important that the parent hypergraph have a meaningful, biologically informed weighting scheme applied. However, it is often the case that data are difficult to acquire and add to the hypergraph, so simple default settings were first applied to each parameter. By default, each node was given a prize of 1, and a dangling penalty of 5. Additionally, each edge was given a default weight of 1, so that the algorithm would try to minimize the number of edges.

## 4.2 Simulated Data

To assess the efficacy of this ILP in generating the correct Prize-Rich Hypershrub, it is first necessary to perform a series of benchmarking tests on known datasets and hypergraphs. This will allow us to determine if the algorithm is performing as expected, and generating results that can be corroborated manually.

### 4.2.1 Simple Example Hypergraphs and Potential Special Cases

Draw and describe:

- Standard graph
- Flower
- Complete graph
- Line graph
- Disconnected graph

[To do]

## 4.3 PRHS in Real Data

Intro. [To do.]

### 4.3.1 Human Cancer Data

Explain the dataset and how it was applied. Finish applying it. [To do.]

## 4.4 Comparison to Standard Graphs

Either do this section or move it to future directions.

*Make python file to convert to standard graphs!*

To test how well this algorithm is able to construct sub-hypergraphs that are more meaningful than their standard counterparts, we look at the same implementation on hypergraphs that have been converted to standard graphs. For this conversion, each hyperedge in the original hypergraph was replaced with a set of directed edges that corresponded to *FINISH THIS* ¶ []

# Conclusion

## 5.1 Future Directions

- Heat mapping to weight hypergraphs (Random walks).
- Computation of individual or multiple subnetworks.

## 5.2 Complexed Hypergraphs

We can define sets of two or more nodes as a hypernode, which are members of the power set of $V$. These can be incorporated into a directed hypergraph to form a *complexed directed hypergraph.* We define a *complexed directed hypergraph*, $\mathcal{H}$, as the triple $(V, U, E)$ in which $V$ is a finite set of vertices, $U \subseteq 2^V$ is a finite set of hypernodes, and $E \subseteq 2^U \times 2^U$ is a finite set of hyperedges such that, for every $e = (T(e), H(e)) \in E$, $T(e) \cap H(e) = \emptyset$ and $T(e)$, $H(e) \neq \emptyset$.

# Appendix A

# Benchmarking Graphs & Supplemental Data

# Appendix B

# Algorithms & Programs

**Construction of LP files.** The constructor `build_lp.py` builds a `.lp` file that can be optimized by the ILP solver CPLEX. The constructor takes the following files as arguments, as well as a column delimeter (default "`;`") and a node delimeter (default "`,`").

Edge file (`ex-edges.txt`):

---

```
tail;head;cost
A;B,D;4
B;F;14
C;F,G;5
D,E;H,K;57
E;H,L;14
F,G;I;33
I;J;9
J;I;31
H;L;8
I;L,M;12
L;H,K;19
E;B;45
```

---

Node file (`ex-nodes.txt`):

```
name;prize;penalty
A;7;10
B;14;32
C;21;1
D;12;16
E;19;98
F;3;14
G;8;37
H;4;4
I;24;12
J;16;24
K;2;inf
L;7;inf
M;7;inf
```

Output file (`ex.lp`):

```
Maximize
 obj: 2.0 Kx + 8.0 Gx + 16.0 Jx + 4.0 Hx + 7.0 Ax + 12.0 Dx + 7.0 Lx + 14.0
      Bx + 7.0 Mx + 3.0 Fx + 24.0 Ix + 21.0 Cx + 19.0 Ex - 0.0 Kd - 37.0 Gd
     - 24.0 Jd - 4.0 Hd - 10.0 Ad - 16.0 Dd - 0.0 Ld - 32.0 Bd - 0.0 Md -
     14.0 Fd - 12.0 Id - 1.0 Cd - 98.0 Ed - 31.0 hyperedge8 - 5.0 hyperedge3
      - 45.0 hyperedge12 - 33.0 hyperedge6 - 14.0 hyperedge2 - 4.0
     hyperedge1 - 12.0 hyperedge10 - 9.0 hyperedge7 - 14.0 hyperedge5 - 57.0
      hyperedge4 - 19.0 hyperedge11 - 8.0 hyperedge9
Subject To
 c44_hyperedge8: Jx - 1 hyperedge8 >= 0
 c43_hyperedge8: Ix - 1 hyperedge8 >= 0
 c44_hyperedge3: Cx - 1 hyperedge3 >= 0
 c43_hyperedge3: Fx + Gx - 2 hyperedge3 >= 0
 c44_hyperedge12: Ex - 1 hyperedge12 >= 0
 c43_hyperedge12: Bx - 1 hyperedge12 >= 0
 c44_hyperedge6: Fx + Gx - 2 hyperedge6 >= 0
 c43_hyperedge6: Ix - 1 hyperedge6 >= 0
 c44_hyperedge2: Bx - 1 hyperedge2 >= 0
 c43_hyperedge2: Fx - 1 hyperedge2 >= 0
 c44_hyperedge1: Ax - 1 hyperedge1 >= 0
 c43_hyperedge1: Dx + Bx - 2 hyperedge1 >= 0
 c44_hyperedge10: Ix - 1 hyperedge10 >= 0
 c43_hyperedge10: Lx + Mx - 2 hyperedge10 >= 0
 c44_hyperedge7: Ix - 1 hyperedge7 >= 0
 c43_hyperedge7: Jx - 1 hyperedge7 >= 0
 c44_hyperedge5: Ex - 1 hyperedge5 >= 0
 c43_hyperedge5: Hx + Lx - 2 hyperedge5 >= 0
 c44_hyperedge4: Dx + Ex - 2 hyperedge4 >= 0
 c43_hyperedge4: Hx + Kx - 2 hyperedge4 >= 0
 c44_hyperedge11: Lx - 1 hyperedge11 >= 0
 c43_hyperedge11: Hx + Kx - 2 hyperedge11 >= 0
 c44_hyperedge9: Hx - 1 hyperedge9 >= 0
 c43_hyperedge9: Lx - 1 hyperedge9 >= 0
 c45_A: Ad - Ax <= 0
 c45_B: Bd - Bx <= 0
 c45_C: Cd - Cx <= 0
 c45_D: Dd - Dx <= 0
 c45_E: Ed - Ex <= 0
 c45_F: Fd - Fx <= 0
 c45_G: Gd - Gx <= 0
 c45_H: Hd - Hx <= 0
 c45_I: Id - Ix <= 0
 c45_J: Jd - Jx <= 0
 c45_K: Kd - Kx <= 0
 c45_L: Ld - Lx <= 0
 c45_M: Md - Mx <= 0
 c46_K: Kd - Kx + hyperedge11 + hyperedge4 >= 0
 c46_G: Gd - Gx + hyperedge3 >= 0
 c46_J: Jd - Jx + hyperedge7 >= 0
 c46_H: Hd - Hx + hyperedge5 + hyperedge11 + hyperedge4 >= 0
 c46_A: Ad - Ax >= 0
 c46_D: Dd - Dx + hyperedge1 >= 0
 c46_L: Ld - Lx + hyperedge5 + hyperedge10 + hyperedge9 >= 0
 c46_B: Bd - Bx + hyperedge1 + hyperedge12 >= 0
 c46_M: Md - Mx + hyperedge10 >= 0
 c46_F: Fd - Fx + hyperedge2 + hyperedge3 >= 0
 c46_I: Id - Ix + hyperedge6 + hyperedge8 >= 0
 c46_C: Cd - Cx >= 0
 c46_E: Ed - Ex >= 0
Bounds
 Kx = 1
 Kd = 0
 Lx = 1
 Ld = 0
 Mx = 1
 Md = 0
Binary
 Ax
 Bx
 Cx
 Dx
 Ex
 Fx
 Gx
 Hx
 Ix
 Jx
 Kx
 Lx
 Mx
 Ad
 Bd
 Cd
 Dd
 Ed
 Fd
 Gd
 Hd
 Id
 Jd
 Kd
 Ld
 Md
 hyperedge1
 hyperedge10
 hyperedge11
 hyperedge12
 hyperedge2
 hyperedge3
 hyperedge4
 hyperedge5
 hyperedge6
 hyperedge7
 hyperedge8
 hyperedge9
End
```

# Appendix C

# Biological Data

# Appendix D

# Version Information

This appendix will be concerned with which versions of various softwares I used.

# References

Agrawal, A., Klein, P., & Ravi, R. (1995). When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem on Networks. *SIAM Journal on Computing*, *24*(3), 440–456.

Aittokallio, T., & Schwikowski, B. (2006). Graph-based methods for analysing networks in cell biology. *Briefings in Bioinformatics*, *7*(3), 243–255.

Aldridge, B. B., Burke, J. M., Lauffenburger, D. A., & Sorger, P. K. (2006). Physicochemical modelling of cell signalling pathways. *Nature cell biology*, *8*(11), 1195–1203. `/Users/yurikoharigaya/Documents/ReadCubeMedia/ncb1497.pdf$\delimiter"026E30F$nhttp://dx.doi.org/10.1038/ncb1497`

Anderson, C. N. K., Ramakrishnan, U., Chan, Y. L., & Hadly, E. A. (2005). Serial SimCoal: A population genetics model for data from multiple populations and points in time. *Bioinformatics*, *21*(8), 1733–1734.

Archer, A., Bateni, M. H., Hajiaghayi, M. T., & Karloff, H. (2009). Improved approximation algorithms for Prize-Collecting Steiner Tree and TSP. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, (pp. 427–436).

Ausiello, G., Franciosa, P. G., & Frigioni, D. (2001). Directed Hypergraphs: Problems, Algorithmic Results, and a Novel Decremental Approach. *Theoretical Computer Science, Vol. 2202 of Lecture Notes in Computer Science*, *2202*, 312–328. `http://link.springer.com/10.1007/3-540-45446-2`

Ausiello, G., Giaccio, R., Italiano, G. F., & Nanni, U. (1992). Optimal traversal of directed hypergraphs. *Operations Research*, (20244), 1–24. `http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Optimal+Traversal+of+Directed+Hypergraphs{#}0$\delimiter"026E30F$nhttp://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:optimal+traversal+of+directed+hypergraphs{#}0`

Bateni, M., Chekuri, C., Ene, A., Hajiaghayi, M. T., Korula, N., & Marx, D. (2011). Prize-collecting Steiner problems on planar graphs. *SIAM*, (pp. 1028–1049). `http://dl.acm.org/citation.cfm?id=2133036.2133115`

Bateni, M., & Hajiaghayi, M. (2011). Euclidean Prize-Collecting Steiner Forest. *Algorithmica (New York)*, (pp. 1–24).

Bateni, M., Hajiaghayi, M., & Marx, D. (2010). Prize-collecting Network Design on Planar Graphs. *Computer*, (pp. 1–27). `http://arxiv.org/abs/1006.4339`

Bateni Mohammadhossein; Hajiaghayi, M. M. D. (2011). Approximation Schemes for Steiner Forest on Planar Graphs and Graphs of Bounded Treewidth. *Journal of the ACM*, *58*(5), 1–37.

Battle, A., Jonikas, M. C., Walter, P., Weissman, J. S., & Koller, D. (2010). Automated identification of pathways from quantitative genetic interaction data. *Molecular systems biology*, *6*(379), 1–13. `http://www.ncbi.nlm.nih.gov/pubmed/20531408`

Bauer-Mehren, A., Furlong, L. I., & Sanz, F. (2009). Pathway databases and tools for their exploitation: benefits, current limitations and challenges. *Molecular systems biology*, *5*(290), 1–13. `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2724977{&}tool=pmcentrez{&}rendertype=abstract`

Biggs, N., Lloyd, E. K., & Wilson, R. J. (1986). *Graph Theory, 1736-1936*. New York, NY, USA: Clarendon Press.

Cambini, R., Gallo, G., & Scutellà, M. G. (1997). Flows on hypergraphs. *Mathematical Programming*, *78*(2), 195–217. `http://link.springer.com/10.1007/BF02614371`

Chekuri, C., Ene, A., & Korula, N. (2010). Prize-Collecting Steiner Tree and Forest in Planar Graphs. *Data Structures and Algorithms*, (pp. 1–24).

Chen, X., Tukachinsky, H., Huang, C. H., Jao, C., Chu, Y. R., Tang, H. Y., Mueller, B., Schulman, S., Rapoport, T. A., & Salic, A. (2011). Processing and turnover of the Hedgehog protein in the endoplasmic reticulum. *Journal of Cell Biology*, *192*(5), 825–838.

Chon, T.-S., Park, Y. S., Moon, K. H., & Cha, E. Y. (1996). Patternizing communities by using an artificial neural network. *Ecological Modelling*, *90*(1), 69–78. `http://www.sciencedirect.com/science/article/pii/0304380095001484`

Conant, J. (2008). Boolean formulae, hypergraphs and combinatorial topology. `http://arxiv.org/abs/0808.0739`

Croft, D., Mundo, A. F., Haw, R., Milacic, M., Weiser, J., Wu, G., Caudy, M., Garapati, P., Gillespie, M., Kamdar, M. R., Jassal, B., Jupe, S., Matthews, L., May, B., Palatnik, S., Rothfels, K., Shamovsky, V., Song, H., Williams, M., Birney, E., Hermjakob, H., Stein, L., & D'Eustachio, P. (2014). The Reactome pathway knowledgebase. *Nucleic Acids Research*, *42*(Database), 472–477.

Cusick, M. E., Yu, H., Smolyar, A., Venkatesan, K., Carvunis, R., Simonis, N., Rual, J.-f., Borick, H., Braun, P., Dreze, M., Galli, M., Yazaki, J., Hill, D. E., Ecker, J. R., Roth, F. P., & Vidal, M. (2009). Literature-curated protein interaction datasets. *Nature Methods*, *6*(1), 39–46.

Demir, E., Babur, Ö., Rodchenkov, I., Aksoy, B. A., Fukuda, K. I., Gross, B., Sümer, O. S., Bader, G. D., & Sander, C. (2013). Using Biological Pathway Data with Paxtools. *PLoS Computational Biology*, *9*(9), e1003194. `http://dx.plos.org/10.1371/journal.pcbi.1003194`

Demir, E., Cary, M. P., Paley, S., Fukuda, K., Lemer, C., Vastrik, I., Wu, G., D'Eustachio, P., Schaefer, C., Luciano, J., Schacherer, F., Martinez-Flores, I., Hu, Z., Jimenez-Jacinto, V., Joshi-Tope, G., Kandasamy, K., Lopez-Fuentes, A. C., Mi, H., Pichler, E., Rodchenkov, I., Splendiani, A., Tkachev, S., Zucker, J., Gopinath, G., Rajasimha, H., Ramakrishnan, R., Shah, I., Syed, M., Anwar, N., Babur, Ö., Blinov, M., Brauner, E., Corwin, D., Donaldson, S., Gibbons, F., Goldberg, R., Hornbeck, P., Luna, A., Murray-Rust, P., Neumann, E., Reubenacker, O., Samwald, M., van Iersel, M., Wimalaratne, S., Allen, K., Braun, B., Whirl-Carrillo, M., Cheung, K.-H., Dahlquist, K., Finney, A., Gillespie, M., Glass, E., Gong, L., Haw, R., Honig, M., Hubaut, O., Kane, D., Krupa, S., Kutmon, M., Leonard, J., Marks, D., Merberg, D., Petri, V., Pico, A., Ravenscroft, D., Ren, L., Shah, N., Sunshine, M., Tang, R., Whaley, R., Letovksy, S., Buetow, K. H., Rzhetsky, A., Schachter, V., Sobral, B. S., Dogrusoz, U., McWeeney, S., Aladjem, M., Birney, E., Collado-Vides, J., Goto, S., Hucka, M., Novère, N. L., Maltsev, N., Pandey, A., Thomas, P., Wingender, E., Karp, P. D., Sander, C., & Bader, G. D. (2010). The BioPAX community standard for pathway data sharing. *Nature Biotechnology*, *28*(9), 935–942. `http://www.nature.com/doifinder/10.1038/nbt.1666`

Estrada, E., & Rodríguez-Velázquez, J. A. (2006). Subgraph centrality and clustering in complex hyper-networks. *Physica A: Statistical Mechanics and its Applications*, *364*, 581–594.

Fukuda, K., & Takagi, T. (2001). Knowledge representation of signal transduction pathways. *Bioinformatics (Oxford, England)*, *17*(9), 829–837. `http://www.ncbi.nlm.nih.gov/pubmed/11590099`

Gallagher, S. R., & Goldberg, D. S. (2013). Clustering Coefficients in Protein Interaction Hypernetworks Categories and Subject Descriptors. *BCB-ACM*, (pp. 552–560).

Gallet, A. (2011). Hedgehog morphogen: From secretion to reception. *Trends in Cell Biology*, *21*(4), 238–246. `http://dx.doi.org/10.1016/j.tcb.2010.12.005`

Gallo, G., Longo, G., Pallottino, S., & Nguyen, S. (1993). Directed hypergraphs and applications. *Discrete Applied Mathematics*, *42*, 177–201. `http://www.sciencedirect.com/science/article/pii/0166218X9390045P`

Garey, M. R., & Johnson, D. S. (2016). The Rectilinear Steiner Tree Problem is NP-Complete. *SIAM Journal on Applied Mathematics*, *32*(4), 826–834.

Hajiaghayi, M., Khandekar, R., Kortsarz, G., & Nutov, Z. (2010). Prize-collecting Steiner network problems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *LNCS 6080*, 71–84.

Haus, U.-U., Niermann, K., Truemper, K., & Weismantel, R. (2009). Logic integer programming models for signaling networks. *Journal of computational biology: a journal of computational molecular cell biology*, *16*(5), 725–743.

Haverty, P. M., Hansen, U., & Weng, Z. (2004). Computational inference of transcriptional regulatory networks from expression profiling and transcription factor binding site identification. *Nucleic acids research*, *32*(1), 179–88. `http://nar.oxfordjournals.org/content/32/1/179.full`

Hu, Z., Mellor, J., Wu, J., Kanehisa, M., Stuart, J. M., & DeLisi, C. (2007). Towards zoomable multidimensional maps of the cell. *Nature Biotechnology*, *25*(5), 547–554.

Hui, C.-c., & Angers, S. (2011). Gli Proteins in Development and Disease. *Annual Review of Cell and Developmental Biology*, *27*(1), 513–537.

Johnson, D. S., Minkoff, M., & Phillips, S. (2000). The prize collecting Steiner tree problem: theory and practice. *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, (pp. 760–769). `http://dl.acm.org/citation.cfm?id=338219.338637`

Kandasamy, K., Mohan, S. S., Raju, R., Keerthikumar, S., Kumar, G. S. S., Venugopal, A. K., Telikicherla, D., Navarro, J. D., Mathivanan, S., Pecquet, C., Gollapudi, S. K., Tattikota, S. G., Mohan, S., Padhukasahasram, H., Subbannayya, Y., Goel, R., Jacob, H. K., Zhong, J., Sekhar, R., Nanjappa, V., Balakrishnan, L., Subbaiah, R., Ramachandra, Y., Rahiman, B. A., Prasad, T. K., Lin, J.-X., Houtman, J. C., Desiderio, S., Renauld, J.-C., Constantinescu, S. N., Ohara, O., Hirano, T., Kubo, M., Singh, S., Khatri, P., Draghici, S., Bader, G. D., Sander, C., Leonard, W. J., & Pandey, A. (2010). NetPath: a public resource of curated signal transduction pathways. *Genome Biology*, *11*(1), R3. `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2847715/$\delimiter"026E30F$nhttp://www.ncbi.nlm.nih.gov/pmc/articles/PMC2847715/pdf/gb-2010-11-1-r3.pdf`

Karlebach, G., & Shamir, R. (2008). Modelling and analysis of gene regulatory networks. *Nature reviews. Molecular cell biology*, *9*(10), 770–780.

Khatri, P., Sirota, M., & Butte, A. J. (2012). Ten years of pathway analysis: Current approaches and outstanding challenges. *PLoS Computational Biology*, *8*(2), e1002375.

Kirouac, D. C., Saez-Rodriguez, J., Swantek, J., Burke, J. M., Lauffenburger, D. A., & Sorger, P. K. (2012). Creating and analyzing pathway and protein interaction compendia for modelling signal transduction networks. *BMC Systems Biology*, *6*(29), 1–18. `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3436686{&}tool=pmcentrez{&}rendertype=abstract`

Klamt, S., Haus, U.-U., & Theis, F. (2009). Hypergraphs and Cellular Networks. *PLoS Computational Biology*, *5*(5), e1000385. `http://dx.plos.org/10.1371/journal.pcbi.1000385`

Klamt, S., Saez-Rodriguez, J., Lindquist, J. A., Simeoni, L., & Gilles, E. D. (2006). A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC bioinformatics*, *7*(56), 1–26. `http://www.biomedcentral.com/1471-2105/7/56`

Knuth, D. E., Larrabee, T., & Roberts, P. M. (1987). Mathematical Writing.

Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., Devon, K., Dewar, K., Doyle, M., FitzHugh, W., et al. (2001). Initial sequencing and analysis of the human genome. *Nature*, *409*(6822), 860–921.

Matthews, L., Gopinath, G., Gillespie, M., Caudy, M., Croft, D., de Bono, B., Garapati, P., Hemish, J., Hermjakob, H., Jassal, B., Kanapin, A., Lewis, S., Mahajan, S., May, B., Schmidt, E., Vastrik, I., Wu, G., Birney, E., Stein, L., & D'eustachio, P. (2009). Reactome knowledgebase of human biological pathways and processes. *Nucleic Acids Research*, *37*(Database), 619–622.

Meysman, P., Saeys, Y., Sabaghian, E., Bittremieux, W., Ban de Peer, Y., Goethals, B., & Laukens, K. (2015). Discovery of Significantly Enriched Subgraphs Associated with Selected Vertices in a Single Discovery of Significantly Enriched Subgraphs Associated with Selected Vertices in a Single Graph. *ACM*, (Augugust).

Milacic, M., Haw, R., Rothfels, K., Wu, G., Croft, D., Hermjakob, H., D'Eustachio, P., & Stein, L. (2012). Annotating cancer variants and anti-cancer therapeutics in Reactome. *Cancers*, *4*(4), 1180–1211.

Murali-group (2015). Hypergraph algorithms package. `https://github.com/Murali-group/halp`.

Navlakha, S., Gitter, A., & Bar-Joseph, Z. (2012). A Network-based Approach for Predicting Missing Pathway Interactions. *PLoS Computational Biology*, *8*(8).

Newman, M. E. J. (2003). The structure and function of complex networks. *Society for Industrial and Applied Mathemathics Review*, *45*(2), 167–256.

Nguyen, S., Pretolani, D., & Markenzon, L. (1998). On some path problems on oriented hypergraphs. *Informatique théorique et applications(Imprimé)*, *32*(1-3), 1–20. `http://cat.inist.fr/?aModele=afficheN{&}amp;cpsidt=2388950`

Pan, A., Chang, L., Nguyen, A., & James, A. W. (2013). A review of hedgehog signaling in cranial bone development. *Frontiers in Physiology*, *4*(61), 1–14.

Papadimitriou, C., & Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Mineola, NY, USA: Dover.

Paz, A., Brownstein, Z., Ber, Y., Bialik, S., David, E., Sagir, D., Ulitsky, I., Elkon, R., Kimchi, A., Avraham, K. B., Shiloh, Y., & Shamir, R. (2011). SPIKE: A database of highly curated human signaling pathways. *Nucleic Acids Research*, *39*(Database), 793–799.

Rahman, A., Poirel, C. L., Badger, D. J., Estep, C., & Murali, T. (2013). Reverse Engineering Molecular Hypergraphs. *IEEE/ACM Trans Comput Biol Bioinform*, *10*(5), 1113–1124.

Ramadan, E., Perincheri, S., & Tuck, D. (2010). A hyper-graph approach for analyzing transcriptional networks in breast cancer. *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, (pp. 556–562). `http://portal.acm.org/citation.cfm?doid=1854776.1854882`

Ritz, A. (2016). private communication. Reed College Department of Biology.

Ritz, A., Avent, B., & Murali, T. M. (2014a). Pathway Analysis with Signaling Hypergraphs. *IEEE Transactions on Computational Biology and Bioinformatics*, (pp. 249–258).

Ritz, A., & Murali, T. M. (2014). Pathway Analysis with Signaling Hypergraphs. *BCB-ACM*, (pp. 249–258).

Ritz, A., Tegge, A. N., Kim, H., Poirel, C. L., & Murali, T. (2014b). Signaling hypergraphs. *Trends in Biotechnology*, *32*(7), 356–362. `http://linkinghub.elsevier.com/retrieve/pii/S0167779914000717`

Samaga, R., & Klamt, S. (2013). Modeling approaches for qualitative and semi-quantitative analysis of cellular signaling networks. *Cell Communication and Signaling*, *11*(43), 1–19.

Schaefer, C. F., Anthony, K., Krupa, S., Buchoff, J., Day, M., Hannay, T., & Buetow, K. H. (2009). PID: the Pathway Interaction Database. *Nucleic Acids Research*, *37*(Database), D674–D679. `http://nar.oxfordjournals.org/lookup/doi/10.1093/nar/gkn653`

Sharan, R., Ulitsky, I., & Shamir, R. (2007). Network-based prediction of protein function. *Molecular Systems Biology*, *3*(88), 1–13. `<GotoISI>://WOS:000245381300003`

SHARPEN (2012). SHARPEN hypergraph implementation. `http://sharpen.engr.colostate.edu/mediawiki/index.php/Hypergraph_implementation`

Shields, R. (2012). Cultural Topology: The Seven Bridges of Konigsburg, 1736. *Theory, Culture & Society*, *29*(4-5), 43–57. `http://tcs.sagepub.com/cgi/doi/10.1177/0263276412451161`

Taylor, I. W., Linding, R., Warde-Farley, D., Liu, Y., Pesquita, C., Faria, D., Bull, S., Pawson, T., Morris, Q., & Wrana, J. L. (2009). Dynamic modularity in protein interaction networks predicts breast cancer outcome. *Nature biotechnology*, *27*(2), 199–204. `http://dx.doi.org/10.1038/nbt.1522`

Thakur, M., & Tripathi, R. (2009). Linear connectivity problems in directed hypergraphs. *Theoretical Computer Science*, *410*, 2592–2618. `http://linkinghub.elsevier.com/retrieve/pii/S0304397509002011`

Tuncbag, N., Braunstein, A., Pagnani, A., Huang, S.-S. C., Chayes, J., Borgs, C., Zecchina, R., & Fraenkel, E. (2013). Simultaneous Reconstruction of Multiple Signaling Pathways via the Prize-Collecting Steiner Forest Problem. *Journal of Computational Biology*, *20*(2), 124–136. `http://online.liebertpub.com/doi/abs/10.1089/cmb.2012.0092`

Vyas, N., Goswami, D., Manonmani, A., Sharma, P., Ranganath, H. A., VijayRaghavan, K., Shashidhara, L. S., Sowdhamini, R., & Mayor, S. (2008). Nanoscale Organization of Hedgehog Is Essential for Long-Range Signaling. *Cell*, *133*(7), 1214–1227.