INTERNATIONAL JOURNAL OF EDUCATION AND INFORMATION TECHNOLOGIES
DOI: 10.46300/9109.2021.15.2 Volume 15, 2021

Random Walks in Hypergraph

Abdelghani Bellaachia, Mohammed Al-Dhelaan
Computer Science Department
The George Washington University
Washington, DC 20052, USA
Email:{bell, mdhelaan}@gwu.edu

Received: January 27, 2021. Revised: February 24, 2021. Accepted: March 8, 2021. Published: March 10, 2021.

***Abstract*—Random walks on graphs have been extensively used for a variety of graph-based problems such as ranking vertices, predicting links, recommendations, and clustering. However, many complex problems mandate a high-order graph representation to accurately capture the relationship structure inherent in them. Hypergraphs are particularly useful for such models due to the density of information stored in their structure. In this paper, we propose a novel extension to defining random walks on hypergraphs. Our proposed approach combines the weights of destination vertices and hyperedges in a probabilistic manner to accurately capture transition probabilities. We study and analyze our generalized form of random walks suitable for the structure of hypergraphs. We show the effectiveness of our model by conducting a text ranking experiment on a real world data set with a 9% to 33% improvement in precision and a range of 7% to 50% improvement in Bpref over other random walk approaches.**

***Keywords*-Hypergraph, Random walk, Weighted Random Walk, Hypergraph Ranking, Learning with Hypergraphs, Lexical Hypergraph;**

I. Introduction

Graph-based approaches are traditionally used for a variety of tasks. Such task includes, but not limited to, ranking, clustering, and recommending elements. By modeling the data as a graph, we capture the relationship between elements where we have elements as vertices and the relationships as edges. The structure of the graph, determined by its edges, provides a characterization of these relationships. To learn from the structure of a graph, a number of approaches exist. A well-known method that helps with learning from graphs is the concept of a random walk. A random walk is the process of randomly traversing the graph from a node to another adjacent node at each step. Such a process has shown tremendous benefits and application in a wide range area in computing.

Graphs, however, can only capture pair-wise relationships between nodes preventing us from modeling complex relations. Let us clarify by an example, let us imagine a similarity graph where nodes are objects and the similarity is represented as weighted edges. A similarity edge connects two vertices where it represents that the two nodes share some similarity. In this case, we can only consider the similarity between two nodes. We cannot model the similarity between more than a pair of nodes. Therefore, the usage of a *hypergraph* becomes important to not lose information and accurately model high-order relations.

Second author is sponsored by King Saud University, Saudi Arabia

A hypergraph is a generalization of simple graphs where the edges, called *hyperedges*, can have any subset of nodes. A hyperedge could contain more than two vertices which makes it more suitable for high-order relations than simple graphs. Moreover, it is abundantly clear that simple graphs are special case of hypergraphs where the edges must have two end-point vertices [1][2][3][4]. To model the aforementioned similarity example in a hypergraph, we could represent any number of objects (vertices) in a hyperedge showing that they all have similar properties if they belong to a hyperedge. Such representation could be crucial for clustering or community detection problems for instance. Regardless of the high-order representation in hypergraphs, there has been a limited number of literature on generalizing random walks to hypergraphs. A number of interesting challenges arises when trying to define random walks in hypergraphs. How can the surfer traverse hyperedges? How can we generalize the random walk in simple graphs to hypergraphs?

In this paper, we seek to generalize the hypergraph random walk found in [3][4]. We redefine the random walk process in hypergraphs where the surfer could differentiate between destination vertices within a hyperedge depending on their features (vertex weights). We propose a probabilistic weighted random walk that takes advantage of the hypergraph structure. By extending the random walk in hypergraphs, we show that the proposed random walk is more general than other approaches. We, then, demonstrate the effectiveness of our approach by comparing it with other proposed random walk approaches in hypergraphs.

The paper is organized as follow: A discussion of the related work is in Section II. We define the hypergraph notation needed for explaining the proposed approach in III. The proposed approach will be thoroughly explained in Section IV. Section V will describe the data and experimental results. The paper conclusion is in Section VI.

II. Related Work

A. Simple Graphs

Random walks on simple graphs have been studied thoroughly and extensively in the literature [5] [6]. With enormous algorithmic applications as in ranking [7] [8] [9] [10], similarity and recommendation [11] [12] [13], or link predication [14]. The most seminal work is PageRank [15] which revolutionized the search world and inspired so many others. Most random walk approaches use a finite Markov chain to

model the walk. A great explanation on the subject can be found in [16] [17]. Moreover, random walks are applied to heterogeneous graphs where we can have vertices of different types. [18] [19].

B. Hypergraphs.

Surprisingly, there is limited literature on random walks in hypergraphs. A hypergraph [20] is a generalization of a graph where an edge could have more than two vertices. Avin et al. proposed the simplest random walk in a hypergraph where the surfer chooses uniformly at random from hyperedges and vertices [3]. They also proposed a random walk over a directed hypergraph. The authors studied the radio cover time for a random walk to be heard by other vertices. Zhou et al. proposed a hypergraph random walk where the surfer considers the weights of hyperedges [4]. A similar walk is found in [21] for hypergraph matching. Zhou et al. used the random walk to study spectral clustering and semi-supervised ranking in hypergraphs. Moreover, Cooper et al. proposed to define the cover time and inform time of a general random walk in a r-uniform hypergraph [22]. Lu and Peng studied a high-order general random walk in hypergraphs in their attempt to do spectral analysis of hypergraphs [23]. They proved that the eigenvalues of their proposed Laplacians affects the mixing rate of random walks in hypergraphs.

Zhou et al. [4] inspired so many to apply the hypergraph random walk approach. In [24], the authors applied ranking in their proposed music recommendation framework that encompasses social media and music acoustic-based content. Similarly, personalized news recommendation model was proposed where the authors applied ranking on hypergraph that includes users, news articles, and topics [25]. Wang et al. proposed a similar semi-supervised framework for sentence ranking that is used in text summarization [26]. Liu et al. used the hypergraph random walk to measure similarity between item sets in a database [2]. In another approach, Agarwal et al. argued that a hypergraph could be transformed to a normal graph by treating each hyperedge as a clique or a star [1]. Similarly, Tan et al. applied similar approach in ranking video hyperlinks by using a random walk in a star-shaped graph [27]. However, these approaches convert the hypergraph into a simple graph which could lead to information-loss. For example, let us assume we want to model faculty members as vertices and their departmental affiliation as a hyperedge to group them together. A faculty member could be affiliated with more than one department where the vertex will be in all hyperedges of his departments. If a hypergraph is converted to a simple graph, we can only tell if two faculty members share the same department but we cannot tell which department that is.

III. Notations and Definitions

Let $G(V, E)$ be an undirected graph with the vertex set V and the set of edges E. The degree of v in a normal graph is denoted $d_G(v)$ which is the number of edges incident with v and defined by:

$$d_G(v) \stackrel{\text{def}}{=} |\{e \in E : v \in e\}| \tag{1}$$

We denote D_G to be the diagonal matrix containing the vertex degrees of the normal graph. The degree of the edges is always $\delta(e) = 2$. When the aforementioned definition is relaxed, we can have a more generalized form of graphs known as hypergraphs.

Let $HG(V, \mathcal{E})$ be a hypergraph with the vertex set V and the set of hyperedges $\mathcal{E}$. A hyperedge e is a subset of V where $\cup_{e\in\mathcal{E}} e = V$. Let $HG(V, \mathcal{E}, w)$ be a weighted hypergraph where $w : \mathcal{E} \to \mathbb{R}^+$ is the hyperedge weight. The hypergraph is said to be *connected* when there is a path between each pair of vertices. A path is a connected sequence of vertices over hyperedges $\{v_1, e_1, v_2, e_2, ..., e_{k-1}, v_k\}$ where $\{v_i, v_{i+1}\} \subseteq e_i$. A hyperedge e is said to be incident with v when $v \in e$. A hypergraph has an incidence matrix $H \in \mathbb{R}^{|V|\times|E|}$ as follows:

$$h(v, e) = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{if } v \notin e \end{cases} \tag{2}$$

The vertex and hyperedge degree are defined as follows:

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

$$\delta(e) = \sum_{v\in V} h(v, e) = |e| \tag{4}$$

D_e and D_v are the diagonal matrices representing the degrees of hyperedges and vertices, respectively. W_e is the diagonal matrix with the hyperedge weights.

IV. Generalizing Random Walks for Hypergraph

Random walk is a process of transitioning between vertices in a graph by starting at a given vertex and moving to another neighboring vertex after each discrete time step t. The process is modeled as a finite Markov chain $\mathcal{M}$ over a set of states $\{s_1, s_2, ..., s_n\}$. Each state s_i is analogous to a vertex v in the graph G, and the transition is a conditional probability defined as $P(u, v) = Prob(s_{t+1} = v|s_t = u)$ which means that the chain $\mathcal{M}$ will be at v at time $t + 1$ given that it was in u at time t. Note that the probability of transitioning from state to another is completely independent of time t which makes the Markov chain *homogeneous*. Moreover, for any vertex u we have $\sum_v P(u, v) = 1$. Given the fact that a Markov chain is memoryless with probabilities computed over only a single transition, we can then define a transition matrix $P \in \mathbb{R}^{|V|\times|V|}$ for all moves. This matrix P represents all transitions that could capture the behavior of a surfer randomly moving between vertices according to such probabilities. In order to generalize a random walk for a hypergraph, we will review the random walk over a normal graph first.

In a normal unweighted graph G, the simplest random walk from u to v is to choose an edge e_{uv} uniformly at random. The transition probability $P(u, v)$ is calculated as follows:

$$P(u, v) = \frac{1}{d(u)} \tag{5}$$

Where $d(u)$ is the degree of source vertex u. Alternatively, let A be the adjacency matrix of G. We can write

$$P = D_G^{-1} A$$

In a more general weighted graph, the surfer chooses an edge with probability proportional to the edge weight. The transition for weighted graph is defined as follows:

$$P(u,v) = \frac{w(u,v)}{d(u)} \tag{6}$$

Where $d(u) = \sum_x w(u,x)$ and x being all neighbors of u. Note that the simple unweighted walk is a special case of the weighted walk where the probability distribution $P(u,v)$ for a given $u \in V$ is uniform for any $v \in V$ adjacent to u.

Whether it is weighted or simple, the random walk in normal graphs is well studied and defined. However, it is not the case in Hypergraphs where the structure of the graph is substantially different. For instance, in a normal graph the imaginary surfer will cross an edge ending at single vertex (destination) meaning that it has just a simple probability of going to that vertex. However, in a hypergraph, a hyperedge could have more than two end-point vertices $\delta(e) \geq 2$.

To generalize the random walk process in hypergraphs, we model the walk as the transition between two vertices that are incident to each other in a *hyperedge* instead of a normal edge. In essence, the random walk is seen to be a two-step process, instead of one, which is the following: the random surfer first chooses a hyperedge e incident with the current vertex u. Then the surfer picks a destination vertex v within the chosen hyperedge satisfying the following $u, v \in e$. The random walk in hypergraph is said to be more general since the random walk in a normal graph is a special case where there is only a single destination vertex v associated with a given normal edge incident with u where in a hypergraph we can have more vertices to choose from. The hypergraph random walk process can be defined as a Markov chain where the vertex set is the state set of the chain similar to a normal graph. At each time step t the surfer moves in the incident hyperedge to another vertex. In [3], the simplest random walk is defined in an unweighted hypergraph $HG(V, \mathcal{E})$ where the surfer chooses a hyperedge e uniformly at random and then chooses a vertex uniformly at random from e. We define the vertex and hyperedge degree as follows:

$$d(u) = |\mathcal{E}(u)| \tag{7}$$

$$\delta(e) = |e| \tag{8}$$

Where $d(u)$ is the vertex degree, $\delta(e)$ is the hyperedge degree, and $\mathcal{E}(u)$ is the set of hyperedges incident to u. P is the transition matrix for this random walk

$$P(u,v) = \frac{1}{d(u)} \sum_{e \in \mathcal{E}(u) \cap \mathcal{E}(v)} \frac{1}{\delta(e)} \tag{9}$$

Alternatively, we can write:

$$P = D_v^{-1} H D_e^{-1} H^T$$

This approach can be extended to a weighted hypergraph $HG(V, \mathcal{E}, w)$ where $w(e)$ is the hyperedge weight. A random walk can be extended to a non-uniform walk proportional to the hyperedge weight $w(e)$ as in the work done in [4]. In this case, hyperedges incident with u do have different probability depending on their weights. So the process will be 1) to choose a hyperedge non-uniformly proportional to $w(e)$. Then 2) choose v within the selected hyperedge uniformly at random. We can calculate the transition matrix P as follows:

$$P(u,v) = \sum_{e \in E} w(e) \frac{h(u,e)}{\sum_{\hat{e} \in \mathcal{E}(u)} w(\hat{e})} \frac{h(v,e)}{\delta(e)} \tag{10}$$

Or in matrix notation:

$$P = D_v^{-1} H W_e D_e^{-1} H^T$$

Where the D_v is the diagonal matrix of the *weighted* degree of vertices as in formula 3.

In this paper, we try to seek a more general definition of a random walk in a weighted hypergraph where not only hyperedges have weights, but vertices as well. In such a case, the random walk process is extended to leverage both hyperedges' and vertices' weights. We define the vertex weight across all incident hyperedges to be a feature vector

$$\vec{v}_w = \{w(v_{e1}), w(v_{e2}), ..., w(v_{d(v)})\} \tag{11}$$

where we have a different vertex weight for every hyperedge e that contain vertex v. We describe the proposed random walk process as the following. Starting from a vertex u, the surfer chooses a hyperedge e incident with u proportional to the hyperedge weight $w(e)$. Then, the surfer, also chooses a vertex v proportional to the vertex weight within the hyperedge where we consider the weight in the current hyperedge only. Let us define a weighted hypergraph incident matrix $W \in \mathbb{R}^{|V| \times |E|}$ where we have the following:

$$w(v,e) = \begin{cases} w(v_e) & \text{if } v \in e \\ 0 & \text{if } v \notin e \end{cases} \tag{12}$$

Therefore, we redefine the hyperedge degree to be as follows:

$$\delta(e) = \sum_{v \in e} w(v,e) \tag{13}$$

We can now calculate the transition matrix P as follows:

$$P(u,v) = \sum_{e \in E} w(e) \frac{h(u,e)}{\sum_{\hat{e} \in \mathcal{E}(u)} w(\hat{e})} \frac{w(v,e)}{\sum_{\hat{v} \in e} w(\hat{v},e)} \tag{14}$$

Or in matrix notation:

$$P = D_v^{-1} H W_e D_{ve}^{-1} W^T$$

Where $w(v,e)$ is the weight of the destination vertex v in hyperedge e. D_{ve} is the diagonal matrix for weighted degree of a hyperedge as in formula 13. Note that the transition matrix P is *stochastic* where we have every row sums to 1.

The proposed random walk is more general than previous approaches which are considered special cases of the proposed approach. For instance, if we restrict the vertex weights to be either 1 or 0, we get the random walk in formula 10. Moreover, our approach is more general than the random walk in formula 9 where it is a special case with hyperedge and vertex weights are either 1 or 0 as well.

We argue that when features are available for vertices, they can enhance the task at hand by adding more knowledge for the imaginary surfer. Hence, modeling these features as vertex weights adds preference to the surfer when performing the walk where the best combination of hyperedges and destination vertices have the highest probability. Such knowledge could be advantageous when performing tasks such as ranking, clustering, predicting links, and personalized search.

Now we move on to explaining the stationary distribution π of a random walk. To calculate the stationary distribution, we start with the initial column vector $v_0 \in \mathbb{R}^{|V|\times 1}$ with equal probabilities $1/|V|$ summing to 1, then after each time step we multiply it by the transition matrix. After the first move, we will have $\vec{v}_1 = P^T\vec{v}_0$ (where P^T is a column stochastic matrix for clarity) giving us a new column vector, then we will have $\vec{v}_2 = P^T(P^T\vec{v}_0)$ for the second move and so on. The reason of multiplying the probability distribution vector $\vec{v}$ by the transition matrix P^T gives us the next step distribution $\vec{x} = P^T\vec{v}$ can be expalined as follows. Let x_i be the probability of being at the current vetrex i. Then we have the following: $x_i = \sum_j p_{ij}v_j$ where v_j being the probability of the surfer being at node j previously, and p_{ij} is the probability of moving from j to i.

The distribution vector $\vec{v}$ stops changing after n steps if the random walk is *ergodic*. Which holds if we have the following:

$$\vec{v} \stackrel{\text{def}}{=} \lim_{x\to\infty}[(P^T)^x\vec{v}_0] = \pi \tag{15}$$

Where π is the stationary distribution. A random walk is *ergodic* when the following conditions are met: 1) the graph is *irreducible*, for any two vertices $u, v \in V$ they must satisfy $P(u, v) > 0$. Also, 2) the graph is *aperiodic*. Note that the vector $\vec{v}$ is a right eigenvector of the transition matrix P^T satisfying $\vec{v} = \lambda P^T\vec{v}$ where λ is the *eigenvalue* of the transtion matrix. So given that the matrix P^T is a stochastic matrix (each column sums to 1), the eigenvector $\vec{v}$ is the *principle* or the dominate eigenvector. Moreover, we know from *Perron-Frobenius Theorem* that if a Markov chain is irreducible and aperiodic, then the largest eigenvalue λ is 1 and all others are strictly less than 1. Therefore, the stationary distribution vector is the principle eigenvector corresponding to the $\lambda = 1$ giving that the matrix P^T is irreducible and aperiodic. Having a principle eigenvector $\vec{v}$ is important to guarantee that regardless of the starting node of the random walk, we will always arrive at a unique dominate eigenvector that does not change anymore.

To guarantee irreducibility and aperiodicity, we use the PageRank algorithm [15]. The algorithm has the idea of *teleporting* which will restart the random walk process making it useful for the previous conditions. The teleporting is depicted with a small probability called the *damping factor* α. It also makes sure to make the graph irreducible since the random walker always has the probability of teleporting to any other node.

$$\vec{v}_{(i+1)} = \alpha P^T\vec{v}_{(i)} + (1-\alpha)e/n \tag{16}$$

The damping factor α is set to 0.85. n is the number of nodes in the graph. $e \in \mathbb{R}^{n\times 1}$ is a vector of all elements being 1. $\alpha P^T\vec{v}$ means that the random walker will choose to go with one of the adjacent edges. $(1-\alpha)\vec{e}/n$ represents a vector of an introductory probabilities with each entry being $(1-\alpha)/n$ to teleport the random walk to a new node.

V. EXPERIEMENT

In this section, we explain the evaluation method of our random walk model. We evaluate our method by conducting an experiment and comparing it with other state-of-the-art models. We test our model's effectiveness in ranking text data by modeling the document keywords in a hypergraph then comparing all the random walk models. First, the data set used will be explained thoroughly. Subsequently, the design of the experiment will be laid out. Finally, the experimental results will be discussed and evaluated.

A. Data Set

To evaluate the ranking accuracy of our model we used the Digital Bibliography and Library Project (DBLP)[1] data set which contains computer science conference proceedings and journals. Specifically, we used the public data set of the DBLP library used by Deng et al. [28] which can be acquired online[2] and is widely used in the areas of topic modeling and ranking. The data set includes 28,569 documents, 28,702 authors collected from 20 major computer science conferences. Moreover, the papers were collected from four computer science areas (1) database, (2) data mining, (3) artificial intelligence, and (4) information retrieval. For quantitative evaluation, we used a labeled subset composed of 4,057 authors, 100 papers and all 20 conferences [29]. The papers were labeled with their topics by assigning one of the four topics to each paper.

B. Experiment Design

We design the experiment as ranking text in short documents. We model the documents in a hypergraph, then apply the different random walk approaches. Furthermore, the evaluation is conducted as ranking text and comparing all the ranked lists of keywords. A full quantitative evaluation is conducted to measure the effectiveness of our model in ranking in a hypergraph. We model each research paper, d_i, as a bag of words from the title of the paper $d_i = \{k_1, k_2, ..., k_s\}$, where k_i is a keyword. Then, we represent the collection of documents $D_i = \{d_1, d_2, ..., d_n\}$ as a lexical hypergraph as in the following. Each document title d_i is being represent as a hyperedge, and each distinct keyword k_i is modeled as a vertex. By constructing the lexical hypergraph, we can rank

[1]http://www.informatik.uni-trier.de/ ley/db/
[2]http://www.cs.uiuc.edu/ hbdeng/data/kdd2011.htm

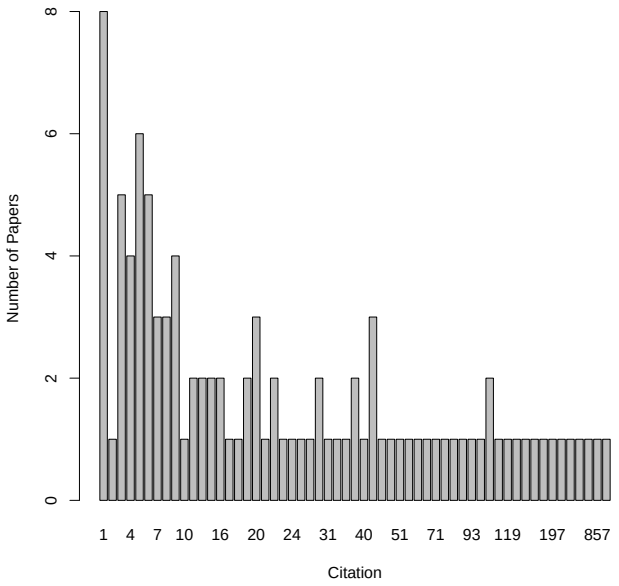

Fig. 1. Citation Data

keywords using the random walk algorithms. The intuition behind the approach is that keywords that tend to co-occur together should be ranked higher than other keywords. Similar approaches had been used in [30][31].

For each hyperedge, we define the hyperedge weight as a document's feature to reflect its importance. Since documents are research papers, we have used the citation to calculate the importance of documents. Given that the original data set does not contain the citation data, we have collected the citation information for all 100 papers used in this experiment. To that end, we used the Google Scholar service to collect the data. In the case where the information is not available in the service, we assume it is not cited. However, a close analysis of the data shows that it is largely spread out with a large standard deviation of $\sigma = 155.0093$. Figure 1 shows how stretched out the data.

Therefore, we calculate the hyperedge weight as the following:

$$W(e) = \frac{c_e + 1}{\sum_e c_e + 1} \tag{17}$$

Where c_e is the number of citation for the document hyperedge e and $\sum_e c_e + 1$ is the total number of citation in the data set. We add 1 to avoid cancelling out papers with no citation. The distribution of the citation data has a standard deviation $\sigma = 0.0233$. The intuition behind the hyperedge weight is that when a paper has been cited a lot, the keywords in its title should be ranked higher than a paper with low or no citation.

For vertices weights, we define the weight to be the Term Frequency-Inverse Document Frequency (tf-idf). However, the short length of document title could minimize the effectiveness of a frequency measure as tf-idf. Therefore, we grouped research papers using the topics they belong to as the following. For each topic, we have a set of papers that will be concatenated to larger virtual document for a given topic $\delta_t = \{d_1 + d_2 + ... + d_n\}$ where δ_t is the concatenation of small documents and $1 \leq t \leq 4$. Then, the we measure tf-idf over the larger documents being the set of $D = \{\delta_1, \delta_2, \delta_3, \delta_4\}$. The tf-idf is measured as follows:

$$W(V_i)_{tf\text{-}idf} = tf(V_i) \cdot log \frac{N}{df(V_i)} \tag{18}$$

Where $tf(V_i)$ as the term frequency on the document d_t. N is the number of documents in the larger document set D_t, and $df(V_i)$ is the number of larger documents that contain the term V_i.

The tf-idf measure is calculated after the initial preprocessing of text has been conducted. We first removed all stopwords that do not have any topical meaning. Moreover, we removed the punctuations from text. For example, data-consistent becomes data consistent. We also removed numbers from text. Moreover, all words are transformed to lower-case. Finally, we used Porter stemmer to stem all textual units to their root base. For instance, mining becomes mine. These steps are standard steps in ranking text.

We compare our random walk model to other baselines by comparing the ranking of all methods. The baselines used to conduct this experiment are described as the following:

- The first baseline is the simplest random walk defined in equation 9. In this model we use an unweighted hypergraph. We denote this baseline as RW1.
- The second baseline is the random walk proposed by Zhou et al. in equation 10. In this model we use a weighted hypergraph where the weight of hyperedges is defined to be the citation. We denote this baseline as RW2.
- The third is the proposed random walk defined in equation 14. The proposed model uses a weighted hypergraph where the weight of the hyperdges are the citation and the weight of vertices are the tf-idf measure. We call our proposed random walk as RW3.

C. Experiment Metrics

To quantitatively measure the performance of a ranking method, we define some common evaluation metrics that are largely used in the information retrieval literature. These measures need a golden standard to test against. Deng et al. [28] ranked keywords according to topics found in the data set. We empirically compare all baselines to the best ranking done by Deng et al. The keyword is considered to be correct if it appears in the top ranked keywords in any topic. The intuition is that a keyword should be ranked high if it uniquely identifies a topic. We compare the top ranked keywords using two metrics precision and Bpref. We focus on the top precision with top 10, 15, and 20. Precision is measured as the following:

$$Precision = \frac{K_{correct}}{K_{extracted}} \quad (19)$$

where $K_{correct}$ is the number of correctly extracted keywords, and $K_{extracted}$ is the total number of extracted keywords. Even though precision measures how precise the extracted result is, it does not differentiate in the within ranking of the extracted top result. Therefore, we used another metric that measure the ranking within the top extracted keywords. The metric is Binary Preference Measure [32]. Bpref is a great choice to measure the performance of the system with the order of the top extracted keywords taken into account. Bpref is measured as the following:

$$Bpref = \frac{1}{R}\sum_{r \in R} 1 - \frac{|\text{n } \textit{ranked higher than} \text{ r}|}{R} \quad (20)$$

where R is the number of correct keywords within extracted keywords in a method, and where r is a correct keyword and n is incorrect keyword.

D. Experimental Results

In this section, we discuss the experimental results for all the random walk approaches used. We evaluate all baselines by comparing them using two metrics. Specifically, we measure the ranking accuracy at different levels at top 10, 15, 20.

For precision as shown in Figure 2, we compare all three approaches of the random walk algorithm on a hypergraph. The simplest random walk and the hyperedge weighted random walk have surprisingly similar top 10 precision. However, the hyperedge weighted random walk showed much improvement over the simple random walk at top 15 and 20 by 10% and 22% respectively. Our approach, where we used vertex and hyperedge weighted walk, outperformed all baselines at all levels. For the top 10 extracted keywords, our method has a 14% improvement over both the simplest and the hyperedge weighted random walk. Moreover, when we compared the performance at top 15, our approach showed an improvement of 10% and 21% over the hyperedge weighted and the simplest walk respectively. For the top 20 keywords, our approach has outperformed both baselines by 9% and 33% respectively.

Moreover, we measured the accuracy of ranking within each top extracted results with Bpref as shown in Figure 3. Similarly, the hyperedge weighted random walk outperformed the simple walk across all top 10, 15, and 20 by 38%, 26%, and 23% respectively. The proposed approach where we consider vertex and hyperedge weights have outperform all other methods. For top 10, the proposed method have a 50% improvement over the simplest random walk and a 9% improvement over the hyperedge weighted walk. Moreover, for the top 15, the proposed method showed a steady improvement of 35% and 7% over the simple and hyperedge weighted approach. Finally, for the top 20, similar improvements have been found of 33% and 8% over both baselines.

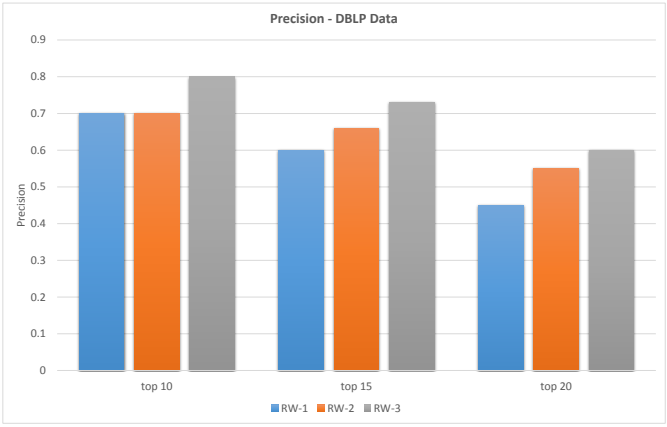

Fig. 2. Comparison of Top 10, 15, and 20 for All Algorithms Using Precision

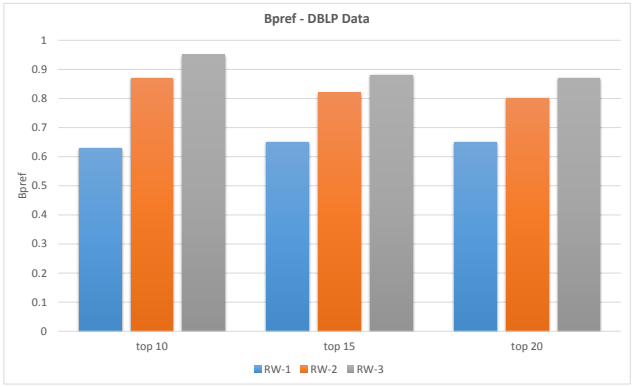

Fig. 3. Comparison of Top 10, 15, and 20 for All Algorithms Using Bpref

E. Discussion and Future Work

In this section, we discuss the results of the experiment. The proposed weighted random walk has showed much improvement over the baselines in both metrics. Using the vertex weights, in addition to hyperedge weight, in calculating the transition probability has enhanced the results in precision and Bpref. The reason for such improvement is that when we choose the right features to be vertex weights, we add preference for nodes that seem more important than others. However, precision does not show the effect of ranking within the top extracted results. Therefore, it does not accurately show the difference in performance. For example, the hyperedge weighted walk RW2 has surprisingly the same precision as the simplest random walk. However, precision ignores the fact that RW2 ranked the correct keywords higher than incorrect keywords which is clear indicator that it is more accurate in ranking. Therefore, we can see that when Bpref is used, the

actual improvement is measured accurately. In the other side, it is clear that if the wrong features are used, the performance of the proposed walk could lead to incorrect results.

In the future, we plan to perform more experiments to test the validity of the approach. An interesting direction for future work is to try applying such an approach to clustering or segmentation instead of ranking. Theoretically, we also plan to study the convergence rate of such a walk and compare it with other standard approaches. Moreover, we plan to investigate random walks with restarts[33][14] in hypergraphs. In random walks with restart, the surfer starts from a given node s, then either keeps traversing edges or teleports back to the original node s. These kind of walks have a variety of application in measuring node proximities and identifying similar nodes.

VI. Conclusion

In this paper, we have studied and analyzed random walks approaches for hypergraphs. We showed that random walk in hypergraphs is visible, and could be naturally generalized from simple graphs. We proposed a new weighted random walk that is suitable for hypergraphs where we devised a probabilistic interpretation for including vertices and hyperedges weights. We performed an experiment over three different random walk approaches in hypergraps and showed that our approach outperformed all existing methods in both metrics. The proposed approach could be used in a variety of application as in ranking, recommendation systems, and clustering. The significance of the proposed random walk model can be greatly seen as leveraging vertex features that could enhance the solution of the problem. If such features exist, the proposed model can show very interesting results.

Acknowledgment

The second author would like to acknowledge the full scholarship and financial support by King Saud University, Riyadh, Saudi Arabia.

References

[1] S. Agarwal, K. Branson, and S. Belongie, "Higher order learning with graphs," in *Proceedings of the 23rd international conference on Machine learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 17–24. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143847

[2] H. Liu, P. Le Pendu, R. Jin, and D. Dou, "A hypergraph-based method for discovering semantically associated itemsets," in *Proceedings of the 11th IEEE International Conference on Data Mining*, ser. ICDM '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 398–406. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2011.12

[3] C. Avin, Y. Lando, and Z. Lotker, "Radio cover time in hypergraphs," in *Proceedings of the 6th International Workshop on Foundations of Mobile Computing*, ser. DIALM-POMC '10. New York, NY, USA: ACM, 2010, pp. 3–12. [Online]. Available: http://doi.acm.org/10.1145/1860684.1860689

[4] D. Zhou, J. Huang, and B. Scholkopf, "Learning with hypergraphs: Clustering, classification, and embedding," *Advances in neural information processing systems*, vol. 19, p. 1601, 2007.

[5] S. Vempala, "Geometric random walks: a survey," *MSRI volume on Combinatorial and Computational Geometry*, 2005.

[6] L. Lovász, "Random walks on graphs: A survey," *Combinatorics, Paul erdos is eighty*, vol. 2, no. 1, pp. 1–46, 1993.

[7] E. Minkov and W. W. Cohen, "Learning to rank typed graph walks: local and global approaches," in *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, ser. WebKDD/SNA-KDD '07. New York, NY, USA: ACM, 2007, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/1348549.1348550

[8] M. Diligenti, M. Gori, and M. Maggini, "Learning web page scores by error back-propagation," in *Proceedings of the 19th international joint conference on Artificial intelligence*, ser. IJCAI'05. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005, pp. 684–689. [Online]. Available: http://dl.acm.org/citation.cfm?id=1642293.1642403

[9] A. Agarwal and S. Chakrabarti, "Learning random walks to rank nodes in graphs," in *Proceedings of the 24th international conference on Machine learning*, ser. ICML '07. New York, NY, USA: ACM, 2007, pp. 9–16. [Online]. Available: http://doi.acm.org/10.1145/1273496.1273498

[10] A. Agarwal, S. Chakrabarti, and S. Aggarwal, "Learning to rank networked entities," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '06. New York, NY, USA: ACM, 2006, pp. 14–23. [Online]. Available: http://doi.acm.org/10.1145/1150402.1150409

[11] M. Jamali and M. Ester, "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*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 397–406. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557067

[12] M. A. Tayebi, M. Jamali, M. Ester, U. Glässer, and R. Frank, "Crimewalker: a recommendation model for suspect investigation," in *Proceedings of the fifth ACM conference on Recommender systems*, ser. RecSys '11. New York, NY, USA: ACM, 2011, pp. 173–180. [Online]. Available: http://doi.acm.org/10.1145/2043932.2043965

[13] X. Li, X. Su, and M. Wang, "Social network-based recommendation: a graph random walk kernel approach," in *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, ser. JCDL '12. New York, NY, USA: ACM, 2012, pp. 409–410. [Online]. Available: http://doi.acm.org/10.1145/2232817.2232915

[14] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *Proceedings of the fourth ACM international conference on Web search and data mining*, ser. WSDM '11. New York, NY, USA: ACM, 2011, pp. 635–644. [Online]. Available: http://doi.acm.org/10.1145/1935826.1935914

[15] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *Proceedings of the Seventh International Conference on World Wide Web 7*, ser. WWW7. Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 1998, pp. 107–117.

[16] D. Aldous and J. Fill, "Reversible markov chains and random walks on graphs," 2002.

[17] J. Jarvis and D. R. Shier, "Graph-theoretic analysis of finite markov chains," *Applied mathematical modeling: a multidisciplinary approach*, 1999.

[18] D. Zhou, S. Orshanskiy, H. Zha, and C. Giles, "Co-ranking authors and documents in a heterogeneous network," in *Proceedings of the 7th IEEE International Conference on Data Mining*, 2007, pp. 739–744.

[19] L. Soulier, L. B. Jabeur, L. Tamine, and W. Bahsoun, "On ranking relevant entities in heterogeneous networks using a language-based model," *Journal of the American Society for Information Science and Technology*, vol. 64, no. 3, pp. 500–515, 2013. [Online]. Available: http://dx.doi.org/10.1002/asi.22762

[20] C. Berge, *Hypergraphs: combinatorics of finite sets*. North holland, 1984.

[21] J. Lee, M. Cho, and K. M. Lee, "Hyper-graph matching via reweighted random walks," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2011, pp. 1633–1640.

[22] C. Cooper, A. Frieze, and T. Radzik, "The cover times of random walks on random uniform hypergraphs," *Theoretical Computer Science*, no. 0, pp. –, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397513000820

[23] L. Lu and X. Peng, "High-ordered random walks and generalized laplacians on hypergraphs," in *Proceedings of the 8th international conference on Algorithms and models for the web graph*, ser. WAW'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 14–25. [Online]. Available: http://dl.acm.org/citation.cfm?id=2022148.2022150

[24] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, L. Zhang, and X. He, "Music recommendation by unified hypergraph: combining

social media information and music content," in *Proceedings of the international conference on Multimedia*, ser. MM '10. New York, NY, USA: ACM, 2010, pp. 391–400. [Online]. Available: http://doi.acm.org/10.1145/1873951.1874005

[25] L. Li and T. Li, "News recommendation via hypergraph learning: encapsulation of user behavior and news content," in *Proceedings of the sixth ACM international conference on Web search and data mining*, ser. WSDM '13. New York, NY, USA: ACM, 2013, pp. 305–314. [Online]. Available: http://doi.acm.org/10.1145/2433396.2433436

[26] W. Wang, F. Wei, W. Li, and S. Li, "Hypersum: hypergraph based semi-supervised sentence ranking for query-oriented summarization," in *Proceedings of the 18th ACM conference on Information and knowledge management*, ser. CIKM '09. New York, NY, USA: ACM, 2009, pp. 1855–1858. [Online]. Available: http://doi.acm.org/10.1145/1645953.1646248

[27] H.-K. Tan, C.-W. Ngo, and X. Wu, "Modeling video hyperlinks with hypergraph for web video reranking," in *Proceedings of the 16th ACM international conference on Multimedia*, ser. MM '08. New York, NY, USA: ACM, 2008, pp. 659–662. [Online]. Available: http://doi.acm.org/10.1145/1459359.1459453

[28] H. Deng, J. Han, B. Zhao, Y. Yu, and C. X. Lin, "Probabilistic topic models with biased propagation on heterogeneous information networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '11. New York, NY, USA: ACM, 2011, pp. 1271–1279. [Online]. Available: http://doi.acm.org/10.1145/2020408.2020600

[29] Y. Sun, Y. Yu, and J. Han, "Ranking-based clustering of heterogeneous information networks with star network schema," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 797–806. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557107

[30] A. Bellaachia and M. Al-Dhelaan, "Ne-rank: A novel graph-based keyphrase extraction in twitter," in *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01*, ser. WI-IAT '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 372–379. [Online]. Available: http://dl.acm.org/citation.cfm?id=2457524.2457617

[31] A. Bellaachia and M. Al-Dhelaan, "Learning from twitter hashtags: Leveraging proximate tags to enhance graph-based keyphrase extraction," in *Proceedings of the 2012 IEEE International Conference on Green Computing and Communications*, ser. GREENCOM '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 348–357. [Online]. Available: http://dx.doi.org/10.1109/GreenCom.2012.58

[32] C. Buckley and E. M. Voorhees, "Retrieval evaluation with incomplete information," in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '04. New York, NY, USA: ACM, 2004, pp. 25–32.

[33] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Proceedings of the Sixth International Conference on Data Mining*, ser. ICDM '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 613–622. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2006.70