Random Walks on Hypergraphs with Edge-Dependent Vertex Weights

Uthsav Chitra [1] **Benjamin J Raphael** [1]

Abstract

Hypergraphs are used in machine learning to model higher-order relationships in data. While spectral methods for graphs are well-established, spectral theory for hypergraphs remains an active area of research. In this paper, we use random walks to develop a spectral theory for *hypergraphs with edge-dependent vertex weights*: hypergraphs where every vertex v has a weight $\gamma_e(v)$ for each incident hyperedge e that describes the contribution of v to the hyperedge e. We derive a random walk-based hypergraph Laplacian, and bound the mixing time of random walks on such hypergraphs. Moreover, we give conditions under which random walks on such hypergraphs are equivalent to random walks on graphs. As a corollary, we show that current machine learning methods that rely on Laplacians derived from random walks on hypergraphs with edge-*independent* vertex weights do not utilize higher-order relationships in the data. Finally, we demonstrate the advantages of hypergraphs with edge-dependent vertex weights on ranking applications using real-world datasets.

1. Introduction

Graphs are ubiquitous in machine learning, where they are used to represent pairwise relationships between objects. For example, social networks, protein-protein interaction (PPI) networks, and the internet are modeled with graphs. One limitation of graph models, however, is that they do not encode higher-order relationships between objects. A social network can represent a community of users (e.g. a friend group) as a collection of edges between each user, but this pairwise representation loses information about the overall group structure (Yang et al., 2017). In biology, protein interactions are not only between pairs of proteins, but also between groups of proteins in protein complexes (Ramadan et al., 2004; Ritz et al., 2014).

Such higher-order interactions can be modeled using a hypergraph: a generalization of a graph containing hyperedges that can be incident to more than two nodes. A hypergraph representation of a social network can model a community of friends with a single hyperedge. In contrast, the corresponding representation of a community in a graph requires many edges that connect pairs of individuals within the community; conversely, it may not be clear which collection of edges in a graph represents a community (e.g. a clique, an edge-dense subnetwork, etc). Hypergraphs have been used in a variety of machine learning tasks, including clustering (Agarwal et al., 2005; Zhou et al., 2006; Li & Milenkovic, 2017; 2018), ranking keywords in a collection of documents (Bellaachia & Al-Dhelaan, 2013), predicting customer behavior in e-commerce (Li et al., 2018), object classification (Zhang et al., 2018a;b), and image segmentation (Kim et al., 2011).

A common approach to incorporate graph information in a machine learning algorithm is to utilize properties of random walks or diffusion processes on the graph. For example, random walks on graphs underlie algorithms for recommendation systems (Jamali & Ester, 2009), clustering (Harel & Koren, 2001; Ng et al., 2001), information retrieval (Brin & Page, 1998), and other applications. In many machine learning applications, the graph is represented through the graph Laplacian. Spectral theory includes many key results regarding the eigenvalues and eigenvectors of the graph Laplacian, and these results form the foundation of spectral learning algorithms.

Spectral theory on hypergraphs is much less developed than on graphs. In seminal work, Zhou et al. (2006) developed learning algorithms on hypergraphs based on random walks on graphs. However, at nearly the same time, Agarwal et al. (2006) showed that the hypergraph Laplacian matrix used by Zhou et al. is equal to the Laplacian matrix of a closely related graph, the star graph. A consequence of this equivalence is that the methods introduced by Zhou et al. utilize only pairwise relationships between objects, rather than the higher-order relationships encoded in the hypergraph. More recently, Chan et al. (2018) and Li & Milenkovic (2017; 2018) developed *nonlinear* Laplacian operators for hyper-

[1]Department of Computer Science, Princeton University, Princeton, NJ, USA. Correspondence to: Benjamin Raphael <braphael@cs.princeton.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

graphs that partially address this issue. However, all existing constructions of linear Laplacian operators utilize only pairwise relationships between vertices, as shown by Agarwal et al. (2006).

In this paper, we develop a spectral theory for hypergraphs with *edge-dependent vertex weights*. In such a hypergraph, each hyperedge e has an edge weight $\omega(e)$, and each vertex v has a collection of vertex weights, with one weight $\gamma_e(v)$ for each hyperedge e incident to v. The edge-dependent vertex weight $\gamma_e(v)$ models the contribution of vertex v to hyperedge e. Edge-dependent vertex weights have previously been used in several applications including: image segmentation, where the weights represent the probability of an image pixel (vertex) belonging to a segment (hyperedge) (Ding & Yilmaz, 2010); e-commerce, where the weights model the quantity of a product (hyperedge) in a user's shopping basket (vertex) (Li et al., 2018); and text ranking, where the weights represent the importance of a keyword (vertex) to a document (hyperedge) (Bellaachia & Al-Dhelaan, 2013). Hypergraphs with edge-dependent vertex weights have also been used in image search (Zeng et al., 2016; Huang et al., 2010) and 3D object classification (Zhang et al., 2018a), where the weights represent contributions of vertices in a k-nearest-neighbors hypergraph.

Unfortunately, because of a lack of a spectral theory for hypergraphs with edge-dependent vertex weights, many of the papers that use these hypergraphs rely on incorrect or theoretically unsound assumptions. For example, Zhang et al. (2018a) and Ding & Yilmaz (2010) use a hypergraph Laplacian with no spectral guarantees, while Li et al. (2018) derive an incorrect stationary distribution for a random walk on such a hypergraph (see Supplement for additional details). The reason such issues arise is because existing spectral methods are developed for hypergraphs with *edge-independent vertex weights*, i.e. hypergraphs where the $\gamma_e(v)$ are identical for all hyperedges e.

In this paper, we derive several results for hypergraphs with edge-dependent vertex weights. First, we show that random walks on hypergraphs with edge-independent vertex weights are *always* equivalent to random walks on the clique graph (Figure 1). This generalizes the results of Agarwal et al. (2006) and gives the underlying reason why existing constructions of hypergraph Laplacian matrices (Rodriguez-Velazquez, 2002; Zhou et al., 2006) do not utilize the higher-order relations of the hypergraph.

Motivated by this result, we derive a random walk-based Laplacian matrix for hypergraphs with edge-dependent vertex weights that utilizes the higher-order relations expressed in the hypergraph structure. This Laplacian matrix satisfies the typical properties one would expect of a Laplacian matrix, including being positive semi-definite and satisfying a Cheeger inequality. We also derive a formula for the stationary distribution of a random walk on a hypergraph with edge-dependent vertex weights, and give a bound on the mixing time of the random walk.

Our paper is organized as follows. In Section 2, we define our notation, and introduce hypergraphs with edge-dependent vertex weights. In Section 3, we formally define random walks on hypergraphs with edge-dependent vertex weights, and show that when the vertex weights are edge-independent, a random walk on a hypergraph has the same transition matrix as a random walk on its clique graph. In Section 4, we derive a formula for the stationary distribution of a random walk, and use it to bound the mixing time. In Section 5, we derive a random-walk based Laplacian matrix for hypergraphs with edge-dependent vertex weights and show some basic properties of the matrix. Finally, in Section 6, we demonstrate two applications of hypergraphs with edge-dependent vertex weights: ranking authors in a citation network and ranking players in a video game. All proofs are in the Supplementary Material.

2. Graphs, Hypergraphs, and Random Walks

Let $G = (V, E, w)$ be a graph with vertex set V, edge set E, and edge weights w. For a vertex v, let $N(v) = \{u \in V : (u, v) \in E\}$ denote the vertices incident to v. The *adjacency matrix* A of a graph is a $|V| \times |V|$ matrix where $A(u, v) = w(e)$ if $(u, v) \in E$ and 0 otherwise.

Let $H = (V, E, \omega)$ be a *hypergraph* with vertex set V; edge set $E \subset 2^V$; and hyperedge weights ω. A graph is a special case of a hypergraph, where each hyperedge e has size $|e| = 2$. For hypergraphs, the terms "hyperedge" and "edge" are used interchangeably. A random walk on a hypergraph is typically defined as follows (Zhou et al., 2006; Ducournau & Bretto, 2014; Cooper et al., 2013; Avin et al., 2014). At time t, a "random walker" at vertex v_t will:

1. Select an edge e containing v_t, with probability proportional to $\omega(e)$.
2. Select a vertex v from e, uniformly at random.
3. Move to vertex $v_{t+1} = v$ at time $t + 1$.

A natural extension is to modify Step 2: instead of choosing v uniformly at random from e, we pick v according to a fixed probability distribution on the vertices in e. This motivates the following definition of a hypergraph with *edge-dependent vertex weights*.

Definition 2.1. *A hypergraph $H = (V, E, \omega, \gamma)$ with* edge-dependent vertex weights *is a set of vertices V, a set $E \subset 2^V$ of hyperedges, a weight $\omega(e)$ for every hyperedge $e \in E$, and a weight $\gamma_e(v)$ for every hyperedge $e \in E$ and every vertex v incident to e.*

We emphasize that a vertex v in a hypergraph with edge-dependent vertex weights has *multiple* weights: one weight

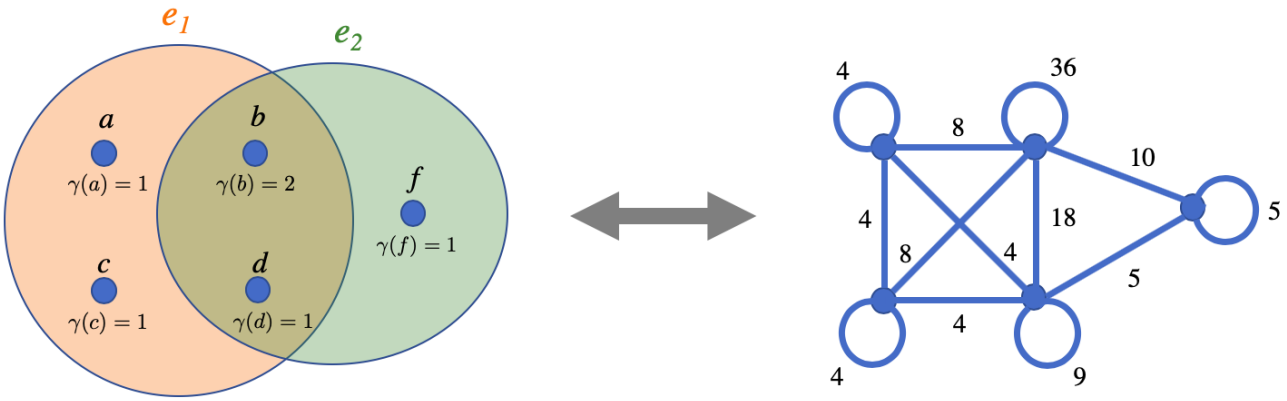

Figure 1. Example illustrating Theorem 3.1. A hypergraph with *edge-independent vertex weights* H (left) and a corresponding edge-weighted clique graph G^H (right) such that random walks on H and G^H are equivalent. Note that, if one changes the vertex weights of b to be edge-*dependent* vertex weights, by setting $\gamma_{e_1}(b) = 1, \gamma_{e_2}(b) = 2$, then it is not possible to choose edge weights $w_{u,v}$ on G^H such that random walks on G^H and H are equivalent.

$\gamma_e(v)$ for each hyperedge e that contains v. Intuitively, $\gamma_e(v)$ measures the contribution of vertex v to hyperedge e. In a random walk on a hypergraph with edge-dependent vertex weights, the random walker will pick a vertex v from hyperedge e with probability proportional to $\gamma_e(v)$. Note that we set $\gamma_e(u) = 0$ if $u \notin e$.

If each vertex has the same contribution to all incident hyperedges, i.e. $\gamma_e(v) = \gamma_{e'}(v)$ for all hyperedges e and e' incident to v, then we say that the hypergraph has *edge-independent vertex weights*, and we use $\gamma(v) = \gamma_e(v)$ to refer to the vertex weights of H. If $\gamma_e(v) = 1$ for all vertices v and incident hyperedges e, we say the vertex weights are *trivial*.

We define $E(v) = \{e \in E : v \in e\}$ to be the hyperedges incident to a vertex v, and $E(u, v) = \{e \in E : u \in e, v \in e\}$ to be the hyperedges incident to both vertices u and v. Let $d(v) = \sum_{e \in E(v)} \omega(e)$ denote the degree of vertex v, and let $\delta(e) = \sum_{v \in e} \gamma_e(v)$ denote the degree of hyperedge e. The *vertex-weight matrix* R of a hypergraph with edge-dependent vertex weights $H = (V, E, \omega, \gamma)$ is an $|E| \times |V|$ matrix with entries $R(e, v) = \gamma_e(v)$, and the *hyperedge weight* matrix W is a $|V| \times |E|$ matrix with $W(v, e) = \omega(e)$ if $v \in e$, and $W(v, e) = 0$ otherwise. The vertex-degree matrix D_V is a $|V| \times |V|$ diagonal matrix with entries $D_V(v, v) = d(v)$, and the hyperedge-degree matrix D_E is a $|E| \times |E|$ diagonal matrix with entries $D_E(e, e) = \delta(e)$.

Given $H = (V, E, \omega, \gamma)$, the *clique graph* of H, G^H, is an unweighted graph with vertices V, and edges $E' = \{(v, w) \subset V \times V : v, w \in e \text{ for some } e \in E\}$. In other words, G^H turns all hyperedges into cliques.

We say a hypergraph H is *connected* if its clique graph G^H is connected. In this paper, we assume all hypergraphs are connected.

For a Markov chain with states S transition probabilities p, we use $p_{u,v}$ to denote the probability of going from state u to state v.

3. Random Walks on Hypergraphs with Edge-Dependent Vertex Weights

Let $H = (V, E, \omega, \gamma)$ be a hypergraph with edge-dependent vertex weights. We first define a random walk on H. At time t, a random walker at vertex v_t will do the following:

1. Pick an edge e containing v, with probability $\omega(e)/d(v)$.
2. Pick a vertex w from e, with probability $\gamma_e(w)/\delta(e)$.
3. Move to vertex $v_{t+1} = w$, at time $t + 1$.

Formally, we define a random walk on H by writing out the transition probabilities according to the above steps.

Definition 3.1. *A random walk on a hypergraph with edge-dependent vertex weights $H = (V, E, \omega, \gamma)$ is a Markov chain on V with transition probabilities*

$$p_{v,w} = \sum_{e \in E(v)} \left(\frac{\omega(e)}{d(v)} \right) \left(\frac{\gamma_e(w)}{\delta(e)} \right). \tag{1}$$

The *probability transition matrix* P of a random walk on H is the $|V| \times |V|$ matrix with entries $P(v, w) = p_{v,w}$ and can be written in matrix form as $P = D_V^{-1} W D_E^{-1} R$. (We use the convention that probability transition matrices have row sum 1.) Using the probability transition matrix P, we can also define a random walk with restart on H (Tong et al., 2006). The random walk with restart is useful when it is unknown whether the random walk is irreducible.

Note that our definition allows self-loops, i.e. $p_{v,v} > 0$, and thus the random walk is lazy. While one can define a non-

lazy random walk (i.e. $p_{v,v} = 0$ for all v), the analysis of such walks is significantly more difficult, as the probability transition matrix cannot be factored as easily. In the Supplement, we show that a weaker version of Theorem 3.1 below holds for a non-lazy random walk. Cooper et al. (2013) also studies the cover time of a non-lazy random walk on a hypergraph with edge-independent vertex weights.

Next, we define what it means for two random walks to be equivalent. Because random walks are Markov chains, we define equivalence in terms of Markov chains.

Definition 3.2. *Let M and N be Markov chains with the same (countable) state space, and let P^M and P^N be their respective probability transition matrices. We say that M and N are **equivalent** if*

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

for all states x and y.

Using this definition, we state our first main theorem: a random walk on a hypergraph with *edge-independent vertex weights* is equivalent to a random walk on its clique graph, for some choice of weights on the clique graph.

Theorem 3.1. *Let $H = (V, E, \omega, \gamma)$ be a hypergraph with edge-*independent *vertex weights. There exist weights $w_{u,v}$ on the clique graph G^H such that a random walk on H is equivalent to a random walk on G^H.*

Theorem 3.1 generalizes the result by Agarwal et al. (2006) who showed that the two hypergraph Laplacian matrices constructed in Zhou et al. (2006) and Rodriguez-Velazquez (2002) are equal to the Laplacian matrix of either the clique graph or the star graph, another graph constructed from a hypergraph. Agarwal et al. (2006) also showed that the Laplacians of the clique graph and the star graph are equal when H is k-uniform (i.e. when all hyperedges have size k), and are very close otherwise. Since the Laplacian matrices in Zhou et al. (2006) and Rodriguez-Velazquez (2002) are derived from random walks on edge-independent vertex weights, Theorem 3.1 implies that both Laplacians are equal to the Laplacian of the clique graph – even when the hypergraph is not k-uniform – thus strengthening the result in Agarwal et al. (2006).

The proof of Theorem 3.1 relies on the fact that a random walk on H satisfies a property known as *time-reversibility*: $\pi_u p_{u,v} = \pi_v p_{v,u}$ for all vertices $u, v \in V$, where π is the stationary distribution of the random walk (Aldous & Fill, 2002). It is well-known that a Markov chain can be represented as a random walk on a graph if and only if it is time-reversible. Moreover, time-reversiblility allows us to derive a formula for the weights $w_{u,v}$ on G^H. Let $\gamma(v) = \gamma_e(v)$ be the edge-independent weight for vertex v. Then,

$$w_{u,v} = \pi_u p_{u,v} = \sum_{e \in E(u,v)} \frac{\omega(e)\gamma(u)\gamma(v)}{\delta(e)}. \tag{2}$$

Conversely, the caption of Figure 1 describes a simple example of a hypergraph with edge-dependent vertex weights that is not time-reversible. This proves the following result.

Theorem 3.2. *There exists a hypergraph with edge-dependent weights $H = (V, E, \omega, \gamma)$ such that a random walk on H is not equivalent to a random walk on its clique graph G^H for any choice of edge weights on G^H.*

Anecdotally, we find from simulations that most random walks on hypergraphs with edge-dependent vertex weights are not time-reversible, and therefore satisfy Theorem 3.2. However, it is not clear how to formalize this observation.

Theorem 3.2 says that random walks on graphs with vertex set V are a *strict* subset of Markov chains on V. A natural follow-up question is whether *all* Markov chains on V can be described as a random walk on some hypergraph H with vertex set V and edge-dependent vertex weights. In the Supplement, we show that the answer to this question is no and provide a counterexample.

In addition, we show in the Supplement that hypergraphs with edge-dependent vertex weights create a rich hierarchy of Markov chains, beyond the division between time-reversible and time-irreversible Markov chains. In particular, we show that random walks on hypergraphs with edge dependent vertex weights and at least one hyperedge of cardinality k cannot in general be reduced to a random walk on a hypergraph with hyperedges of cardinality at most $k - 1$.

Finally, note that our definition of equivalent random walks (Definition 3.2) requires the probability transition matrices to be equal. Thus, another natural question is: given $H = (V, E, \omega, \gamma)$, do there exist weights on the clique graph G^H such that random walks on H and G^H are "close"? We provide a partial answer to this question in Section 5, where we show that, for a specific choice of weights on G^H, the second-smallest eigenvalues of the Laplacian matrices of H and G^H are close.

4. Stationary Distribution and Mixing Time

4.1. Stationary Distribution

Recall the formula for the stationary distribution of a random walk on a graph. If $G = (V, E, w)$ is a graph, then the stationary distribution π of a random walk on G is

$$\pi_v = \rho \sum_{e \in E(v)} w(e), \tag{3}$$

where $\rho = \left(2\sum_{e\in E} w(e)\right)^{-1}$. We derive a formula for the stationary distribution for a random walk on a hypergraph with edge-dependent vertex weights; the formula is analogous to equation (3) above with two important changes: first, the proportionality constant ρ depends on the hyperedge, and second, each term in the sum is multiplied by the vertex weight $\gamma_e(v)$.

Theorem 4.1. *Let $H = (V, E, \omega, \gamma)$ be a hypergraph with edge-independent vertex weights. There exist positive constants ρ_e such that the stationary distribution π of a random walk on H is*

$$\pi_v = \sum_{e\in E(v)} \rho_e \omega(e)\gamma_e(v). \tag{4}$$

Moreover, ρ_e can be computed in time $O\left(|E|^3 + |E|^2 \cdot |V|\right)$.

Note that while the vertex weights $\gamma_e(v)$ can be scaled arbitrarily without affecting the properties of the random walk, Theorem 4.1 suggests that ρ_e is the "correct" scaling factor.

When the hypergraph has edge-independent vertex weights (i.e. $\gamma_e(v) = \gamma(v)$ for all incident hyperedges e), $\rho_e = \left(\sum_{v\in V} \gamma(v)d(v)\right)^{-1}$, leading to the following formula for the stationary distribution:

$$\pi_v = \frac{d(v)\gamma(v)}{\sum_{v\in V} d(v)\gamma(v)}. \tag{5}$$

Furthermore, if the vertex weights are trivial (i.e. $\gamma(v) = 1$) then $\pi_v = d(v)/\sum_{v\in V} d(v)$, recovering the formula derived in Zhou et al. (2006) for the stationary distribution of hypergraphs with trivial vertex weights.

4.2. Mixing Time

In this section, we derive a bound on the mixing time of a random walk on $H = (V, E, \omega, \gamma)$. First, we recall the definition of the mixing time of a Markov chain.

Definition 4.1. *Let M be a Markov chain with states S and probability transition matrix P. The* mixing time *of M is*

$$t_{mix}(\epsilon) = \min\{t \geq 0 : ||P^t(s, \cdot) - \pi||_{TV} \leq \epsilon, \forall s \in S\},$$

where $||\cdot||_{TV}$ is the total variation distance.

We derive the following bound on the mixing time for a random walk on a hypergraph with edge-dependent vertex weights.

Theorem 4.2. *Let $H = (V, E, \omega, \gamma)$ be a hypergraph with edge-dependent vertex weights. Without loss of generality, assume $\rho_e = 1$ (i.e. by multiplying the vertex weights in hyperedge e by ρ_e). Then,*

$$t^H_{mix}(\epsilon) \leq \left\lceil \frac{8\beta_1}{\Phi^2} \log\left(\frac{1}{2\epsilon\sqrt{d_{min}\beta_2}}\right)\right\rceil, \tag{6}$$

where

- *Φ is the Cheeger constant of a random walk on H (Montenegro & Tetali, 2006; Jerison, 2013)*
- *d_{min} is the minimum degree of a vertex in H, i.e. $d_{min} = \min_v d(v)$,*
- *$\beta_1 = \min\limits_{e\in E, v\in e}\left(\dfrac{\gamma_e(v)}{\delta(e)}\right)$,*
- *$\beta_2 = \min\limits_{e\in E, v\in e}\left(\gamma_e(v)\right)$.*

This bound on the mixing time of the hypergraph random walk has a similar form to the bound on the mixing time bound for a random walk on a graph (Jerison, 2013). For a graph G with edge weights $w(e)$ satisfying $\sum_v d(v) = 1$, we have,

$$t^G_{mix}(\epsilon) \leq \left\lceil \frac{2}{\Phi^2} \log\left(\frac{1}{2\epsilon\sqrt{d_{min}}}\right)\right\rceil. \tag{7}$$

Note that both $t^H_{mix}(\epsilon)$ and $t^G_{mix}(\epsilon)$ have the same dependence on $1/\Phi^2, \log(1/\epsilon)$, and $\log(1/\sqrt{d_{min}})$. Intuitively, the additional dependence of $t^H_{mix}(\epsilon)$ on β_1 and β_2 is because small values of β_1 and β_2 correspond to the hypergraph having vertices that are hard to reach, and the presence of such vertices increases the mixing time.

5. Hypergraph Laplacian

Let $H = (V, E, \omega, \gamma)$ be a hypergraph with edge-dependent vertex weights. Since a random walk on H is a Markov chain, we can model the transition probabilities $p^H_{u,v}$ of the random walk using a weighted *directed* graph G with the same vertex set V. Specifically, let $G = (V, E', w')$ be a directed graph with directed edges $E' = \{(u, v) : \exists\ e \in E \text{ with } u, v \in e\}$, and edge weights $w'_{u,v} = p^H_{u,v}$. Extending the definition of the Laplacian matrix for directed graphs (Chung, 2005), we define a Laplacian matrix L for the hypergraph H as follows.

Definition 5.1 (Random walk-based hypergraph Laplacian)**.** *Let $H = (V, E, \omega, \gamma)$ be a hypergraph with edge-dependent vertex weights. Let P be the probability transition matrix of a random walk on H with stationary distribution π. Let Π be a $|V| \times |V|$ diagonal matrix with $\Pi_{v,v} = \pi_v$. Then, the random walk-based hypergraph Laplacian matrix L is*

$$L = \Pi - \frac{\Pi P + P^T\Pi}{2}. \tag{8}$$

At first glance, one might hypothesize that the hypergraph Laplacian L defined above does not model higher-order relations between vertices, since L is defined using a directed graph containing edges only between pairs of vertices. Indeed, if H has edge-independent vertex weights, then it is

true that L does not model higher-order relations between vertices. This is because the transition probabilities $p^H_{u,v}$ are completely determined by the edge weights of the *undirected* clique graph G^H (Theorem 3.1). Thus, for each pair (u, v) of vertices in H, only a single quantity $w_{u,v}$, which encodes a pairwise relation between u and v, is required to define the random walk. As such, the Laplacian matrix L defined in Equation (8) is equal to the Laplacian matrix of an undirected graph, showing that L only encodes pairwise relationships between vertices.

In contrast, when H has edge-dependent vertex weights, the transition probabilities $p^H_{u,v}$ generally cannot be computed from a single quantity $w_{u,v}$ defined for each pair (u, v) of vertices (Theorem 3.2). The absence of such a reduction implies that the transition probabilities $p^H_{u,v}$, which are the edge weights of the directed graph G', encode higher-order relations between vertices. Thus, the Laplacian matrix L also encodes these higher-order relations.

From Chung (2005), the hypergraph Laplacian matrix L given in equation (8) is positive semi-definite and has a Rayleigh quotient for computing its eigenvalues. L can be used in developing spectral learning algorithms for hypergraphs with edge-dependent vertex weights, or to study the properties of random walks on such hypergraphs. For example, the following Cheeger inequality for hypergraphs follows directly from the Cheeger inequality for directed graphs (Chung, 2005).

Theorem 5.1 (Cheeger inequality for hypergraphs)**.** *Let $H = (V, E, \omega, \gamma)$ be a hypergraph with edge-dependent vertex weights. Let L be the Laplacian matrix given in equation (8), and let Φ be the Cheeger constant of a random walk on H. Let λ_i be the non-zero eigenvalues of L, and let $\lambda = \min_i \lambda_i$. We have*

$$\frac{\Phi^2}{2} \leq \lambda \leq 2\Phi. \tag{9}$$

5.1. Approximating the Hypergraph Laplacian with a Graph Laplacian

In Section 3, we posed the following question: given a hypergraph H with edge-dependent vertex weights, can we find weights on the clique graph G^H such that the random walks of H and G are close? We prove the following result.

Theorem 5.2. *Let $H = (V, E, \omega, \gamma)$ be a hypergraph, with the edge-dependent vertex weights normalized so that $\rho_e = 1$ for all hyperedges e. Let G^H be the clique graph of H, with edge weights*

$$w_{u,v} = \sum_{e \in E(u,v)} \frac{\omega(e)\gamma_e(u)\gamma_e(v)}{\delta(e)}. \tag{10}$$

Let L^H, L^G be the Laplacians of H and G^H, respectively, and let λ^H_1, λ^G_1 be the second-smallest eigenvalues of L^H, L^G, respectively. Then

$$\frac{1}{c(H)}\lambda^H_1 \leq \lambda^G_1 \leq c(H)\lambda^H_1, \tag{11}$$

where $c(H) = \max_{v \in V} \left(\frac{\max_{e \in E} \gamma_e(v)}{\min_{e \in E} \gamma_e(v)} \right)$.

This theorem says that there exist edge weights $w_{u,v}$ on G^H such that second smallest eigenvalues of the Laplacians of H and G^H are within a constant factor $c(H)$ of each other, where $c(H)$ is determined by the vertex weights. We do not know if the edge weights in Equation (10) give the tightest bound, or if another choice of edge weights on G^H will yield a Laplacian L^G that is "closer" to the hypergraph Laplacians L^H.

Interestingly, Zhang et al. (2018a) use a variant of L^G as the Laplacian matrix of a hypergraph with edge-dependent vertex weights, and obtain state-of-the-art results on an object classification task. Theorem 5.2 provides some theoretical evidence for why Zhang et al. (2018a) are able to obtain good results, even with the "wrong" Laplacian.

6. Experiments

We demonstrate the utility of hypergraphs with edge-dependent vertex weights in two different ranking applications: ranking authors in an academic citation network, and ranking players in a video game.

6.1. Citation Network

We construct a citation network of all machine learning papers from NIPS, ICML, KDD, IJCAI, UAI, ICLR, and COLT published on or before 10/27/2017, and extracted from the ArnetMiner database (Tang et al., 2008). We represent the network as a hypergraph whose vertices V are authors and whose hyperedges E are papers, such that each hyperedge e connects the authors of a paper. The hypergraph has $|V| = 28551$ vertices and $|E| = 25423$ hyperedges.

We consider two vertex weighted hypergraphs: $H_T = (V, E, \omega, \mathbf{1})$ has trivial vertex weights with $\gamma_e(v) = 1$ for all for all vertices v and incident hyperedges e, and $H_D = (V, E, \omega, \gamma_e)$ has edge-dependent vertex weights

$$\gamma_e(v) = \begin{cases} 2 & \text{if vertex } v \text{ is the first or last author of paper,} \\ 1 & \text{if vertex } v \text{ is a middle author of paper.} \end{cases}$$

The edge-dependent vertex weights $\gamma_e(v)$ model unequal contributions by different authors. For papers whose authors are in alphabetical order (as is common in theory papers), we set vertex weights $\gamma_e(v) = 1$ for all $v \in e$. We set the hyperedge weights $\omega(e) = (\text{number of citations for paper } e) + 1$ in both hypergraphs.

We calculate the stationary distribution of a random walk with restart on both H_T and H_D (restart parameter $\beta = 0.4$), and rank authors v in each hypergraph by their value in the stationary distribution. This yields two different rankings of authors: one with edge-independent vertex weights, and one with edge-dependent vertex weights.

The two rankings have a Kendall τ correlation coefficient (Kendall, 1938) of 0.77, indicating modest similarity. Examining individual authors, we typically see that authors who are first/last authors on their most cited papers have higher rankings in H_D compared to H_T, e.g. Ian Goodfellow (Goodfellow et al., 2014). In contrast, authors who are middle authors on their most cited papers have lower rankings in H_D relative to their rankings in H_T. Table 1 shows the authors with rank above 700 in at least one of the two hypergraphs, and with the largest gain in rank in H_D relative to H_T.

Table 1. Highly ranked authors with the largest increase in rank when edge-dependent vertex weights are used in the hypergraph citation network.

Name	Rank in H_T	Rank in H_D
Richard Socher	687	382
Zhongzhi Shi	543	304
Daniel Rueckert	619	391
Lars Schmidt-Thieme	673	454
Tat-Seng Chua	650	435
Ian J. Goodfellow	612	413

We emphasize that this example is intended to illustrate how a straightforward application of vertex weights leads to alternative author rankings. We do not anticipate that our simple scheme for choosing edge-dependent vertex weights will always yield the best results in practice. For example, Christopher Manning drops in rank when edge-dependent vertex weights are added, but this is because he is the second-to-last, and co-corresponding, author on his most cited papers in the database. A more robust vertex weighting scheme would include knowledge of such equal-contribution authors, and would also incorporate different relative contributions of first, middle, and corresponding authors.

6.2. Rank Aggregation

We illustrate the usage of hypergraphs with edge-dependent vertex weights on the *rank aggregation problem*. The rank aggregation problem aims to combine many partial rankings into one complete ranking. Formally, given a universe $\{1, 2, ..., n\}$ of items and a collection of partial rankings $\tau_1, ..., \tau_k$ (e.g. $\tau_i = (3, 1, 5)$ is a partial ranking expressing item 3 $<$ item 1 $<$ item 5), a rank aggregation algorithm should find a permutation σ on $\{1, 2, ..., n\}$ that is "close" to the partial rankings τ_i.

We consider a particular application of rank aggregation: ranking players in a multiplayer game. Here, the outcome of a game/match gives a partial ranking τ of the players participating in the match. In addition to the ranking, one may also have additional information such as the scores of each player in the match. The latter setting has been extensively studied; classic ranking methods are the ELO (Elo, 1978), and Glicko (Glickman, 1995) systems that are used to rank chess players. More recently, online multiplayer games such as Halo have led to the development of alternative ranking systems such as Microsoft's TrueSkill (Herbrich et al., 2006) and TrueSkill 2 (Minka et al., 2018).

We develop a rank aggregation algorithm that uses random walks on hypergraphs with edge-dependent vertex weights, and evaluate the performance of this algorithm on a real-world datasets of Halo 2 games. In the Supplement, we also include results on experiments with synthetic data.

Data. We analyze the Halo 2 dataset from the TrueSkill paper (Herbrich et al., 2006). This dataset contains two kinds of matches: free-for-all matches with up to 8 players, and 1-v-1 matches. There are 31028 free-for-all matches and 5093 1-v-1 matches among 5507 players. Using the free-for-all matches as partial rankings, we construct rankings of all players in the dataset, and evaluate those rankings on the 1-v-1 matches.

Methods. A well-known class of rank aggregation algorithms are Markov chain-based algorithms, first developed by Dwork et al. (2001). Markov-chain based algorithms create a Markov chain M whose states are the players and whose the transition probabilities depend in some way on the partial rankings. The final ranking of players is determined by sorting the values in the stationary distribution π of M. In our experiments, we use a random walk with restart ($\beta = 0.4$) instead of just a random walk, so that the stationary distribution always exists (Tong et al., 2006).

Using the free-for-all matches, we construct rankings of the players using four algorithms. The first three algorithms use Markov chains: a random walk on hypergraph H with edge-dependent vertex weights; a random walk on a clique graph; and *MC3*, a Markov chain-based rank aggregation algorithm designed by Dwork et al. (2001). The fourth algorithm is TrueSkill (Herbrich et al., 2006).

First, we derive a rank aggregation algorithm using a random walk on a hypergraph $H = (V, E, \omega, \gamma)$ with edge-dependent vertex weights. The vertices V are the players, and the hyperedges E correspond to the free-for-all matches. We set the hyperedge and vertex weights to be

$$\omega(e) = (\text{standard deviation of scores in match } e) + 1,$$
$$\gamma_e(v) = \exp[(\text{score of player } v \text{ in match } e)].$$

This choice of hyperedge weights are inspired by Ding &

Yilmaz (2010), who also use variance to define the hyperedge weights of their hypergraph. For vertex weights, we use $\exp(\text{score})$. We choose these vertex weights instead of raw scores for two reasons: first, scores in Halo 5 can be negative, but vertex weights should be positive, and second, exponentiating the score gives more importance to the winner of a match. We chose to use relatively simple formulas for the hyperedge and vertex weights to evaluate the potential benefits of utilizing edge-dependent vertex weights; further optimization of vertex and edge weights may yield better performance.

Second, we derive a rank aggregation algorithm using a random walk on the clique graph G^H of hypergraph H described above, with the edge weights of G^H given by Equation 10. Specifically, if $H = (V, E, \omega, \gamma)$ is the hypergraph defined above, then G^H is a graph with vertex set V and edge weights $w_{u,v}$ defined by

$$w_{u,v} = \sum_{e \in E(u,v)} \frac{\omega(e)\gamma_e(u)\gamma_e(v)}{\delta(e)}. \tag{12}$$

In contrast to Equation 10, here we do not normalize vertex weights on H so that $\rho_e = 1$ for each hyperedge e, since computing ρ_e is computationally infeasible on our large dataset. Instead, we normalize vertex weights so that $\delta(e) = 1$ for all hyperedges e.

Third, we use *MC3*, a Markov chain-based rank aggregation algorithm designed by Dwork et al. (2001). MC3 uses the partial rankings in each match; it does not use the score information. MC3 is very similar to a random walk on a hypergraph with edge-independent vertex weights. We convert the scores from each player in match i into a partial ranking τ_i of the players, and use the τ_i as input to MC3.

Fourth, we use TrueSkill (Herbrich et al., 2006). TrueSkill models each player's skill with a normal distribution. We rank players according to the mean of this distribution. We also implemented the probabilistic decision procedure for ranking players from the TrueSkill paper, and found no difference in performance between ranking by the mean of the distribution and the probabilistic decision procedure.

Evaluation and Results: We evaluate the rankings of each algorithm by using them to predict the outcomes of the 1-v-1 matches. Specifically, given a ranking π of players, we predict that the winner of a match between two players is the player with the higher ranking in π. Table 2 shows the fraction of 1-v-1 matches correctly predicted by each of the four algorithms. Random walks on the hypergraph with edge-dependent vertex weights have significantly better performance than both MC3 and random walks on the clique graph G^H, and comparable performance to TrueSkill. Moreover, on 8.9% of 1-v-1 matches, the hypergraph method correctly predicts the outcome of the match, while TrueSkill incorrectly predicts the outcome—suggesting that the hypergraph model is capturing some information about the players that TrueSkill is missing. Unfortunately, we are unable to identify any specific pattern in the matches where the hypergraph predicted the outcome correctly and TrueSkill predicted incorrectly.

Table 2. Result of ranking players for Halo 2 Dataset.

	Correctly Predicted
TrueSkill	73.4%
Hypergraph	71.1%
Clique Graph	61.1%
MC3	52.3%

7. Conclusion

In this paper, we use random walks to develop a spectral theory for hypergraphs with edge-dependent vertex weights. We demonstrate both theoretically and experimentally how edge-dependent vertex weights model higher-order information in hypergraphs and improve the performance of hypergraph-based algorithms. At the same time, we show that random walks on hypergraphs with edge-independent vertex weights are equivalent to random walks on graphs, generalizing earlier results tha showed this equivalence in special cases (Agarwal et al., 2006).

There are numerous directions for future work. It would be desirable to evaluate additional applications where hypergraphs with edge-dependent vertex weights have previously been used (e.g. (Zhang et al., 2018a; Li et al., 2018)), replacing the Laplacian used in some of these works with the hypergraph Laplacian introduced in Section 5. Sharper bounds on the approximation of the hypergraph Laplacian by a graph Laplacian are also desirable. Another direction is to examine the relationship between the linear hypergraph Laplacian matrix introduced here and the nonlinear Laplacian operators that were recently introduced in the case of trivial vertex weights (Chan et al., 2018) or submodular vertex weights (Li & Milenkovic, 2017; 2018).

Another interesting direction is in extending graph convolutional neural networks (GCNs) to hypergraphs. Recent approaches to GCNs implement the graph convolution operator as a non-linear function of the graph Laplacian (Kipf & Welling, 2016; Defferrard et al., 2016). GCNs have also been generalized to hypergraph convolutional neural networks (HGCNs), where the convolution layer operates on a hypergraph with edge-independent vertex weights instead of a graph (Yadati et al., 2018; Feng et al., 2018). The hypergraph Laplacian matrix introduced in this paper would allow one to extend HGCNs to hypergraphs with edge-dependent vertex weights.

References

Agarwal, S., Lim, J., Zelnik-Manor, L., Perona, P., Kriegman, D., and Belongie, S. Beyond pairwise clustering. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pp. 838–845 vol. 2, June 2005.

Agarwal, S., Branson, K., and Belongie, S. Higher order learning with graphs. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 17–24, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143847.

Aldous, D. and Fill, J. A. *Reversible Markov Chains and Random Walks on Graphs*. 2002.

Avin, C., Lando, Y., and Lotker, Z. Radio cover time in hyper-graphs. *Ad Hoc Networks*, 12:278 – 290, 2014. ISSN 1570-8705. doi: http://doi.org/10.1016/j.adhoc.2012.08.010.

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

Brin, S. and Page, L. The anatomy of a large-scale hypertextual web search engine. In *Seventh International World-Wide Web Conference (WWW 1998)*, 1998.

Chan, T.-H. H., Louis, A., Tang, Z. G., and Zhang, C. Spectral properties of hypergraph laplacian and approximation algorithms. *J. ACM*, 65(3):15:1–15:48, March 2018. ISSN 0004-5411. doi: 10.1145/3178123.

Chung, F. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9(1):1–19, Apr 2005. ISSN 0219-3094. doi: 10.1007/s00026-005-0237-z.

Cooper, C., Frieze, A., and Radzik, T. The cover times of random walks on random uniform hypergraphs. *Theoretical Computer Science*, 509:51 – 69, 2013. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/j.tcs.2013.01.020.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016.

Ding, L. and Yilmaz, A. Interactive image segmentation using probabilistic hypergraphs. *Pattern Recognition*, 43 (5):1863 – 1873, 2010. ISSN 0031-3203.

Ducournau, A. and Bretto, A. Random walks in directed hypergraphs and application to semi-supervised image segmentation. *Comput. Vis. Image Underst.*, 120:91–102, March 2014. ISSN 1077-3142. doi: 10.1016/j.cviu.2013.10.012.

Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pp. 613–622, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. doi: 10.1145/371920.372165.

Elo, A. E. *The rating of chessplayers, past and present*. Arco Pub., New York, 1978. ISBN 0668047216 9780668047210.

Feng, Y., You, H., Zhang, Z., Ji, R., and Gao, Y. Hypergraph neural networks. *CoRR*, abs/1809.09401, 2018.

Glickman, M. E. The glicko system. *Boston University*, 1995.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Harel, D. and Koren, Y. On clustering using random walks. In *Proceedings of the 21st Conference on Foundations of Software Technology and Theoretical Computer Science*, FST TCS '01, pp. 18–41, Berlin, Heidelberg, 2001. Springer-Verlag. ISBN 3-540-43002-4.

Herbrich, R., Minka, T., and Graepel, T. Trueskill™: A bayesian skill rating system. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pp. 569–576, Cambridge, MA, USA, 2006. MIT Press.

Huang, Y., Liu, Q., Zhang, S., and Metaxas, D. N. Image retrieval via probabilistic hypergraph ranking. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3376–3383, June 2010. doi: 10.1109/CVPR.2010.5540012.

Jamali, M. and Ester, M. Trustwalker: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 397–406, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: 10.1145/1557019.1557067.

Jerison, D. General mixing time bounds for finite markov chains via the absolute spectral gap, October 2013.

Kendall, M. G. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. ISSN 00063444.

Kim, S., Nowozin, S., Kohli, P., and Yoo, C. D. Higher-order correlation clustering for image segmentation. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F.,

and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 1530–1538. Curran Associates, Inc., 2011.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

Li, J., He, J., and Zhu, Y. E-tail product return prediction via hypergraph-based local graph cut. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, pp. 519–527, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5552-0.

Li, P. and Milenkovic, O. Inhomogeneous hypergraph clustering with applications. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2308–2318. Curran Associates, Inc., 2017.

Li, P. and Milenkovic, O. Submodular hypergraphs: p-laplacians, Cheeger inequalities and spectral clustering. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3014–3023, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Minka, T., Cleven, R., and Zaykov, Y. Trueskill 2: An improved bayesian skill rating system. March 2018.

Montenegro, R. and Tetali, P. Mathematical aspects of mixing times in markov chains. *Found. Trends Theor. Comput. Sci.*, 1(3):237–354, May 2006. ISSN 1551-305X. doi: 10.1561/0400000003.

Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pp. 849–856, Cambridge, MA, USA, 2001. MIT Press.

Ramadan, E., Tarafdar, A., and Pothen, A. A hypergraph model for the yeast protein complex network. In *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, pp. 189–, April 2004. doi: 10.1109/IPDPS.2004.1303205.

Ritz, A., Tegge, A. N., Kim, H., Poirel, C. L., and Murali, T. Signaling hypergraphs. *Trends in Biotechnology*, 32(7): 356 – 362, 2014. ISSN 0167-7799. doi: http://doi.org/10.1016/j.tibtech.2014.04.007.

Rodriguez-Velazquez, J. A. On the laplacian eigenvalues and metric parameters of hypergraphs. *Linear and Multilinear Algebra*, 50:1–14, 03 2002.

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pp. 990–998, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1402008.

Tong, H., Faloutsos, C., and Pan, J. Fast random walk with restart and its applications. In *Sixth International Conference on Data Mining (ICDM'06)*, pp. 613–622, Dec 2006. doi: 10.1109/ICDM.2006.70.

Yadati, N., Nimishakavi, M., Yadav, P., Louis, A., and Talukdar, P. Hypergcn: Hypergraph convolutional networks for semi-supervised classification. *CoRR*, abs/1809.02589, 2018.

Yang, W., Wang, G., Bhuiyan, M. Z. A., and Choo, K.-K. R. Hypergraph partitioning for social networks based on information entropy modularity. *Journal of Network and Computer Applications*, 86:59 – 71, 2017. ISSN 1084-8045. Special Issue on Pervasive Social Networking.

Zeng, K., Wu, N., Sargolzaei, A., and Yen, K. Learn to rank images: A unified probabilistic hypergraph model for visual search. *Mathematical Problems in Engineering*, 2016:1–7, 01 2016. doi: 10.1155/2016/7916450.

Zhang, Z., Lin, H., and Gao, Y. Dynamic hypergraph structure learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3162–3169. International Joint Conferences on Artificial Intelligence Organization, 7 2018a.

Zhang, Z., Lin, H., Zhao, X., Ji, R., and Gao, Y. Inductive multi-hypergraph learning and its application on view-based 3d object classification. *IEEE Transactions on Image Processing*, 27(12):5957–5968, Dec 2018b.

Zhou, D., Huang, J., and Schölkopf, B. Learning with hypergraphs: Clustering, classification, and embedding. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pp. 1601–1608, Cambridge, MA, USA, 2006. MIT Press.