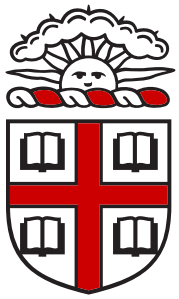
Brown University



Honors Thesis

Random Walks on Hypergraphs with Applications to Disease-Gene Prioritization

Uthsav Chitra

Advisor: Professor Ben Raphael

April 15, 2017

Abstract

Typically, gene interaction networks are expressed as graphs, storing pairwise interactions between genes. Because of the vast amount of literature on statistical graph inference, this is a useful representation in practice. However, such a pairwise representation ignores more complex features of gene interactions, such as gene regulation and assembly. In this thesis, we propose a hypergraph model for gene interaction networks. Since many network-based algorithms rely on random walks, we analyze our model through the viewpoint of random walks. We outline a general framework for random walks on a hypergraph, and we show in a precise sense that random walks on hypergraphs are not special cases of random walks on graphs.

We also define a mapping between hypergraphs and graphs, and use that mapping to build a hypergraph from an already-existing graph representation of a gene interaction network. We then use this hypergraph network to perform disease-gene prioritization via the PageRank algorithm. For monogenic diseases, we find that the hypergraph noticeably outperforms the graph, demonstrating the value of using hypergraphs as gene interaction networks.

Contents

1 Introduction

A gene interaction network is a graph $G = (V, E)$, where V, the vertices, are genes, and $E \subset V \times V$, the edges, are functional associations between pairs of genes. Because of the vast amount of gene interaction data, network approaches have proved fruitful in both inferring high-level associations between groups of genes and in determining the relative importance of genes in such groups. These insights, in turn, have powered recent algorithms in many biological problems. For example, many algorithms analyze gene interaction networks in order to determine which genes have mutations associated with different types of cancer [22, 20, 5, 31].

Because there is so much literature on graph inference, it is easy to work with a graph representation of a gene interaction network. However, it is unclear if a graph is the *best* representation of genetic information. For example, a graph might represent a protein complex—a group of genes whose protein products have associated functions—by adding edges between all of the genes in the complex. But such a pairwise representation ignores that all of the genes in a complex act *together*. Graphs are also unable to encode other high-level features of gene interactions, such as regulation and assembly [27].

To account for this, we propose a hypergraph model for gene interaction networks. A hypergraph is a pair (V, E), where V is a set of vertices, and $E \subset 2^V$ is a set of hyperedges. (To avoid confusion, we refer to graphs, in the sense of the first two paragraphs, as simple graphs.) The difference between a hypergraph and a simple graph is that edges in a hypergraph can have more than two vertices, which allows for higher-order relations between genes. Going back to our protein complex example, a hypergraph can represent a protein complex as one single hyperedge, whereas a simple graph would have to use many more edges to connect each gene in the complex (and even then, it may not be clear whether a particular group of vertices is a complex).

The problem with wanting to use a hypergraph representation of a gene interaction network is that, currently, none exist. (That is, all existing biological databases record only pairwise gene interactions.) Despite their apparent modeling advantages, hypergraphs are not studied nearly as much as simple graphs. One solution, which we explore in this thesis, is to build a hypergraph from an already-existing simple graph. There are many simple graph representations of gene interaction networks in the literature [15, 26, 13, 18], so we can take one of these simple graphs, identify the protein complexes in the graph, and turn the protein complexes into hyperedges. This process yields a hypergraph which can be used as a gene interaction network.

We examine the effectiveness of the hypergraph representation of a gene interaction network by looking at the process of *disease-gene prioritization*. In disease-gene prioritization, one wants to infer genes associated with a specific disease, given a partially complete list of genes already known to be associated with the disease. Köhler et al. [21] use the PageRank algorithm [24], also known as the random walk with restart, on a gene interaction network (represented as a simple graph) to perform disease-gene prioritization. If one builds a hypergraph from a simple graph, then one can adapt Köhler's framework to perform disease-gene prioritization on the hypergraph.

Since random walks are the basis of the PageRank algorithm, we study random walks on hyper-

graphs in this thesis. In Section 2, we define a hypergraph, give a mapping between hypergraphs and simple graphs, and outline a general framework for random walks on hypergraphs. In Section 3, we try to answer the following question: given a hypergraph H, does there exist a simple graph G such that a random walk on G has the same probability transition matrix as a random walk on H? We show that such a G does not exist in general—telling us that random walks on hypergraphs are, in fact, different than random walks on simple graphs—and give conditions on the weights of H for when such a G exists. We also give a counterexample for the other direction: given a simple graph G, there does not always exist a hypergraph H such that a random walk on H has the same probability transition matrix as a random walk on G. Finally, in Section 4, we use our mapping from Section 2 to build a hypergraph from simple graph model of a gene interaction network. Following Köhler's framework, we use PageRank to perform disease-gene prioritization on that hypergraph. We find that, for monogenic diseases, the hypergraph model outperforms the simple graph model, demonstrating the value of using hypergraphs as gene interaction networks.

1.1 Related Work

A fundamental concept in spectral graph theory is the *Laplacian matrix* L of a (simple) graph G. The Laplacian is defined as $L = D - A$, where D is the degree matrix and A is the adjacency matrix of G. The spectrum of L captures many important properties of G, such as mixing time of random walks on G, and is extremely useful for performing inference on G. Because of the close connection between the Laplacian and graph inference, most of the early literature on learning and inference using hypergraphs focuses on defining a Laplacian matrix for hypergraphs, and then either proving theorems about the spectra of the Laplacian or using the Laplacian for applications.

Every paper on the spectra of hypergraph Laplacians owes its beginnings to Fan Chung, who was one of the first people to start investigating such spectra. In her first paper on the subject, Chung [10] defines the Laplacian of a d-regular hypergraph (where every hyperedge has size d) through homology theory and gives bounds on its spectra. Other instances and applications of spectral hypergraph theory include: graph clustering [2, 32], multi-label classification [28], classifying gene expression data [29], and image segmentation [16].

Agarwal et al. [1] survey different constructions of the hypergraph Laplacian throughout the literature. They show that each of these hypergraph Laplacian constructions is the same as the Laplacian of a closely related simple graph, indicating that studying hypergraph spectra provides no additional information compared to studying only the spectra of simple graphs. While this is certainly a negative result, our application of hypergraphs uses the PageRank algorithm, which does not rely directly on the Laplacian.

Chan et al. [9] define the hypergraph Laplacian through the notion of heat diffusion, rather than as a function of adjacency and incidence matrices. Their Laplacian is a non-linear operator, so it does not fall under Agarwal's conclusions above. However, because of the complexity of their Laplacian, computations involving it are time-consuming. The authors mention random walks when defining the framework for their hypergraph Laplacian, but most of the work consists of examing spectral

properties of their Laplacian and using it to give algorithms for hypergraph cuts.

Bellaachia and Al-Dhelaan [6] give a framework for a hypergraph random walks that is slightly less general than ours. They use their random walk model to perform the PageRank algorithm on a real-world hypergraph, with various choices of weights, and compare the results of different weight choices. Berlt et al. [8] also perform PageRank on a (directed) hypergraph, although they do not relate PageRank to random walks on a hypergraph. They then apply their hypergraph PageRank to the problem of ranking web pages and show that they get better results compared to the original simple graph version of PageRank.

Cooper et al. [12] and Avin et al. [4] both analyze random walks on hypergraphs from a theoretical perspective, proving bounds on the cover times of hypergraph random walks. Cooper considers random hypergraphs, while Avin looks at a fixed hypergraph.

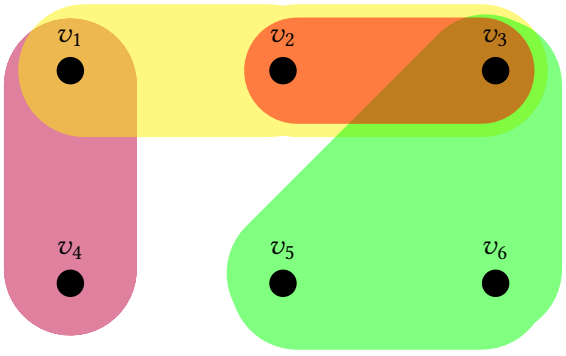
2 Hypergraphs

Definition 2.1

Let V be a (finite) set, and let $E \subset 2^V$, with $|e| > 1$ for all $e \in E$. We call $H = (V, E)$ a **hypergraph** with vertices V and hyperedges E. We call a hypergraph a **simple graph** if $|e| = 2$ for all $e \in E$.

Moreover, for each vertex $u \in V$, we define $N(u) \triangleq \{e \in E : u \in e\}$, or equivalently, the set of hyperedges incident to u.

Hypergraphs are a generalization of simple graphs, where edges can connect more than two vertices.

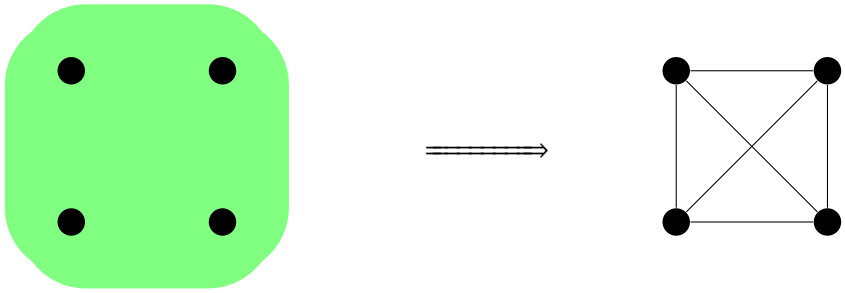


The following two definitions are motivated by our goal of mapping hypergraphs to simple graphs and vice-versa.

Definition 2.2

Let $H = (V, E)$ be a hypergraph. We define the **clique graph** $G(H)$ as a simple graph on V, with an edge between $u, v \in V$ if there is a hyperedge $e \in E$ with $u, v \in e$.

The name clique graph comes from the fact that, if $e = \{v_1, ..., v_n\}$ is a hyperedge in H, then $v_1, ..., v_n$ form a clique in G.



H $G(H)$

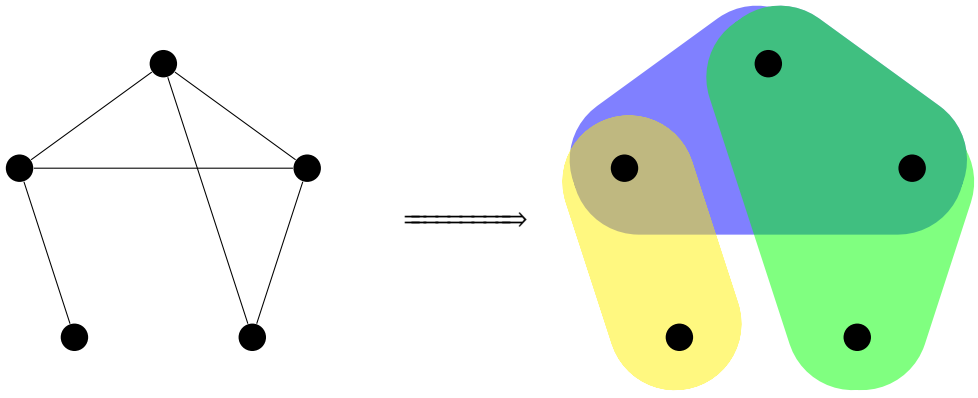
Definition 2.3

Let $G = (V, E)$ be a simple graph. Let $C_1, ..., C_k$ be the maximal cliques of G, and let

$$E_1 = \{\{u, v\} : u, v \in C_i \text{ for some } i\}$$
$$E_2 = \{C_1, ..., C_k\}.$$

The **maximal hypergraph associated to** G, $H(G)$, is a hypergraph on V with hyperedges $(E \setminus E_1) \cup E_2$.

Less formally, we replace all the maximal cliques of G with hyperedges. Note that maximal cliques of size 2 are unchanged.



G $H(G)$

We note that $H(G)$, the maximal hypergraph of G, is the same as the "maximal covering" definition of Ennis et al. [17]. Given a graph G, there are many effective algorithms, e.g. Bron-Kerbosch, that find all maximal cliques in a graph.

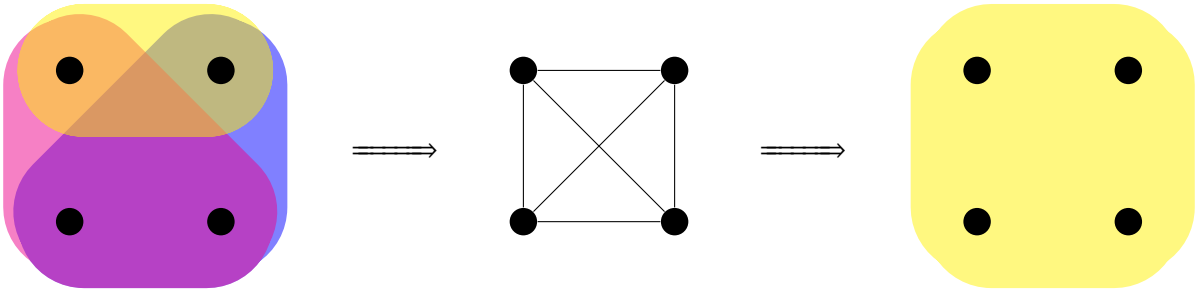
Via the clique graph and maximal hypergraph operators, we get maps between the set of hypergraphs and simple graphs.

$$\text{Hypergraphs} \underset{G(\cdot)}{\overset{H(\cdot)}{\rightleftarrows}} \text{Simple Graphs}$$

The maps are not inverses of each other. In one direction, starting with a simple graph G, it is easy to see that

$$G(H(G)) = G,$$

since the maximal hypergraph operator turns all maximal cliques of G to hyperedges, and the clique graph operator turns all hyperedges back into (maximal) cliques. However, if H is a hypergraph, then $H(G(H))$ is not necessarily H; see below for an example of this.



H $G(H)$ $H(G(H))$

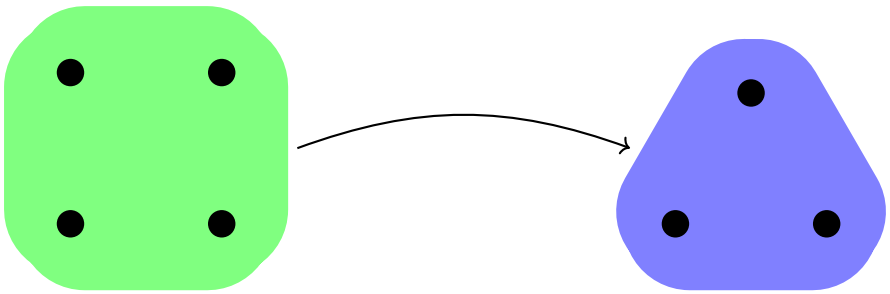
2.1 Directed Hypergraphs

We can define directed hypergraphs, and other related constructions, in a similar way.

Definition 2.4

A **directed hypergraph** $H = (V, E)$ is a set of vertices V together with a set of directed hyperedges E. Each hyperedge is of the form (T, S), with $T, S \subset V$ and $|T|, |S| > 0$. We call T the **tail** of e, and S the **head** of e. H is a **simple directed graph** if $|S| = |T| = 1$ for every hyperedge $e = (T, S)$.

Each hyperedge in a directed hypergraph can be thought of as an arrow from a set of vertices T to a set of vertices S.



T S

Definition 2.5

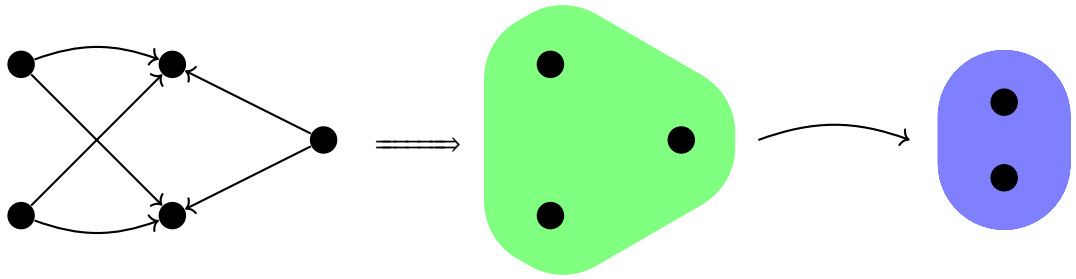
Let $H = (V, E)$ be a directed hypergraph. The **(directed) clique graph** of H, $G(H)$, is a simple directed graph on V, where there is an edge from u to v in $G(H)$ if there is a hyperedge $e = (T, S) \in E$ with $u \in T$ and $v \in S$.

Directed hypergraphs appear to be a more realistic way to model protein interactions: for instance, directed hyperedges can model protein activation.

Definition 2.6

Let $G = (V, E)$ be a simple directed graph. The **maximal directed hypergraph associated to** G, $H(G)$, is a directed hypergraph on V. The hyperedges of $H(G)$ are all pairs $(T, S) \subset V \times V$, where

- $|T|, |S| > 0$,
- for all $u \in T, v \in S$, $(u, v) \in E$, and
- T and S are maximal, in the sense that, if we add any additional vertices to T or S, the previous property will not hold.



G $H(G)$

We can adapt the Bron-Kerbosch algorithm to build a maximal directed hypergraph. Define $A(v) \triangleq \{w \in V : (v, w) \in E\}$, $B(v) \triangleq \{w \in V : (w, v) \in E\}$, and $N(v) \triangleq A(v) \cap B(v)$.

Algorithm 1 Bron-Kerbosch for building maximal directed hypergraph

```
1: procedure BK-MDH(T, P_T, X_T, S, P_S, X_S)
2:     if P_T, X_T, P_S, X_S empty then
3:         return T, S
4:     for each vertex v ∈ P_T do
5:         BK-MDH(T ∪ {v}, P_T ∩ N(v), X_T ∩ N(v), S ∩ A(v), P_S ∩ A(v), X_S ∩ A(v))
6:         P_T ← P_T \ {v}
7:         X_T ← X_T ∪ {v}
8:     for each vertex v ∈ P_S do
```

9: BK-MDH($T \cap B(v), P_T \cap B(v), X_T \cap B(v), S \cup \{v\}, P_S \cap N(v), X_S \cap N(v)$)
10: $P_S \leftarrow P_S \setminus \{v\}$
11: $X_S \leftarrow X_S \cup \{v\}$

The algorithm recursively finds all maximal T, such that T includes some of the vertices in P_T and none of the vertices in X_T, by looping through each vertex $v \in P_T$ and either including v in T or not including v. When it includes v in T, the other parameters are adjusted accordingly. When it does not include v, it removes v from P_T and adds v to X_T. The algorithm then does the same thing for S.

3 Random Walks

In this section, and in the rest of the thesis, assume all hypergraphs are undirected.

Definition 3.1

A **random walk** on a hypergraph H is a time-homogeneous Markov chain with states V and transition probabilities $p_{u,v} = \mathbb{P}(v_t = v \mid v_{t-1} = u)$.

What are the transition probabilities $p_{u,v}$? Intuitively, at vertex v_t, we can think of the random walk as doing the following.

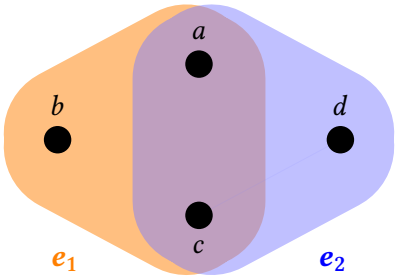
- With probability distribution π, pick a hyperedge e incident to v_t.
- Then, with probability distribution γ, pick a vertex $v_{t+1} \in e$.

To specify π, γ, we propose the following model: each hyperedge e has weight $\omega(e) > 0$, and each vertex v in a hyperedge e has weight $\lambda_e(v)$. Going back to our gene interaction network example, if we let hyperedges denote protein complexes, then the hyperedge weights express the general importance of a protein complex, while the vertex weights represent the relative importance of genes in a protein complex.

Then, π is the distribution on $N(v_t)$ with $\pi(e) \propto \omega(e)$, and γ is the distribution with $\gamma(v) \propto \lambda_e(v)$. It would be natural to write that γ is a distribution on vertices in e. However, that would make it so that $v_{t+1} = v_t$ has positive probability, an event which does not happen in a simple random walk on a simple graph. To fix this issue, we define two types of random walks on a hypergraph H. In the *simple random walk*, γ is a distribution on $e \setminus \{v_t\}$, while in the *lazy random walk*, γ is instead a distribution on e.

Example 3.2

Consider the following hypergraph.



Suppose the hyperedge weights are $\omega(e_1) = 1$ and $\omega(e_2) = 2$, and the vertex weights are

$$\begin{aligned}\big(\lambda_{e_1}(a), \lambda_{e_1}(b), \lambda_{e_1}(c)\big) &= (1, 1, 1),\\ \big(\lambda_{e_2}(a), \lambda_{e_2}(c), \lambda_{e_2}(d)\big) &= (2, 1, 1).\end{aligned}$$

Then, in the simple random walk, the probability of going from c to a in one step is

$$\begin{aligned}p_{c,a} &= \sum_{i=1,2} \mathbb{P}(\text{start from } c, \text{ pick } e_i) \cdot \mathbb{P}(\text{pick } a \text{ from } e_i)\\ &= \left(\frac{\omega(e_1)}{\omega(e_1)+\omega(e_2)}\right)\left(\frac{\lambda_{e_1}(a)}{\lambda_{e_1}(a)+\lambda_{e_1}(b)}\right) + \left(\frac{\omega(e_2)}{\omega(e_1)+\omega(e_2)}\right)\left(\frac{\lambda_{e_2}(a)}{\lambda_{e_2}(a)+\lambda_{e_2}(d)}\right)\\ &= \frac{1}{3}\cdot\frac{1}{2} + \frac{2}{3}\cdot\frac{2}{3}\\ &= \frac{11}{18},\end{aligned}$$

while in the lazy random walk, the probability of going from c to a in one step is

$$\begin{aligned}p_{c,a} &= \sum_{i=1,2} \mathbb{P}(\text{start from } c, \text{ pick } e_i) \cdot \mathbb{P}(\text{pick } a \text{ from } e_i)\\ &= \left(\frac{\omega(e_1)}{\omega(e_1)+\omega(e_2)}\right)\left(\frac{\lambda_{e_1}(a)}{\lambda_{e_1}(a)+\lambda_{e_1}(b)+\lambda_{e_1}(c)}\right)\\ &\qquad + \left(\frac{\omega(e_2)}{\omega(e_1)+\omega(e_2)}\right)\left(\frac{\lambda_{e_2}(a)}{\lambda_{e_2}(a)+\lambda_{e_2}(c)+\lambda_{e_2}(d)}\right)\\ &= \frac{1}{3}\cdot\frac{1}{3} + \frac{2}{3}\cdot\frac{2}{4}\\ &= \frac{4}{9}.\end{aligned}$$

In general, the simple and lazy random walks do not have identical transition probabilities. Another important point is that each vertex v has multiple vertex weights $\lambda_e(v)$, depending on which hyperedge e we are looking at. For example, a and c have the same weight in e_1, while a has twice the weight of c in e_2.

We can formally define the simple and lazy random walks via their transition probabilities. For

example, the simple random walk has transition probabilities

$$\begin{aligned} p_{u,v} &= \sum_{e \in E} \left(\mathbb{P}(\text{start at } u, \text{ pick hyperedge } e) \cdot \mathbb{1}\{v \in e\} \cdot \mathbb{P}(\text{pick } v \text{ from hyperedge } e) \right) \\ &= \sum_{e \in E} \left[\left(\frac{\omega(e) \cdot \mathbb{1}\{u \in e\}}{\sum_{f \in N(u)} \omega(f)} \right) \cdot \mathbb{1}\{v \in e\} \cdot \left(\frac{\lambda_e(v) \cdot \mathbb{1}(u \neq v)}{\sum_{\substack{z \in e \\ z \neq u}} \lambda_e(z)} \right) \right] \\ &= \sum_{e \in E} \left[\frac{\omega(e)}{\sum_{f \in N(u)} \omega(f)} \cdot \mathbb{1}(u, v \in e) \cdot \frac{\lambda_e(v) \cdot \mathbb{1}(u \neq v)}{\sum_{\substack{z \in e \\ z \neq u}} \lambda_e(z)} \right]. \end{aligned}$$

We summarize this discussion in the definition below.

Definition 3.3

The **simple random walk** on an undirected hypergraph H is a time-homogeneous Markov chain, with states V and transition probabilities

$$p_{u,v} = \sum_{e \in E} \left[\frac{\omega(e)}{\sum_{f \in N(u)} \omega(f)} \cdot \mathbb{1}(u, v \in e) \cdot \frac{\lambda_e(v) \cdot \mathbb{1}(u \neq v)}{\sum_{\substack{z \in e \\ z \neq u}} \lambda_e(z)} \right].$$

The **lazy random walk** on an undirected hypergraph H is also a time-homogeneous Markov chain on V, with transition probabilities

$$p_{u,v} = \sum_{e \in E} \left[\frac{\omega(e)}{\sum_{f \in N(u)} \omega(f)} \cdot \mathbb{1}(u, v \in e) \cdot \frac{\lambda_e(v)}{\sum_{z \in e} \lambda_e(z)} \right].$$

Below we define two particular choices of weights.

- In the **uniform edge-weighted simple random walk on** H (resp. **uniform edge-weighted lazy random walk on** H), $\omega(e) = |e| - 1$ (resp. $\omega(e) = |e|$) and $\lambda_e(v) = 1$, for all hyperedges e and vertices $v \in e$.
- In the **unweighted (simple/lazy) random walk on** H, $\omega(e) = 1$ and $\lambda_e(v) = 1$ for all hyperedges e and vertices $v \in e$.

While the uniform edge-weighted random walk is defined using hyperedge weights, in some sense we consider both the above walks to be unweighted. Both the uniform and unweighted random walks are natural generalizations of random walks on a simple graph; indeed, if H is simple, the uniform edge-weighted simple random walk and the unweighted simple random walk are equivalent to the standard definition of a simple random walk on a simple graph, and likewise for their lazy counterparts. (In the lazy random walk on a simple graph, with probability $\frac{1}{2}$ you stay at a vertex, and with probability $\frac{1}{2}$ you move to a different vertex).

None of the existing literature makes a distinction between simple and lazy random walks. Bellaachia [6] uses the same general framework as us, but only for the lazy random walk; Zhou [32] looks at general edge-weighted lazy random walks; and Avin [4] works with unweighted lazy random walks.

Cooper [12] works with simple random walks on a hypergraph, but not in full generality, as he only considers the uniform edge-weighted and unweighted random walks.

4 Equivalent Random Walks

In this section, assume all hypergraphs are connected, i.e. random walks on hypergraphs are irreducible.

Definition 4.1

Let M and N be Markov chains with the same (countable) state space. Let P^M be the probability transition matrix of M, and P^N the probability transition matrix of N. We say that M and N are **equivalent** if

$$P^M_{x,y} = P^N_{x,y}$$

for all states x, y.

In particular, since a random walk on a hypergraph is a Markov chain on the vertices, two random walks are equivalent if they have the same probability transition matrix. Below are some relatively simple results about equivalent random walks on hypergraphs.

Proposition 4.2

If H is d-regular (that is, every hyperedge has size d), then the uniform edge-weighted and unweighted (simple/lazy) random walks are equivalent.

Proof. Assume we are working with the simple random walk; the proof for lazy random walk follows by the same reasoning. The transition probabilities of the uniform edge-weighted simple random walk are

$$p_{u,v} = \sum_{e \in E} \left[\frac{|e| - 1}{\sum_{f \in N(u)} (|e| - 1)} \cdot \mathbb{1}(u, v \in e) \cdot \frac{1}{\sum_{\substack{z \in e \\ z \neq u}} 1} \right] = \sum_{e \in E} \left[\frac{1}{|N(u)|} \cdot \mathbb{1}(u, v \in e) \cdot \frac{1}{|e| - 1} \right],$$

and the transition probabilities of the unweighted simple random walk are

$$p_{u,v} = \sum_{e \in E} \left[\frac{1}{\sum_{f \in N(u)} 1} \cdot \mathbb{1}(u, v \in e) \cdot \frac{1}{\sum_{\substack{z \in e \\ z \neq u}} 1} \right] = \sum_{e \in E} \left[\frac{1}{|N(u)|} \cdot \mathbb{1}(u, v \in e) \cdot \frac{1}{|e| - 1} \right].$$

Thus, the random walks have the same transition probabilities, so they are equivalent. □

Proposition 4.3

Suppose H satisfies $|e_1 \cap e_2| \leq 1$ for every pair of distinct hyperedges e_1, e_2. Then, the uniform

edge-weighted simple random walk on H and the simple random walk on $G(H)$ are equivalent.

Proof. Suppose we are at vertex v, with incident hyperedges $e_1, \ldots, e_n$. Let w be a neighbor of v, with $w \in e_i$. By assumption, $w \notin e_j$ for $j \neq i$. Then, in the uniform edge-weighted simple random walk on H,

$$p_{v,w} = \frac{|e_i| - 1}{\sum_j (|e_j| - 1)} \cdot \frac{1}{|e_i| - 1} = \frac{1}{\sum_j (|e_j| - 1)}.$$

Thus, for any neighbors w_1, w_2 of v, $p_{v,w_1} = p_{v,w_2}$, so we get a walk equivalent to the simple random walk on $G(H)$. □

4.1 Overview of Results

Using our notion of equivalence, we want to answer the following two problems.

Problem 1

Let H be a hypergraph with hyperedge weights $\omega(e)$ and vertex weights $\lambda_e(v)$. Do there exist (positive) edge weights $w_{u,v}$ on $G(H)$ such that the (simple/lazy) random walk on H and the weighted random walk on $G(H)$ are equivalent?

Problem 2

Conversely, given a simple graph G with edge weights $w_{u,v}$, do there exist (positive) hyperedge weights $\omega(e)$ and vertex weights $\lambda_e(v)$ on $H(G)$ such that the weighted random walk on G and the (simple/lazy) random walk on $H(G)$ are equivalent?

Before we go on, there is one small point to make. In $G(H)$, if we have $w_{v,v} > 0$ for some vertex v, then we pretend there is a self-loop at v in $G(H)$. We impose a similar restriction on G.

Unsurprisingly, the answer to Problem 1 is no. We first give a counterexample, and then show that such weights $w_{u,v}$ exist if and only if the random walk on H satisfies a property known as *detailed balance*. This connection to detailed balance will let us prove the following two theorems.

Theorem 4.4

Let H be a hypergraph with hyperedge weights $\omega(e)$ and vertex weights $\lambda_e(v)$, such that $\lambda_e(v) = \lambda_e(w)$ for all $v, w \in e$. Then, there exist weights $w_{u,v}$ on $G(H)$ such that the simple random walk on H and the weighted random walk on $G(H)$ are equivalent, and the weights are given by

$$w_{u,v} = \sum_{e \in N(u) \cap N(v)} \frac{\omega(e)}{|e| - 1}.$$

Theorem 4.5

Let H be a hypergraph with hyperedge weights $\omega(e)$ and vertex weights $\lambda_e(v)$. Suppose that for each hyperedge e, there exists exists a constant c_e such that $c_e\lambda_e(v)$ is constant for all vertices v and hyperedges $e \in N(v)$. Then, there exist weights $w_{u,v}$ on $G(H)$ such that the lazy random walk on H and the weighted random walk on $G(H)$ are equivalent. In particular, constants c_e exist if any of the following hold:

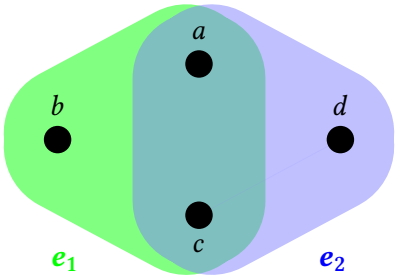
- $\lambda_e(v)$ is constant for all hyperedges e incident to v, or
- for all hyperedges $e_1, e_2 \in H$, $|e_1 \cap e_2| \leq 1$.

More surprisingly, the answer to Problem 2 is no as well, and we give a counterexample to demonstrate this.

4.2 Counterexamples

Counterexample 1 (Counterexample to Problem 1)

Consider the following hypergraph.



Suppose $\omega(e_1) = \omega(e_2) = 1$, and

$$\lambda_{e_1}(v) = 1 \text{ for all } v \in e_1,$$
$$\lambda_{e_2}(a) = \lambda_{e_2}(d) = 1,$$
$$\lambda_{e_2}(c) = 2.$$

The stationary distribution of the hypergraph random walk induced by these weights is

$$(\pi_a, \pi_b, \pi_c, \pi_d) = \left(\frac{204}{643}, \frac{110}{643}, \frac{236}{643}, \frac{93}{643}\right).$$

Let $\{w_{a,b}, w_{a,c}, w_{b,c}, w_{a,d}, w_{c,d}\}$ be (non-negative) weights on $G(H)$, and without loss of generality, assume they sum to 1. Then, the stationary distribution of a simple random walk on $G(H)$ is

$$(w_{a,b} + w_{a,c} + w_{a,d}, w_{a,b} + w_{b,c}, w_{a,c} + w_{b,c} + w_{c,d}, w_{a,d} + w_{c,d}).$$

For a random walk on $G(H)$ to be equivalent to the random walk on H, a necessary condition is that the system of equations

$$(w_{a,b} + w_{a,c} + w_{a,d}, w_{a,b} + w_{b,c}, w_{a,c} + w_{b,c} + w_{c,d}, w_{a,d} + w_{c,d}) = \left(\frac{204}{643}, \frac{110}{643}, \frac{236}{643}, \frac{93}{643}\right)$$
$$w_{a,b} + w_{a,c} + w_{b,c} + w_{a,d} + w_{c,d} = 1,$$

has a non-negative solution. However,

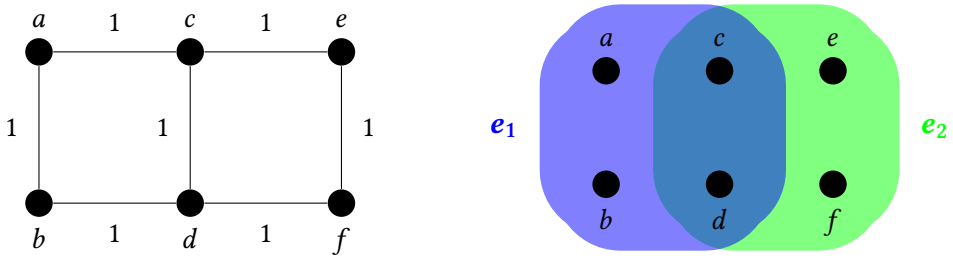
$$\frac{440}{643} = (w_{a,b} + w_{a,c} + w_{a,d}) + (w_{a,c} + w_{b,c} + w_{c,d}) = w_{a,c} + 1,$$

so $w_{a,c} < 0$. Thus, it is impossible to find weights on $G(H)$ that yield a random walk equivalent to the random walk on H.

Below we present a counterexample to Problem 2.

Counterexample 2 (Counterexample to Problem 2)

Consider the following simple graph and its maximal hypergraph.



G $H(G)$

Assume there exist weights hyperedge weights $\omega(e)$ and vertex weights $\lambda_e(v)$ on $H(G)$ such that a random walk on $H(G)$ is equivalent to a simple random walk on G. Because G has no self-loops, the random walk on $H(G)$ must be simple. In $H(G)$, we must have

$$p_{a,b} = p_{a,c} = p_{a,d} = p_{a,e} = \frac{1}{4}.$$

This implies $\lambda_{e_1}(b) = \lambda_{e_1}(c) = \lambda_{e_1}(d)$. Replacing a with b and using the same argument implies $\lambda_{e_1}(a) = \lambda_{e_1}(c) = \lambda_{e_1}(d)$, so all of the vertices in e_1 are weighted equally. The same argument shows that all of the vertices inside e_2 are weighted equally as well.

Let $w(e_1) = p$ and $w(e_2) = 1 - p$. Then, in $H(G)$,

$$p_{d,a} = p \cdot \frac{1}{3} = \frac{1}{5} \implies p = \frac{3}{5}.$$

However, if $p = \frac{3}{5}$, then

$$p_{d,f} = \frac{2}{5} \cdot \frac{1}{3} = \frac{2}{15} \neq \frac{1}{5}.$$

Thus, one cannot find weights on $H(G)$ such that a random walk on $H(G)$ is equivalent to a random walk on G.

4.3 Detailed Balance

We introduce the idea of detailed balance, which is crucial to proving Theorems 4.4 and 4.5.

Definition 4.6

Let $M = (S, P)$ be a Markov chain with state space S and transition probabilities P. We say M satisfies **detailed balance** if there exists a probability mass function π [a] such that

$$\pi_x P_{x,y} = \pi_y P_{y,x}$$

for all states $x, y \in S$.

[a] i.e. $\sum_{v \in V} \pi_v = 1$

It is a well-known result that if such a π exists, then π must be a stationary distribution of M. Thus, π is unique if M is irreducible. Another well-known result is Kolmogorov's criterion, which gives necessary and sufficient conditions for a Markov chain to satisfy detailed balance.

Theorem 4.7 (Kolmogorov's Criterion)

Let $M = (S, P)$ be a Markov chain. M satisfies detailed balance if and only if

$$P_{j_1,j_2} P_{j_2,j_3} \cdots P_{j_{n-1},j_n} = P_{j_n,j_{n-1}} P_{j_{n-1},j_{n-2}} \cdots P_{j_2,j_1} \tag{1}$$

for all finite sequences of states $j_1, \ldots, j_n \in S$,

The following theorem shows that the detailed balance condition characterizes exactly the Markov chains induced by undirected (simple) graphs, and is precisely why detailed balance is so important to us.

Theorem 4.8

Suppose $M = (S, P)$ is an irreducible Markov chain that satisfies detailed balance. Then, there exists an undirected, weighted (simple) graph G on S such that M and the simple random walk on G are equivalent. Furthermore, the weights of G are given by $w_{x,y} = \pi_x P_{x,y}$.

Conversely, if G is a connected, undirected, weighted (simple) graph on S, then a simple random walk on G satisfies detailed balance.

Proof. First, suppose $M = (S, P)$ is an irreducible Markov chain that satisfies detailed balance. Let π be the stationary distribution of M. Create a graph G on S, where the edge between states x and y has

weight

$$w_{x,y} = \pi_x P_{x,y}.$$

By detailed balance, this is a well-defined construction. Then, in a simple random walk on G, the probability of going from x to y in one time-step is

$$\frac{w_{x,y}}{\sum_{z\in S} w_{x,z}} = \frac{\pi_x P_{x,y}}{\sum_{z\in S} \pi_x P_{x,z}} = \frac{P_{x,y}}{\sum_{z\in S} P_{x,z}} = P_{x,y},$$

since $\sum_{z\in S} P_{x,z} = 1$. Note that, because M is irreducible, $\pi_x \neq 0$ for all states x.

Conversely, suppose G is a connected, undirected, weighted (simple) graph on S, with edge weights $w_{x,y}$. Since G is connected, the unique stationary distribution of G is

$$\pi_x = \sum_{y\in S} w_{x,y}.$$

Abusing notation, let $P_{x,y}$ be the probability of going from x to y in a simple random walk on G. Then,

$$\pi_x P_{x,y} = \left(\sum_{y\in S} w_{x,y}\right) \cdot \frac{w_{x,y}}{\sum_{z\in S} w_{x,z}} = w_{x,y}.$$

By symmetry, $\pi_y P_{x,y} = w_{y,x} = w_{x,y}$. Thus, G satisfies detailed balance. □

Corollary 4.9

Suppose $M = (S, P)$ is an irreducible Markov chain that satisfies detailed balance. Let G be an undirected, weighted, simple graph on S such that M and a simple random walk on G are equivalent. Then, there exists a constant c such that the weights of G are $w_{x,y} = c\pi_x P_{x,y}$.

Proof. At each vertex v, for the random walks to be equivalent, there must exist a constant c_v such that $w_{v,u} = c_v \pi_v P_{v,u}$ for all vertices u. But since M is irreducible, the c_v must all be equal. □

4.4 Proofs of Main Theorems

Using Theorem 4.8, we can prove Theorems 4.4 and 4.5.

Proof of Theorem 4.4. It suffices to show that the simple random walk on H satisfies detailed balance. Define the probability mass function π by

$$\pi_v = c \sum_{e\in N(v)} w(e),$$

where $N(v)$ is the set of hyperedges incident to v, and $c > 0$ is a normalizing constant. Let $P_{v,w}$ be the probability of going from v to w in a simple random walk on H. Then,

$$\begin{aligned}\pi_v P_{v,w} &= \left(\sum_{e \in N(v)} w(e)\right)\left(\sum_{e \in N(v)} \left(\frac{w(e)}{\sum_{e \in N(v)} w(e)}\right) \frac{\mathbb{1}(w \in e)}{|e| - 1}\right) \\ &= \sum_{e \in N(v)} \left(w(e) \cdot \frac{\mathbb{1}(w \in e)}{|e| - 1}\right) \\ &= \sum_{e \in E} \left(\frac{w(e)}{|e| - 1} \cdot \mathbb{1}(v, w \in e)\right).\end{aligned}$$

Thus, by symmetry, $\pi_v P_{v,w} = \pi_w P_{w,v}$. □

Proof of Theorem 4.5. For each vertex $v \in V$, define $\lambda(v) = c_e \lambda_e(v)$ for some hyperedge e incident to v. By assumption, $\lambda(v)$ is well-defined.

It suffices to show that the lazy random walk on H satisfies detailed balance. By Kolmogorov's criterion, we need to show

$$p_{v_1,v_2} p_{v_2,v_3} \cdots p_{v_{n-1},v_n} p_{v_n,v_1} = p_{v_1,v_n} p_{v_n,v_{n-1}} \cdots p_{v_2,v_1}.$$

for any sequence of vertices $v_1, \ldots, v_n$. Define $v_{n+1} = v_1$, $v_0 = v_n$, and $E_i = N(v_i) \cap N(v_{i+1})$. Then,

$$\begin{aligned}p_{v_1,v_2} p_{v_2,v_3} \cdots p_{v_{n-1},v_n} p_{v_n,v_1} &= \prod_{i=1}^{n} \left(\sum_{e \in E_i} \frac{\omega(e)}{\sum_{f \in E_i} \omega(f)} \cdot \frac{\lambda_e(v_{i+1})}{\sum_{w \in e} \lambda_e(w)}\right) \\ &= \prod_{i=1}^{n} \left(\sum_{e \in E_i} \frac{\omega(e)}{\sum_{f \in E_i} \omega(f)} \cdot \frac{c_e \lambda_e(v_{i+1})}{\sum_{w \in e} c_e \lambda_e(w)}\right) \\ &= \prod_{i=1}^{n} \left(\sum_{e \in E_i} \frac{\omega(e)}{\sum_{f \in E_i} \omega(f)} \cdot \frac{\lambda(v_{i+1})}{\sum_{w \in e} \lambda(w)}\right) \\ &= \left(\prod_{i=1}^{n} \lambda(v_{i+1})\right) \cdot \prod_{i=1}^{n} \left(\sum_{e \in E_i} \frac{\omega(e)}{\sum_{f \in E_i} \omega(f)} \cdot \frac{1}{\sum_{w \in e} \lambda(w)}\right) \\ &= \left(\prod_{i=1}^{n} \lambda(v_i)\right) \cdot \prod_{i=1}^{n} \left(\sum_{e \in E_i} \frac{\omega(e)}{\sum_{f \in E_i} \omega(f)} \cdot \frac{1}{\sum_{w \in e} \lambda(w)}\right).\end{aligned}$$

Similarly,

$$\begin{aligned}
p_{v_1,v_n}p_{v_n,v_{n-1}}\cdots p_{v_2,v_1} &= \prod_{j=1}^{n}\left(\sum_{e\in E_i}\frac{\omega(e)}{\sum_{f\in E_i}\omega(f)}\cdot\frac{\lambda_e(v_{i-1})}{\sum_{w\in e}\lambda_e(w)}\right)\\
&= \prod_{j=1}^{n}\left(\sum_{e\in E_i}\frac{\omega(e)}{\sum_{f\in E_i}\omega(f)}\cdot\frac{\lambda(v_{i-1})}{\sum_{w\in e}\lambda(w)}\right)\\
&= \left(\prod_{i=1}^{n}\lambda(v_{i-1})\right)\cdot\prod_{i=1}^{n}\left(\sum_{e\in E_i}\frac{\omega(e)}{\sum_{f\in E_i}\omega(f)}\cdot\frac{1}{\sum_{w\in e}\lambda(w)}\right)\\
&= \left(\prod_{i=1}^{n}\lambda(v_i)\right)\cdot\prod_{i=1}^{n}\left(\sum_{e\in E_i}\frac{\omega(e)}{\sum_{f\in E_i}\omega(f)}\cdot\frac{1}{\sum_{w\in e}\lambda(w)}\right).
\end{aligned}$$

Thus, Kolmogorov's criterion holds, so the lazy random walk on H satisfies detailed balance. □

4.5 Discussion

4.5.1 Problem 1

In all of our results, the vertex weights $\lambda_e(v)$ are quite structured. This may not be true in general, and in that case, it is unlikely that H satisfies detailed balance (e.g. Counterexample 1). Indeed, basic simulations suggest that for "random" hypergraphs H with integer vertex weights, both the simple and lazy random walks do not satisfy detailed balance.

If the vertex weights $\lambda_e(v)$ are nontrivial, checking whether the simple/lazy random walk on H satisfies detailed balance becomes a lot harder. Kolmogorov's condition is impractical, since one cannot check (1) for all finite sequences of vertices. Thus, one would need to compute the stationary distribution π of the random walk, which requires solving a linear system of equations, and then verify that $\pi_x p_{x,y} = \pi_y p_{y,x}$ for all pairs of states x, y.

Below is another special case for when a Markov chain satisfies detailed balance. We omit the proof, as it follows straight from the definition of detailed balance.

Theorem 4.10

Let $M = (S, P)$ be a finite-state Markov chain. Suppose the following two conditions hold:

1. for some state z, $P_{x,z} > 0$ for all states $x \neq z$, and
2. $P_{x,y}P_{y,z}P_{z,x} = P_{x,z}P_{z,y}P_{y,x}$ for all states x, y, z.

Then, M satisfies detailed balance, with stationary distribution

$$\pi_x = \begin{cases} c\frac{P_{z,x}}{P_{x,z}} & \text{for } x \neq z, \\ c & \text{for } x = z, \end{cases}$$

where $c > 0$ is a normalizing factor.

So if H is a "star hypergraph", in the sense that there exists a vertex v that is connected to every other vertex w via some hyperedge, and H satisfies a simplified version of Kolmogorov's criterion, then there exist edge weights $w_{u,v}$ on $G(H)$ such that a simple/lazy random walk on H is equivalent to a random walk on $G(H)$.

4.5.2 Problem 2

For Problem 2, because we don't have any nice structure like in Problem 1 with detailed balance, and because our probability transition functions are nonlinear in $\omega(e)$ and $\lambda_e(v)$, it is harder to say much. If we combine Corollary 4.9 and Theorem 4.4, we get the following.

Proposition 4.11

Let G be a simple graph with weights $w_{u,v}$. Suppose there exist constants $\widetilde{\omega}(e)$ for each hyperedge $e \in H(G)$ such that

$$w_{u,v} = \sum_{e \in N(u) \cap N(v)} \frac{\widetilde{\omega}(e)}{|e| - 1}. \tag{2}$$

Then, the simple random walk on $H(G)$, with hyperedge weights $\widetilde{\omega}(e)$ and vertex weights $\lambda_e(v) = 1$, is equivalent to the simple random walk on G.

Note that the system of equations in (2) has $|E(G)|$ equations and $|E(H(G))|$ variables, where $|E(G)|$ is the number of edges of G, and $|E(H(G))|$ is the number of hyperedges in $H(G)$. It is easy to see that $|E(G)| \geq |E(H(G))|$, so in general, we do not expect this system of equations to be solvable. Thus, in most cases, given a simple graph G, for simple random walks on G and $H(G)$ to be equivalent, we need non-trivial vertex weights on $H(G)$, i.e. we cannot have $\lambda_e(v)$ be independent of v. (Note that the same reasoning also applies to lazy random walks on $H(G)$ if we replace $|e| - 1$ with $|e|$ in (2).)

5 Applications to Disease-Gene Prioritization

5.1 Method

Disease-gene prioritization is the process of identifying genes associated with a specific disease from a larger list of "candidate genes." The idea is that, rather than having biologists carefully study all 300 candidate genes, computational efforts can narrow the list down to a more reasonable number, like 5 or 10. To determine which genes are more likely, we use a gene interaction network. The results of Köhler [21] show that kernel-based methods outperform more naïve methods, e.g. methods based on

the shortest paths in the network, because kernel-based methods take into account the topology of the whole network.

To mimic the process of disease-gene prioritization, we follow Köhler's framework. Let K be a similarity kernel on V, the vertices of G, where $K_{i,j} \in [0,1]$ is the similarity between gene i and gene j. Köhler gives a list of around 100 diseases and known disease-genes for three different disease classes: monogenic, polygenic, and cancer. Represent each disease d as a list of disease-genes. For each disease d and disease-gene $g \in d$:

- Collect the 100 genes closest to g (including g), with the metric being genomic distance. These 100 genes comprise our list of candidate genes L.
- Pretend g is not associated with disease d, and let $d' = d \setminus \{g\}$.
- For each gene $h \in L$, compute the score of h, where

$$\text{score}(h) \stackrel{\Delta}{=} \sum_{i \in d'} K_{i,h}.$$

- Rank genes in L according to their score, and return the rank of g.

The rank of g is an integer between 1 and 100, inclusive. Ideally, g would have rank 1, since it is known to be associated with disease d.

This above framework associates a list of ranks with each disease d. Using this, we can get an "average rank" for each disease d; averaging these over each disease in a disease class gives us the average rank for each disease class.

5.2 Personalized PageRank Kernel

The above framework is, for the most part, independent of the network G. The relevant information from G is encoded in the similarity kernel K. We will build our kernel by simulating a random walk with restart on G, also known as PageRank. Our derivation follows Raphael [25].

Let W be the probability transition matrix of a simple random walk on G (recall $W = D^{-1}A$, where A is the adjacency matrix of G, and D is the degree matrix of G). Let v_t be the vertex we are visiting at time t. Let $r \in (0,1)$.

To choose v_{t+1}, we perform the *random walk with restart*. With probability $1 - r$, let v_{t+1} be a neighbor of v_t, chosen uniformly at random, and with probability r, jump to a vertex according to some fixed distribution f. If p_t be the vertex distribution of the random walk at time t, then we can express this process as

$$p_{t+1} = rf + (1-r)p_t W.$$

The stationary distribution of our walk, p, satisfies

$$p = rf + (1-r)pW \implies p = rf(I - (1-r)W)^{-1}.$$

Thus, the PageRank kernel K is given by

$$K \triangleq r(I - (1-r)W)^{-1}.$$

By looking at eigenvalues, it can be shown that $I - (1-r)W$ is invertible for $r \in (0,1)$, so K is always well-defined.

5.3 Results

Let G be the ReactomeFI network, version 2016 [13, 18]. In addition to giving us a list of gene interactions, the ReactomeFI network also provides annotations for each edge $e = (u, v)$. For example, some of the annotations are: whether gene u is catalyzed by gene v, whether genes u and v are in a common complex, or whether the interaction between u and v is a predicted interaction and has not actually been observed. (There is also a score for each interaction, so we filter out predicted interactions with a low score before we create our network.)

Let E' be all edges $e = (u, v)$ such that u and v are in a protein complex with each other, and let G' be the graph induced by edges E'. We build our (undirected) hypergraph H in the following way. First initialize H to be G. Then, for each maximal clique $\{v_1, v_2, ..., v_n\}$ in G', add a hyperedge $e = \{v_1, ..., v_n\}$. Finally, remove all (simple) edges from all maximal cliques of G'.

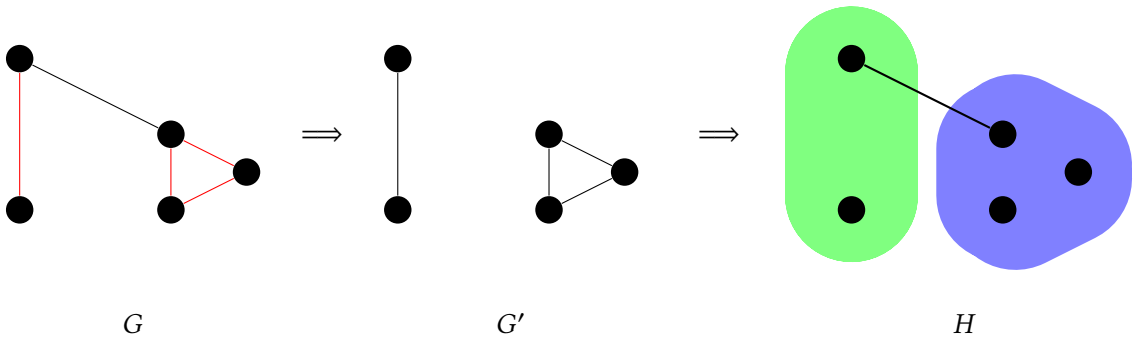
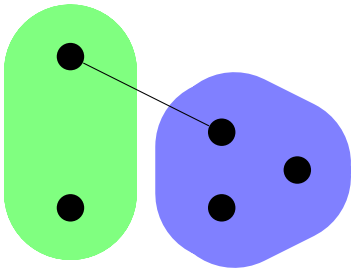


Figure 1: Example of how to convert G to H. Red edges denote protein complexes.

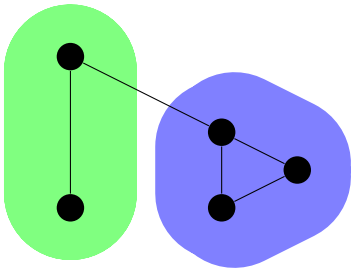
We can write this more succinctly:

$$H \triangleq (G \setminus G') \cup H(G'). \tag{3}$$

We also consider the hypergraph $H' \triangleq G \cup H(G')$. The difference between H and H' is that, for each maximal clique $\{v_1, ..., v_n\}$ in G', we leave in simple edges between each v_i and v_j. So in H', there is a simple clique inside each hyperedge.



H



H'

Below are some summary statistics of G, H, H'.

	Number of Vertices	Number of Edges	Number of Hyperedges
G	10,495	195,081	0
H	10,495	116,663	466,671
H'	10,495	195,081	466,671

We compute the PageRank kernel ($r = 0.4$) for the simple random walk on G, as well as for four different types of hypergraph simple random walks:

- the unweighted simple random walk on H,
- the uniform edge-weighted simple random walk on H,
- the unweighted simple random walk on H', and
- the uniform edge-weighted simple random walk on H'.

For each PageRank kernel, we find the rankings for each gene in each disease, and use this to compute the average rank for each disease class. Our list of diseases and disease-genes is the same one used by Köhler [21]. (Recall that rankings are between 1 and 100, inclusive, and a lower rank is better.)

We summarize our results in Figure 2 below.

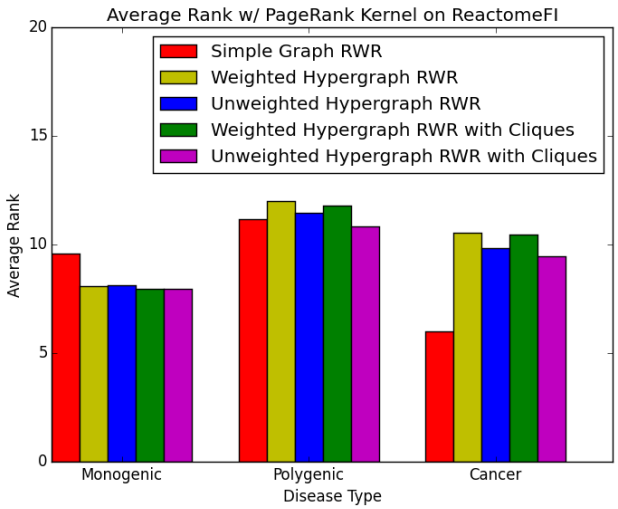


Figure 2: Average rank for each disease class using the ReactomeFI network.

The "Hypergraph RWR with Cliques" refers to walks on H', since it has cliques in addition to each hyperedge, while the "Weighted Hypergraph RWR" refers to the uniform edge-weighted simple random walk.

5.3.1 Degree Distribution in Simple Graph

In the simple network G, the average rank for cancer diseases is much lower than for monogenic and polygenic diseases. One would expect the situation to be reversed, since determining the mechanisms and pathways of cancer is still a very active and underdeveloped area of research. However, the reason for this behavior is because of the network itself—in the ReactomeFI network, cancer disease-genes tend to have a higher degree than other disease-genes.

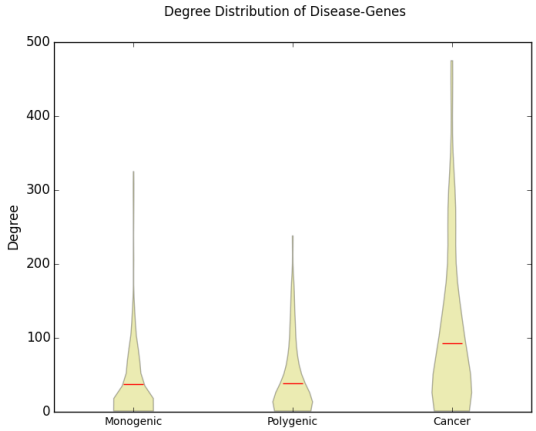


Figure 3: Degree distribution of disease-genes in different disease classes.

This is not caused by something inherent to cancer genes: it is simply because cancer is studied more than other diseases. Researchers study cancer genes more often, so they will discover more interactions involving cancer genes than those involving non-cancer genes.

Connecting this observation to our results, we note that, in random walk models, vertices with higher degree naturally have higher similarities. Intuitively, suppose v_1, v_2 are vertices, and v_1 has a much higher degree than v_2. If w is an arbitrary vertex, then the probability of going from w to v_1 is, in most cases, going to be higher than the probability of going from w to v_2, so one would expect $K[w, v_1] > K[w, v_2]$.

5.3.2 Simple Graphs versus Hypergraphs

When comparing the simple graph results with the hypergraph results in each disease class, we see that, in general, the hypergraph has better performance than the simple graph with monogenic diseases; has about the same performance with polygenic diseases; and has worse performance with cancer diseases. We suspect the worse performance with cancer diseases is because cancer genes are in many more maximal cliques than other disease-genes, indicating that there are too many artificially-created hyperedges of cancer genes.

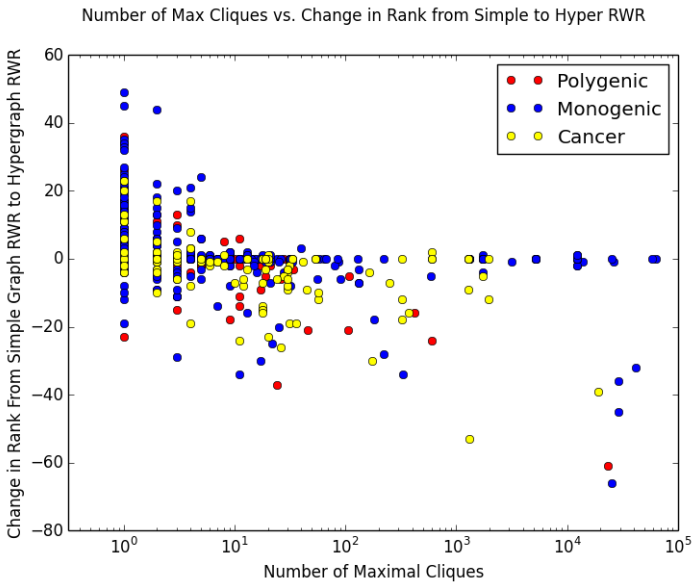


Figure 4: Change in rank when going from simple graph kernel to hypergraph kernel, versus number of maximal cliques of G' that contain the gene.

In Figure 4, for each disease-gene g, we look at the change in rank of g when going from the simple graph kernel to a hypergraph kernel, versus the number of maximal cliques of G' that contain g (i.e. the number of hyperedges containing g). We adopt the convention that a positive change in rank means that the rank of the disease-gene in simple graph G is greater than its rank in the hypergraph H. This corresponds to the notion that a positive change in rank is "good", while a negative change in rank is

"bad".

We see that genes in 10 or more maximal cliques—that is, genes that are in 10 or more hyperedges in H—almost never have a positive change in rank. This seems reasonable, since our hyperedges are supposed to be protein complexes, and most genes won't be in more than 10 protein complexes. Thus, it is likely that we are adding too many artificial connections in the network, obfuscating the true relationships between vertices.

	Average Number of Max Cliques
Monogenic	13.794
Polygenic	5.923
Cancer	18.454

From the above table, we see that cancer genes are in more hyperedges than other disease-genes. (We use the geometric mean, rather than the arithmetic mean, to mitigate the effect of large-degree outliers.)

It seems that there is a "sweet spot" for the number of hyperedges we include. If we include too many hyperedges, then that artificially creates similarity between vertices in the same hyperedge, and makes it harder to detect similar vertices not in the same hyperedge. On the other hand, if we don't add enough hyperedges, then the hypergraph network does not look that different from the simple graph network, so we will get similar results from either network.

5.3.3 Different Types of Hypergraphs

It is interesting to compare the different types of hypergraph kernels as well. In Figure 2, each of the hypergraph kernels performs more or less the same, although generally the unweighted simple random walk kernels perform better than the weighted ones.

To compare the different kernels, we do the following. First, we generate a random graph G using the preferential attachment mechanism [3] on $n = 100$ vertices. Then, we generate the maximal hypergraph associated to G, $H(G)$, which we use to construct each of the four hypergraph kernels (weighted, unweighted, weighted with cliques, unweighted with cliques). Note that, by the nature of the preferential attachment model, the vertices with low numbers are also the vertices with the highest degree—the "hubs" of the graph.

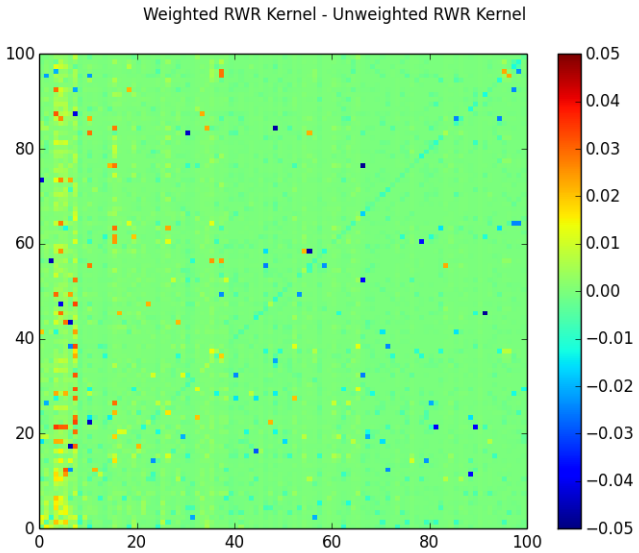


Figure 5: Weighted hypergraph kernel minus unweighted hypergraph kernel.

In Figure 5, compared to the unweighted kernel, we see that the weighted kernel assigns higher similarity between vertices and hubs. (Note that the PageRank kernel is asymmetric, so in this case, we are specifically talking about the similarity $K[v, \text{hub vertex}]$.) Because of this, we hypothesize that it is harder to detect connections between non-hub vertices using the weighted kernel, since the non-hub vertices have low similarity with one another.

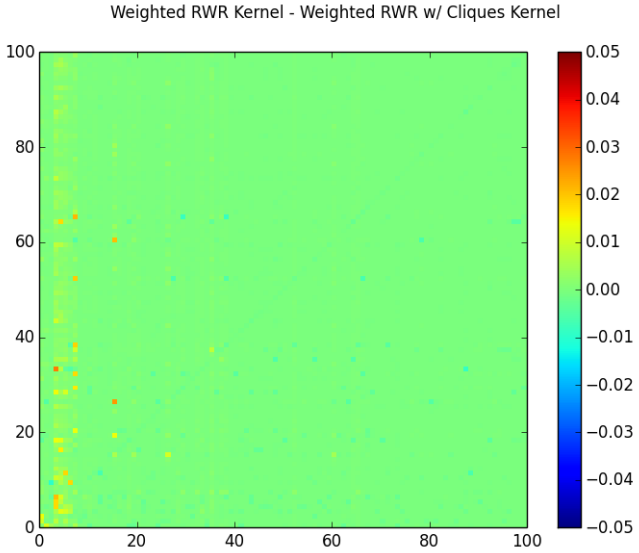


Figure 6: Weighted hypergraph kernel minus weighted hypergraph with cliques kernel.

We see a similar pattern above. The weighted kernel assigns more weight to hubs of the graph, compared to the weighted kernel with cliques.

6 Conclusion

In this thesis, we gave a general framework for hypergraphs and random walks on hypergraphs. We showed that random walks on hypergraphs are conceptually distinct from random walks on simple graphs, in the sense that, given a hypergraph, there does not always exist a simple graph with an equivalent random walk, and vice-versa. We also applied our theory of random walks on hypergraphs to disease-gene prioritization via the PageRank algorithm. We found that, compared to simple graphs, hypergraphs achieved better results for monogenic diseases, did around the same for polygenic ones, and had worse results for cancer.

Below, we list a few different areas to explore in the future.

6.1 Creating a Different Hypergraph

One possible reason for why our hypergraph model did worse with cancer genes than the simple graph model was that, in our hypergraph, cancer genes are in more hyperedges than other disease-genes. This observation motivates the following problem.

Problem 3

Given a simple graph $G = (V, E)$, find a hypergraph H on V such that

- H **preserves the connectivity** of G (that is, for any vertices $u, v \in V$, there is an edge between u and v in G if and only if there is a hyperedge in H that contains both u and v), and
- H has the **minimal number of hyperedges** of all hypergraphs that satisfy the previous property.

If we regard hyperedges in H as cliques in G, then a hypergraph H that satisfies the above two properties is known as the **minimum edge clique cover** of G. Unfortunately, finding the minimum edge clique cover of a graph is NP-hard. Moreover, there are no fast algorithms for finding such a covering in graphs of comparable size to ReactomeFI [19, 14].

Ennis [17] gives a more practical algorithm for finding "assignment-minimal edge clique coverings", where an assignment-minimal edge clique covering is an edge clique covering C such that, for all vertices v, there is no edge clique subcovering of C where v is in fewer cliques. It would be interesting to perform our disease-gene prioritization process by using Ennis's algorithm to build a hypergraph from G' (the graph induced by "protein complex" edges), instead of using $H(G')$, and see how the results change. Another possible approach to building our hypergraph could be to use a graph clustering

algorithm, e.g. HotNet2 [22] or Markov Clustering [30], to cluster G', and then making the clusters our hyperedges.

6.2 Alternative Versions of Problems 1 and 2

In Problem 1, we want to find weights $w_{u,v}$ on $G(H)$ such that P^H, the probability transition matrix of a random walk on H, is *equal* to $P^{G(H)}$. However, we could also investigate a relaxed version of this problem, where we instead want to minimize the (Frobenius, max, L^1, etc.) norm of $P^H - P^{G(H)}$. (We can also ask a similar question for Problem 2.)

Another problem related to Problem 2 is the following optimization problem.

Problem 4

Given a simple graph G with weights $w_{u,v}$, find the smallest hypergraph H (in terms of number of hyperedges) such that

- H preserves the connectivity of G, and
- there exist hyperedge weights $\omega(e)$ and vertex weights $\lambda_e(v)$ on H such that a (simple/lazy) random walk on H is equivalent to the (weighted) random walk on G.

Since we can take $H = G$, a hypergraph satisfying these two properties always exists. We suspect this problem is NP-hard, via a reduction to the minimum edge clique cover problem. However, a proof of this currently eludes us.

7 Acknowledgements

First and foremost, I'm incredibly grateful to Ben Raphael for mentoring me over the past two years. Maybe it's because Ben and I both studied math, but the projects Ben recommends to me, and the directions he leads me in, are always really interesting. I'm glad I worked on this thesis with Ben—doing this thesis has solidified my intent to apply to PhD programs next year.

I'd also like to thank Matt Reyna for helping me with whatever questions I had. I'm especially grateful for his advice to use Julia, which ended up being a really good choice. Thanks also to Jasper Lee, who has always been willing to listen to (and provide his far superior opinion on) whatever I'm thinking about at the moment, and to Paul Valiant, for agreeing to be my reader. Other people I've entertained with stories of hypergraphs, and struggled through this year-long journey with, include Keshav Vemuri and David Liu.

I'd be nothing without my friends, both at Brown and at home, who have always encouraged me and pushed me to be a better person. It's hard to name everyone, but I think (or rather, I hope) everyone who I'm referencing in the previous sentence knows who they are. A special thanks to Gina, who has always been there for me, and whose intelligence and drive continue to amaze me.

Finally, this thesis wouldn't have been possible without my family. Between the countless trips to libraries and museums, the boxes of Amar Chitra Katha books they'd make me read, and the Art of Problem Solving books and classes they continued to buy for me, my parents deserve all the credit for sharing with me their love of learning and their intellectual curiosity, and for shaping me into the person I am today. And through our countless discussions on math—usually about the most recent brainteaser one of us heard—my brother also deserves much of the credit for cultivating my interest in math and problem-solving. I'm eternally grateful for the love and support of my family.

References

[1] S. Agarwal, K. Branson, and S. Belongie. Higher order learning with graphs. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 17–24, New York, NY, USA, 2006. ACM.

[2] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie. Beyond pairwise clustering. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 838–845, Washington, DC, USA, 2005. IEEE Computer Society.

[3] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.

[4] C. Avin, Y. Lando, and Z. Lotker. Radio cover time in hyper-graphs. *Ad Hoc Networks*, 12:278 – 290, 2014.

[5] P. Bailey et al. Genomic analyses identify molecular subtypes of pancreatic cancer. *Nature*, 531(7592):47–52, 03 2016.

[6] A. Bellaachia and M. Al-Dhelaan. Random walks in hypergraph. In *Proceedings of the 2013 International Conference on Applied Mathematics and Computational Methods, Venice Italy*, pages 187–194, 2013.

[7] P. Berkhin. Bookmark-coloring algorithm for personalized pagerank computing. *Internet Math.*, 3(1):41–62, 2006.

[8] K. Berlt, E. S. de Moura, A. Carvalho, M. Cristo, N. Ziviani, and T. Couto. Modeling the web as a hypergraph to compute page reputation. *Inf. Syst.*, 35(5):530–543, July 2010.

[9] T. H. Chan, A. Louis, Z. G. Tang, and C. Zhang. Spectral properties of hypergraph Laplacian and approximation algorithms. *CoRR*, abs/1605.01483, 2016.

[10] F. Chung. The Laplacian of a hypergraph. *Expanding graphs (DIMACS series)*, pages 21–36, 1993.

[11] F. Chung. The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences*, 104(50):19735–19740, 2007.

[12] C. Cooper, A. Frieze, and T. Radzik. The cover times of random walks on random uniform hypergraphs. *Theoretical Computer Science*, 509:51 – 69, 2013.

[13] D. Croft, A. F. Mundo, R. Haw, M. Milacic, J. Weiser, G. Wu, M. Caudy, P. Garapati, M. Gillespie, M. R. Kamdar, B. Jassal, S. Jupe, L. Matthews, B. May, S. Palatnik, K. Rothfels, V. Shamovsky, H. Song, M. Williams, E. Birney, H. Hermjakob, L. Stein, and P. D'Eustachio. The reactome pathway knowledgebase. *Nucleic Acids Research*, 42(Database issue):D472–D477, 01 2014.

[14] M. Cygan, M. Pilipczuk, and M. Pilipczuk. Known algorithms for edge clique cover are probably optimal. *SIAM Journal on Computing*, 45(1):67–83, 2016.

[15] J. Das and H. Yu. Hint: High-quality protein interactomes and their applications in understanding human disease. *BMC Systems Biology*, 6(1):92, 2012.

[16] A. Ducournau and A. Bretto. Random walks in directed hypergraphs and application to semi-supervised image segmentation. *Computer Vision and Image Understanding*, 120:91 – 102, 2014.

[17] J. M. Ennis, C. M. Fayle, and D. M. Ennis. Assignment-minimum clique coverings. *J. Exp. Algorithmics*, 17:1.5:1.1–1.5:1.17, July 2012.

[18] A. Fabregat, K. Sidiropoulos, P. Garapati, M. Gillespie, K. Hausmann, R. Haw, B. Jassal, S. Jupe, F. Korninger, S. McKay, L. Matthews, B. May, M. Milacic, K. Rothfels, V. Shamovsky, M. Webber, J. Weiser, M. Williams, G. Wu, L. Stein, H. Hermjakob, and P. D'Eustachio. The reactome pathway knowledgebase. *Nucleic Acids Research*, 44(Database issue):D481–D487, 01 2016.

[19] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Data reduction and exact algorithms for clique cover. *J. Exp. Algorithmics*, 13:2:2.2–2:2.15, Feb. 2009.

[20] G. Kar, A. Gursoy, and O. Keskin. Human cancer protein-protein interaction network: A structural perspective. *PLOS Computational Biology*, 5(12):1–18, 12 2009.

[21] S. Köhler, S. Bauer, D. Horn, and P. N. Robinson. Walking the interactome for prioritization of candidate disease genes. *American Journal of Human Genetics*, 82(4):949–958, 04 2008.

[22] M. D. M. Leiserson, F. Vandin, H.-T. Wu, et al. Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes. *Nat Genet*, 47(2):106–114, 02 2015.

[23] J. Orlin. Contentment in graph theory: Covering graphs with cliques. *Indagationes Mathematicae (Proceedings)*, 80(5):406 – 424, 1977.

[24] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia, 1998.

[25] B. J. Raphael. Lecture notes in CSCI 2950-C: Algorithms for cancer genomics. http://cs.brown.edu/courses/csci2950-c/Spring2015/index.html, March 2015.

[26] S. Razick, G. Magklaras, and I. M. Donaldson. irefindex: A consolidated protein interaction database with provenance. *BMC Bioinformatics*, 9(1):405, 2008.

[27] A. Ritz, A. N. Tegge, H. Kim, C. L. Poirel, and T. Murali. Signaling hypergraphs. *Trends in Biotechnology*, 32(7):356 – 362, 2014.

[28] L. Sun, S. Ji, and J. Ye. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 668–676, New York, NY, USA, 2008. ACM.

[29] Z. Tian, T. Hwang, and R. Kuang. A hypergraph-based learning algorithm for classifying gene expression and arraycgh data with prior knowledge. *Bioinformatics*, 25(21):2831, 2009.

[30] S. Van Dongen. Graph clustering via a discrete uncoupling process. *SIAM J. Matrix Anal. Appl.*, 30(1):121–141, Feb. 2008.

[31] G. Wu, X. Feng, and L. Stein. A human functional protein interaction network and its application to cancer data analysis. *Genome Biology*, 11(5):R53–R53, 2010.

[32] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems 19*, pages 1601–1608, Cambridge, MA, USA, Sept. 2007. Max-Planck-Gesellschaft, MIT Press.