Winning the Clay Prize in Mathematics feat. Barney Potter

A Thesis
Presented to
The Interdisciplinary Committee for Mathematics & Biology
Reed College

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

#bdawg

May 2016

Approved for the Committee
()

Dr. Anna Ritz

Dr. James Fix

Acknowledgements

I want to thank a few people.

Preface

This is an example of a thesis setup to use the reed thesis document class.

Table of Contents

List of Tables

List of Figures

Abstract

The preface pretty much says it all.

Dedication

You can have a dedication here if you wish.

Introduction

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 *cell signaling networks*, which are responsible for almost every process within a cell. 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 (CITATION).

0.1 Representations of PPI Networks

0.2 Graph Theory

Chapter 1

Cell Signaling Networks

Chapter 2

Hypergraphs & Hyperpaths

While standard 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 only 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 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 not possible *(is this the case?)* to tell what the inputs or outputs of a biochemical reaction may be.

Another shortcoming of standard graphs[1] 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

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

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.

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 for. 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.

2.1 Hypergraph Definitions

Where a directed graph represents directed, pairwise interactions between only two 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 $T(e)$, $H(e) \neq \emptyset$. We refer to $T(e)$ as the *tail* of the hyperedge, and to $H(e)$ as the *head* of the hyperedge.

Furthermore, 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 $T(e)$, $H(e) \neq \emptyset$.

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 *(cite Anna's paper on K shortest hyperpaths).*

2.2 Hyperpaths

In order to find the shortest route between two vertices, in a directed hypergraph, we define the notion of a *hyperpath*, P, on the directed hypergraph $\mathcal{H}$. We think of P as the list of vertices and hyperedges that one must pass through in order to traverse through the directed hypergraph from vertex s to get to t.

2.2.1 Definition

2.2.2 Finding k-Shortest Hyperpaths

2.3 Algorithms on Hypergraphs

2.3.1 Integer Linear Programming

2.3.2 NP-Hardness

Chapter 3

The Prize-Collecting Steiner Problem

3.1 Definition of Prize-Collecting Steiner Trees

3.1.1 Prize-Collecting Steiner Forests

Chapter 4

PCST in Directed Signaling Hypergraphs

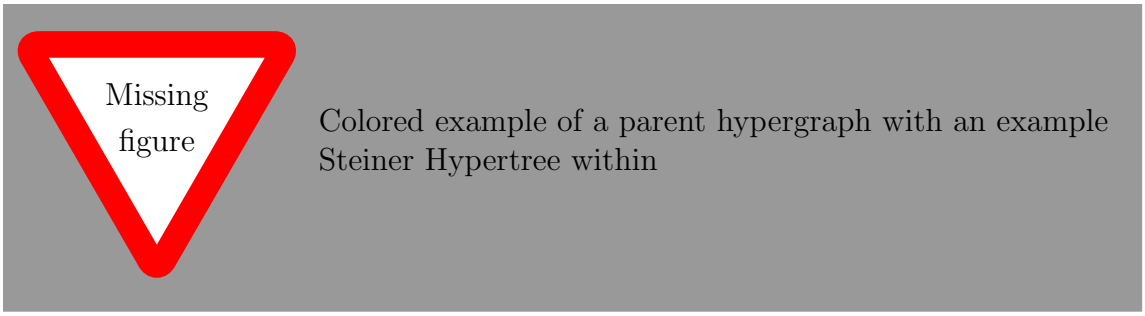
In order for us define and develop a Prize-Collecting Steiner Hyptertree, we begin by formulating the notion of a *Steiner Hypertree*. To make our Steiner Hypertree, which we will call $\mathcal{S}$ we begin with a given "parent" hypergraph, $\mathcal{H} = (V, E)$, and a set of target nodes, such that $T \subseteq V$. We refer to each hypernode as some x such that $x \in V$[1], and each edge is some e such that $e \in E$. When building $\mathcal{H}$, we initially "seed" the hypergraph. Each hyperedge is given a cost associated with including that edge in $\mathcal{S}$. We 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 *dangling* hypernode. We say a hypernode is dangling if it is included in $\mathcal{S}$, and it has no incoming hyperedge which is also included in $\mathcal{S}$ (i.e. the vertex is not in the head of any edge in $\mathcal{S}$).

We now can define the Steiner Hypertree, $\mathcal{S}$, to be the set of hypernodes and hyperedges for which total vertex prizes are maximized, and total edge weights (and dangling penalties) are minimized. The Steiner Hypertree produced should satisfy the following properties:

- The prizes for vertices included in $\mathcal{S}$ should be maximized.
- The costs for edges included in $\mathcal{S}$ should be minimized.
- All members of T are included in $\mathcal{S}$.
- Each vertex that is incident (included in either the head or the tail) on an edge included in $\mathcal{S}$ is also in $\mathcal{S}$.
- All vertices included in $\mathcal{S}$ satisfy one of two conditions:
  - The vertex has at least one incoming hyperedge that is also in $\mathcal{S}$.
  - The vertex is dangling.

[1] We use "x" as an element of V, rather than lowercase "v" to avoid the visual confusion between lowercase v and uppercase V (REMOVE THIS IF CHANGING TO V'S LOOKS OKAY)

- Hanging penalties for dangling nodes included in $\mathcal{S}$ are minimized.



Colored example of a parent hypergraph with an example Steiner Hypertree within

Figure 4.1: Here is an example parent graph, $\mathcal{H}$, and its corresponding Steiner Hypergraph, $\mathcal{S}$. Target hypernodes, T are shown in red, Steiner hypernodes and hyperedges included in $\mathcal{S}$ are shown in blue, and the the dangling node(s) are in green.

4.1 Steiner Hypertree ILP

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

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

$$\underset{\alpha,\delta}{\operatorname{argmax}} \sum_{v \in V} g_v \alpha_v - \sum_{e \in E} c_e \alpha_e - \sum_{v \in V} h_v \delta_v \tag{4.1}$$

Subject to the set of linear constraints:

$$\alpha_v \geq 1 \qquad \forall\, v \in T \tag{4.2}$$

$$\sum_{v \in H_e} \alpha_v \leq |H_e| \alpha_e \qquad \forall\, v \in H_V, e \in H_E \tag{4.3}$$

$$\sum_{v \in T_e} \alpha_v \leq |T_e| \alpha_e \qquad \forall\, v \in H_V, e \in H_E \tag{4.4}$$

$$\delta_v \leq \alpha_v \qquad \forall\, v \in \mathcal{H} \tag{4.5}$$

$$\delta_v \geq 1 - \sum_{e : v \in H_e} \alpha_e \qquad \forall\, v \in \mathcal{H} \tag{4.6}$$

$$0 \leq \delta_v \leq 1 \qquad \forall\, v \in \mathcal{H} \tag{4.7}$$

Constraint (4.2) encodes that every target node in T is in the solution, $\mathcal{S}$. Constraints (4.3) and (4.4) ensure that if an edge is in $\mathcal{S}$, any nodes incident on that edge will also be in $\mathcal{S}$. Finally, constraints (4.5), (4.6), and (4.7) encode the ability for nodes to dangle if they do not have an incoming edge. Though this formulation encodes all of the properties of a Steiner Hypertree, on some hypergraphs, it can yield subtrees that are trivial, due to the possibility of simple cycles between target nodes and Steiner nodes. We solve this problem by introducing one more variable to the program to impose a topological ordering on the nodes.

4.2 Figures

If your thesis has a lot of figures, LaTeX might behave better for you than that other word processor. One thing that may be annoying is the way it handles "floats" like tables and figures. LaTeX will try to find the best place to put your object based on the text around it and until you're really, truly done writing you should just leave it where it lies. There are some optional arguments to the figure and table environments to specify where you want it to appear; see the comments in the first figure.

If you need a graphic or tabular material to be part of the text, you can just put it inline. If you need it to appear in the list of figures or tables, it should be placed in the floating environment.

To get a figure from StatView, JMP, SPSS or other statistics program into a figure, you can print to pdf or save the image as a jpg or png. Precisely how you will do this depends on the program: you may need to copy-paste figures into Photoshop or other graphic program, then save in the appropriate format.

Below we have put a few examples of figures. For more help using graphics and the float environment, see our online documentation.

And this is how you add a figure with a graphic:

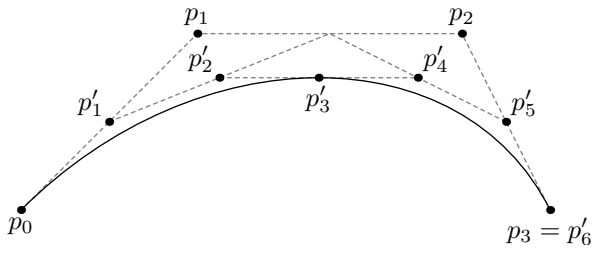


Figure 4.2: A Figure

4.3 More Figure Stuff

You can also scale and rotate figures.

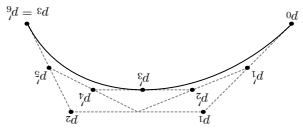


Figure 4.3: A Smaller Figure, Flipped Upside Down

4.4 Even More Figure Stuff

With some clever work you can crop a figure, which is handy if (for instance) your EPS or PDF is a little graphic on a whole sheet of paper. The viewport arguments are the lower-left and upper-right coordinates for the area you want to crop.

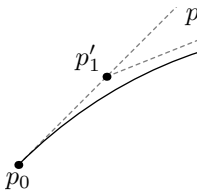


Figure 4.4: A Cropped Figure

4.4.1 Common Modifications

The following figure features the more popular changes thesis students want to their figures. This information is also on the web at `web.reed.edu/cis/help/latex/graphics.html`.

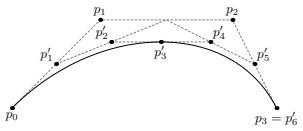


Figure 0.9: Interaction bar plot showing the degree of specialization for each flower type.

Conclusion

Here's a conclusion, demonstrating the use of all that manual incrementing and table of contents adding that has to happen if you use the starred form of the chapter command. The deal is, the chapter command in LaTeX does a lot of things: it increments the chapter counter, it resets the section counter to zero, it puts the name of the chapter into the table of contents and the running headers, and probably some other stuff.

So, if you remove all that stuff because you don't like it to say "Chapter 4: Conclusion", then you have to manually add all the things LaTeX would normally do for you. Maybe someday we'll write a new chapter macro that doesn't add "Chapter X" to the beginning of every chapter title.

4.1 More info

And here's some other random info: the first paragraph after a chapter title or section head *shouldn't be* indented, because indents are to tell the reader that you're starting a new paragraph. Since that's obvious after a chapter or section title, proper typesetting doesn't add an indent there.

Appendix A

The First Appendix

Appendix B

The Second Appendix, for Fun

References

Ausiello, G., Italiano, G., & 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$\backslash$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. (pp. 1028–1049). http://dl.acm.org/citation.cfm?id=2133036.2133115

Gallo, G., Longo, G., Pallottino, S., & Nguyen, S. (1993). Directed hypergraphs and applications. *Discrete Applied Mathematics*, *42*(2-3), 177–201. http://www.sciencedirect.com/science/article/pii/0166218X9390045P

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

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

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. (AUGUST).

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

Ritz, A., Tech, V., & Murali, T. M. (2014a). Pathway Analysis with Signaling Hypergraphs. *XX*(XX), 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

Thakur, M., & Tripathi, R. (2009). Linear connectivity problems in directed hypergraphs. *Theoretical Computer Science*, *410*(27-29), 2592–2618. http://linkinghub.elsevier.com/retrieve/pii/S0304397509002011