Assessing and Ranking Structural Correlations in Graphs

Ziyu Guan[1], Jian Wu[2,*], Qing Zhang[3], Ambuj Singh[1], Xifeng Yan[1]

[1]Dept. of Computer Science
University of California
Santa Barbara, CA, 93106, USA
{ziyuguan, ambuj, xyan}@cs.ucsb.edu

[2]College of Computer Science
Zhejiang University
Hangzhou, 310027, China
wujian2000@zju.edu.cn

[3]TaoBao.com
99 Huaxing Road
Hangzhou, 310099, China
yunzheng@taobao.com

ABSTRACT

Real-life graphs not only have nodes and edges, but also have events taking place, e.g., product sales in social networks and virus infection in communication networks. Among different events, some exhibit strong correlation with the network structure, while others do not. Such structural correlation will shed light on viral influence existing in the corresponding network. Unfortunately, the traditional association mining concept is not applicable in graphs since it only works on homogeneous datasets like transactions and baskets.

We propose a novel measure for assessing such structural correlations in heterogeneous graph datasets with events. The measure applies hitting time to aggregate the proximity among nodes that have the same event. In order to calculate the correlation scores for many events in a large network, we develop a scalable framework, called gScore, using sampling and approximation. By comparing to the situation where events are randomly distributed in the same network, our method is able to discover events that are highly correlated with the graph structure. gScore is scalable; it can accomplish one proximity estimation in 0.2 second on a graph with 10 million nodes. gScore was successfully applied to the co-author DBLP network and social networks extracted from TaoBao.com, the largest online shopping network in China, with many interesting discoveries.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - Data Mining

General Terms

Algorithms, Experimentation

Keywords

Graph, Structural Correlation, Hitting Time

*Corresponding author

SIGMOD'11, June 12–16, 2011, Athens, Greece.

1. INTRODUCTION

The rise of the web, social networks, and bioinformatics has presented scientists with numerous graphs, each consisting of millions of nodes and edges. Hidden in these large datasets are the answers to important questions in networking, sociology, business, and biology. These graphs not only have topological structures, but also contain events/activities that occurred on their nodes. For example, an eBay customer could sell or bid a product. A Facebook user could play a Zynga game with his/her friends. This complex combination raises new research problems in graph data analysis [29, 22, 8, 2].

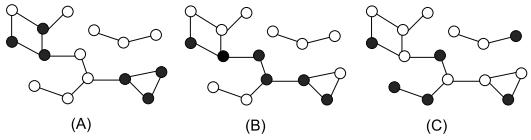

Figure 1: Structural Correlation

Among different events taking place in a network, some exhibit strong correlation with the network structure, while others do not. Such structural correlation might shed light on viral influence existing in the corresponding network, which is the key to many research problems in product marketing [7], online advertisement [3], and recommendation [16]. Figure 1 shows the distribution of three types of events over the same graph. We can easily incarnate Figure 1 into different application scenarios. For example, it could be three kinds of products that were bought by members in a social network. Dark nodes in Figure 1(A), 1(B) and 1(C) represent members who purchased products A, B and C, respectively. Intuitively, Figure 1 shows that in this network, people who bought product A (or B) are closer to each other. In contrary, black nodes for C seem to be randomly distributed. In this scenario, the network would be suitable for promoting A and B and we can promote A and B among people who have not bought them. While it is hard to derive deterministic relationship between sales and network structure, it is possible to study how the sales is correlated to the structure. In fact, one can raise several interesting questions related to structure and events distributed over structure:

1. In regard to the sales of products A and B, which one is more related to the underlying social network?
2. Given two networks G and G' for the same group of users, e.g., their email network and Facebook network, is the sales of product A more related to G than G'?

3. If we have snapshots of network G during different periods, can we measure how product A was dispersed over the network over time? Was it purchased by a small circle of friends at the very beginning?

In order to answer the above questions, we need to address the following research problems:

1. How to define and measure the correlation between the graph structure and events?

2. How to compute the measure efficiently in large graphs, if we want to rank all events according to the measure?

Unfortunately, the classic association mining concept is not applicable in this setting since it only works on homogeneous datasets like transactions and baskets [1, 30]. In this paper, we propose a novel measure to assess structural correlations in a graph. The measure aggregates the proximity among nodes on which the same event has occurred, using a proximity function such as hitting time [20]. We develop an efficient computation framework, gScore (**G**raph **S**tructural **Cor**relation **E**stimation), to quickly calculate correlation scores in large scale networks. By estimating the deviation from the expected correlation score of a random situation, our method is able to discover the events of nodes that are highly correlated with the graph structure.

Our contributions. We propose a novel concept, structural correlation, to measure how an event is distributed over a graph and address a key research problem in analyzing the relation between structure and contents. While many studies have demonstrated that social links could significantly influence the behavior of human beings [5, 17, 7], we suspect that such influence should be further scrutinized for more fine-grained knowledge: *in which kind of social links (e.g., phone network, email network, employee network, friend network, etc) social influence is observed, and how strong, and for which kind of behaviors (e.g.,shopping, hobby, interest, and opinion).* In this study, we quantify the correlation between link structure and human behaviors, and make different behaviors' correlations comparable using statistical significance. We discover that the correlation actually fluctuates dramatically with regard to link types, event types, and time, implying a need to further examine the cause of the fluctuation. Note that in this work, we are not going to perform a causality study between correlation and influence [2].

We systematically introduce a framework to define and measure structural correlations in graphs. The principle is to aggregate the proximity among nodes which have the same event and compare the aggregated proximity against the situations where these events are randomly distributed in the graph. This framework can integrate various graph proximity measures [6] such as hitting time [20], personalized PageRank [24, 14] and Katz [15].

We take hitting time as an example and propose a modified version named *Decayed Hitting Time* (**DHT**) to better and faster calculate structural correlation. Scalable algorithms are developed using sampling and approximation techniques to calculate DHT for individual nodes and the average DHT for all the nodes which share the same event. We investigate the expectation and variance of the correlation when an event is randomly distributed over a graph and derive several important properties. These properties can help us to quickly estimate the deviation from random cases, thus making online computation of structural correlations possible. Our algorithm was tested in real networks including co-author DBLP network and social networks extracted from TaoBao.com, the largest online shopping network in China, with many exciting discoveries. Finally, we show that our method can accomplish one DHT estimation in 0.2 second on a graph with 10 million nodes, indicating that it is very scalable.

2. PROBLEM FORMULATION

An attributed graph $G = (V, E)$ has an event set Q. The event set of a node v is written as $Q(v) \subseteq Q$. In this work, we consider undirected and unweighted graphs. Nevertheless, the proposed measure and algorithms can be easily generalized to weighted and/or directed graphs.

Suppose there is an event q (e.g. purchasing a specific product) taking place in G. Each node v can take two values in terms of q: $f_q(v) = 1$ if $q \in Q(v)$; otherwise, $f_q(v) = 0$. Let $m = \sum_v f_q(v)$ denote the number of nodes where q occurred. Let $n = |V|$ be the number of nodes in G. We could formulate the following two research problems:

Problem 1: Determine whether there is a correlation between q and the graph structure of G. If not, q is just randomly distributed in G.

Its accompanying ranking problem is as follows:

Problem 2: Given a set of different events $Q = \{q_i\}$ on G, rank $\{q_i\}$ according to their correlation strength with respect to the graph structure of G.

In order to address the above problems, we need a measure that captures the distribution of an event over a graph, and evaluates whether (and to what degree) the event is correlated with the graph structure. A simple measure could be to assess the probability that a node's neighbors have q given that node having q. However, such a measure has drawbacks. Figure 2 shows a star structure in a graph, where an influential person (the center node) recommended a product to friends. These friends then buy the product in this network. The center node might have bought the product through other channels, e.g., but never recorded. The above measure, which returns "0", cannot catch this situation. Therefore, we need a more principled measure which can reflect the topological proximity between nodes.

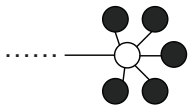

Figure 2: Limitation of assessing the probability that neighbors have event q given a node having q.

3. STRUCTURAL CORRELATIONS

Given m nodes where event q took place, one would decide if the occurrence of q is related to the graph structure or not. Intuitively, if these m nodes are close to one another, then the correlation is high. Otherwise, it will be more similar to a random situation where q is randomly distributed in G. Therefore, we propose using the average proximity between one node in those m nodes and the remaining $m - 1$ nodes to assess the **structural correlation** between q and G,

$$\rho(V_q) = \frac{\sum_{v \in V_q} s(v, V_q \setminus \{v\})}{|V_q|}, \quad (1)$$

where V_q is the set of m nodes on which q occurred and $s(v, V_q \setminus \{v\})$ is the closeness of the remaining nodes to v. $s(\cdot)$ can be any graph proximity measure that measures the proximity between a node and a set of nodes in a graph topology notion. We could rewrite $s(\cdot)$ by decomposing the contribution of each node in $V_q \setminus \{v\}$,

$$s(v, V_q \setminus \{v\}) = \sum_{u \in V_q \setminus \{v\}} s_q(v \rightsquigarrow u), \quad (2)$$

where $s_q(v \rightsquigarrow u)$ is the contribution of u to $s(v, V_q \setminus \{v\})$. Eq. (2) is naturally defined for pairwise measures, but not for holistic measures. We assume holistic measures can be decomposed to facilitate analysis. $s_q(v \rightsquigarrow u)$ could be either invariant to q, or related to the distribution of q in G. For the first case, one can choose personalized PageRank score of u with respect to v, or the Katz measure [15]. Both of them are invariant to q. Readers are referred to [23] for an introduction of various graph proximity functions. For the second case, $s_q(v \rightsquigarrow u)$ is determined not only by the proximity between v and u, but also the other nodes in $V_q \setminus \{v\}$. One example is hitting time: starting from node v, it calculates the expected number of steps needed to reach one of nodes in $V_q \setminus \{v\}$. Hence, it is a holistic measure. For different distribution of q, $p_q(v \rightsquigarrow u)$ is not fixed. In this work, we will focus on hitting time. Nevertheless, our framework is applicable to other proximity measures too.

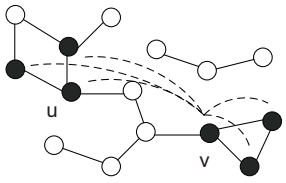

Figure 3: Measuring Structural Correlations

Figure 3 illustrates the idea of measuring the proximity between one node v and the remaining nodes that have the same event. Intuitively, if those nodes are not close to v, e.g., randomly distributed, then $s(v, V_q \setminus \{v\})$ will be close to an average value calculated from random cases. This indicates that we should also consider the ρ value for random cases. Let $\rho(\widehat{V_q})$ denote the correlation score for a randomly selected set $\widehat{V_q}$ of m nodes where $m = |\widehat{V_q}| = |V_q|$. As m increases, there will be an increasing chance for m randomly selected nodes to be close to one another (in the extreme case $m = n$, nodes are obviously very close). Thus, given two events q and q', we cannot simply decide q is more correlated with G if $\rho(V_q) > \rho(V_{q'})$. We should estimate the "deviation" of $\rho(V_q)$ from the expectation of $\rho(\widehat{V_q})$. Since it is hard to obtain the distribution of $\rho(\widehat{V_q})$, we propose to estimate the expectation and variance of $\rho(\widehat{V_q})$ (denoted by $E[\rho(\widehat{V_q})]$ and $Var[\rho(\widehat{V_q})]$, respectively), and compare $\rho(V_q)$ with them. More details on this are presented in Section 5.2. We refer this framework as gScore (**G**raph **S**tructural **Cor**relation **E**stimation). In this work, we instantiate it using hitting time discussed as follows.

3.1 Random Walk and Hitting Time

A *random walk* is defined as follows: we start from a node v and perform a random move among neighboring nodes. If in the t-th step we are at node v_t, we move to a neighbor of v_t with probability $\frac{1}{d(v_t)}$, where $d(v)$ is the degree of node v. The sequence $\{v_t\}$ is a Markov chain. Let $\mathbf{A}$ denote the adjacency matrix of G. A_{ij} equals 1 if there is an edge between node v_i and v_j, and 0 otherwise. We use $\mathbf{P} = [p_{ij}]_{n \times n}$ to denote the transition probability matrix of the random walk. We have $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = d(v_i)$. We use $\Pr(pa_{1 \rightsquigarrow l}) = p_{12}p_{23}\dots p_{(l-1)l}$ to denote the probability that a random walk takes path $v_1, v_2, \dots, v_l$, starting from v_1 (also called the probability of that path).

Let B be a subset of V. The hitting time (also called access time) [20] from a node v_i to B is defined as the expected number of steps before a node $v_j \in B$ is visited in a random walk starting from v_i. Let $h(v_i, B)$ denote the hitting time from v_i to B and x_t denote the position of the random walk at time step t. By definition, we have

$$h(v_i, B) = \sum_{t=1}^{\infty} t \Pr(T_B = t | x_0 = v_i), \quad (3)$$

where $\Pr(T_B = t | x_0 = v_i)$ is the probability that the random walk starting from v_i first hits a node in B after t steps. One can easily derive [20]

$$h(v_i, B) = \sum_{v_k \in V} p_{ik} h(v_k, B) + 1. \quad (4)$$

Eq. (4) expresses a one step look-ahead property of hitting time. The expected time to reach a node in B from v_i is equivalent to one step plus the mean of hitting times of v_i's neighbors to a node in B. By the definition of hitting time (i.e. Eq. (3)), we have $h(v_i, B) = 0$ for $v_i \in B$. From the above analysis, we obtain a linear system for computing hitting time from all nodes to B:

$$\begin{cases} h(v_i, B) = 0 & v_i \in B \\ h(v_i, B) = \sum_{v_k \in V} p_{ik} h(v_k, B) + 1 & v_i \notin B \end{cases}. \quad (5)$$

It can be shown that this linear system has a unique solution [20]. Hitting time can be used to measure a notion of asymmetrical proximity among nodes on a graph in terms of graph topology, with applications such as Web query suggestion [21], recommendation [4] and link prediction [23].

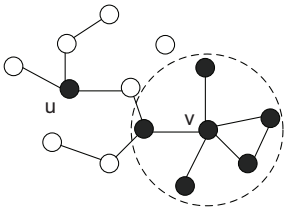

Figure 4: Hitting Time

Figure 4 shows one example of hitting time. For node v, its hitting time with regard to the set of other dark nodes is 1. In comparison with other proximity measures, e.g., the average distance between v and other dark nodes, hitting time will be less influenced by remote nodes and anomalies, and focus on close neighbors. While hitting time has this nice property, it has one drawback: its value range is between 1 and $+\infty$. That is, if there is no connection between two nodes u and v, it is possible to have infinite hitting time because u can never be reached. This could also result in infinite value of $\rho(V_q)$ (Eq. (1)). In this work, we propose

decayed hitting time, which inverses the range of $[1, +\infty)$ to $[0, 1]$. The meaning of hitting time also changes slightly.

3.2 Decayed Hitting Time

Inspired by hitting time, we propose a new graph proximity measure called *Decayed Hitting Time* (DHT), which is defined as follows

$$\tilde{h}(v_i, B) = \sum_{t=1}^{\infty} e^{-(t-1)} \Pr(T_B = t | x_0 = v_i). \quad (6)$$

For DHT, when nodes in B are close to v_i, $\tilde{h}(v_i, B)$ will be high. The reason to substitute the exponentially decaying term for the number of steps is two-fold. First, in our problem setting, we want to emphasize the importance of neighbors and discount the importance of longer range weak relationships. Second, as we will show later, such a definition can facilitate the approximation of the measure. Although hitting time is shown to be empirically effective in practical applications [4, 23], solving a linear system for a large scale graph can be computationally expensive (i.e. $O(n^3)$). Motivated by this issue, Sarkar *et al.* proposed *truncated hitting time* [26] and developed sampling techniques to approximate hitting time [27]. However, a drawback of truncated hitting time is that it sets an arbitrary upper bound (the truncated path length) for actual hitting time. It is not accurate since the bound treats all of hitting paths with length greater than the bound equally. The definition in Eq. (6) allows us to derive proper lower bounds for DHT.

In terms of decayed hitting time, $s(v, V_q \setminus \{v\})$ can be instantiated as $\tilde{h}(v, V_q \setminus \{v\})$. In the following discussion, we will first discuss how to compute DHT and approximate it efficiently in large scale graphs. Then we turn to the issue of efficiently estimating $\rho(V_q)$, $E[\rho(\widehat{V_q})]$ and $Var[\rho(\widehat{V_q})]$.

4. COMPUTING DECAYED HITTING TIME

Typical social networks usually contain millions of nodes. Thus, computational efficiency is a critical issue. We present techniques for efficiently estimating DHT.

4.1 Exact Solution

For DHT from any node v_i to a subset $B \subset V$, we can also derive a linear system. For $v_i \notin B$, we have

$$\begin{aligned} \tilde{h}(v_i, B) &= \sum_{t=1}^{\infty} e^{-(t-1)} \sum_{v_k \in V} p_{ik} \Pr(T_B = t-1 | x_0 = v_k) \\ &= e^{-1} \sum_{v_k \in V} p_{ik} \tilde{h}(v_k, B) + \sum_{v_k \in B} p_{ik}. \quad (7) \end{aligned}$$

By the definition of DHT, we have $\tilde{h}(v_i, B) = 0$ for $v_i \in B$. Then the linear system is

$$\begin{cases} \tilde{h}(v_i, B) = 0 & v_i \in B \\ \tilde{h}(v_i, B) = e^{-1} \sum_{v_k \in V} p_{ik} \tilde{h}(v_k, B) + \sum_{v_k \in B} p_{ik} & v_i \notin B \end{cases}. \quad (8)$$

Solving this linear system, we can derive $\tilde{h}(v_i, B)$ for all $v_i \notin B$. However, solving a linear system can be computationally expensive. Moreover, examining our correlation measure in Eq. (1), one can find that for each target node set (i.e. $V_q \setminus \{v\}$) we only need the DHT from one node v. Computing $\tilde{h}(v_i, V_q \setminus \{v\})$ for all $v_i \notin V_q \setminus \{v\}$ is wasteful.

4.2 Iterative Approximation

Suppose we want to compute the DHT from v_i to a node set B. Examining the definition in Eq. (6), we know that if we can obtain $\Pr(T_B = t | x_0 = v_i), t = 1, 2, \ldots$, then we can compute $\tilde{h}(v_i, B)$. Let $\Pr(T_{\bar{B}, v_j} = t | x_0 = v_i)$ be the probability that the random walk starting from v_i hits v_j after t steps without visiting any node in B. In other words, it means $x_i \notin B$ for $i \in \{1, 2, \ldots, t-1\}$. Therefore, we have

$$\Pr(T_B = t | x_0 = v_i) = \sum_{v_j \in B} \Pr(T_{\bar{B}, v_j} = t | x_0 = v_i).$$

The problem becomes how to compute $\Pr(T_{\bar{B}, v_j} = t | x_0 = v_i)$. In particular, we have

$$\Pr(T_{\bar{B}, v_j} = t | x_0 = v_i) = \sum_{v_k \notin B} \Pr(T_{\bar{B}, v_k} = t-1 | x_0 = v_i) p_{kj},$$

which implies that one can first get to $v_k \notin B$ using $t-1$ steps (without visiting any $v_j \in B$), and then move from v_k to v_j with probability p_{kj}. It takes the sum over all possible v_k's. Therefore, we can derive the following iterative computation method: let $\mathbf{P}_B$ be a modification of the original transition probability matrix $\mathbf{P}$ where rows corresponding to the nodes in B are set to all zeros. Let $\mathbf{u}_t$ be a $n \times 1$ vector containing $\Pr(T_{\bar{B}, v_k} = t | x_0 = v_i)$ as its k-th element. $\mathbf{u}_0$ is the vector with i-th element set to 1 and all other elements to 0. One can easily verify $\mathbf{u}_t = \left(\mathbf{P}_B^T\right)^t \mathbf{u}_0$. In fact, $\mathbf{P}_B$ and $\mathbf{u}_0$ define the corresponding random walk model for computing DHT from v_i to B. Letting $\mathbf{z}_B$ be the vector with elements corresponding to nodes in B set to 1 and all other elements to 0, we can rewrite Eq. (6) as

$$\begin{aligned} \tilde{h}(v_i, B) &= e^0 \mathbf{z}_B^T \mathbf{u}_1 + e^{-1} \mathbf{z}_B^T \mathbf{u}_2 + \ldots \\ &= e^0 \mathbf{z}_B^T \mathbf{P}_B^T \mathbf{u}_0 + e^{-1} \mathbf{z}_B^T \mathbf{P}_B^T \mathbf{u}_1 + \ldots \end{aligned}$$

We can iteratively compute $\mathbf{u}_1, \mathbf{u}_2, \ldots$ and accumulate elements corresponding to nodes in B from these vectors (multiplied by $e^0, e^{-1}, \ldots$ respectively). If we stop after a number of iterations, it results in an approximation of the actual DHT. In the following discussion, we will derive the upper bound and lower bound for DHT from such an approximation. From now on, we use $d_B(v_k)$ to denote the number of neighbors of v_k which are in B and $\lambda_{kB} = d_B(v_k)/d(v_k)$ to denote the corresponding fraction.

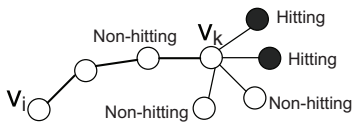

Figure 5: Upper Bound for One Path

LEMMA 1. *Let $pa^l_{i \rightsquigarrow k}$ be a length-l path from v_i to v_k which has not yet hit any node in B. Let $\Pr(pa^l_{i \rightsquigarrow k})$ be the probability that the random walk takes this path. We call paths sharing $pa^l_{i \rightsquigarrow k}$ as a prefix subpaths of $pa^l_{i \rightsquigarrow k}$. The contribution to $\tilde{h}(v_i, B)$ of all subpaths of $pa^l_{i \rightsquigarrow k}$ which finally hit a node in B is upper bounded by $\lambda_{kB} \Pr(pa^l_{i \rightsquigarrow k}) e^{-l} + (1 - \lambda_{kB}) \Pr(pa^l_{i \rightsquigarrow k}) e^{-(l+1)}$.*

PROOF. According to the Markov property of random walks, the sum of probabilities of all paths sharing the same

prefix equals to the probability of that prefix. Therefore, when the random walk follows $pa^l_{i\rightsquigarrow k}$ to get v_k, it will further distribute $\Pr(pa^l_{i\rightsquigarrow k})$ to $pa^l_{i\rightsquigarrow k}$'s subpaths and some subpaths will eventually hit a node in B. By querying the neighbors of v_k, we know the probability $\lambda_{kB}\Pr(pa^l_{i\rightsquigarrow k})$ will be distributed on length-$(l+1)$ hitting subpaths and the probability those length-$(l+1)$ non-hitting subpaths account for (i.e. $(1-\lambda_{kB})\Pr(pa^l_{i\rightsquigarrow k})$) will be further distributed on their subpaths with longer lengths, as illustrated in Figure 5. We can apply the above analysis iteratively to the length-$(l+1)$ non-hitting subpaths of $pa^l_{i\rightsquigarrow k}$ to obtain tighter upper bound, but it is equivalent to performing one more iteration. Thus, we can just optimistically assume the whole $(1-\lambda_{kB})\Pr(pa^l_{i\rightsquigarrow k})$ is distributed on length-$(l+2)$ hitting subpaths. Then, the conclusion follows. □

THEOREM 1. *Let $\tilde{h}_t(v_i,B) = e^0\mathbf{z}_B^T\mathbf{u}_1 + \cdots + e^{-(t-1)}\mathbf{z}_B^T\mathbf{u}_t$ be the approximation of $\tilde{h}(v_i,B)$ after $\mathbf{u}_t$ is computed. We have $\tilde{h}_t(v_i,B) \le \tilde{h}(v_i,B) \le \tilde{h}_t(v_i,B) + \sum_{v_k\notin B}\Pr(T_{\bar{B},v_k} = t|x_0=v_i)\left(\lambda_{kB}e^{-t} + (1-\lambda_{kB})e^{-(t+1)}\right)$.*

PROOF. Omitted. □

4.3 A Sampling Algorithm for $\tilde{h}(v_i,B)$

We propose a standard Monte Carlo sampling method for estimating $\tilde{h}(v_i,B)$. The straightforward sampling scheme can be as follows: we run c independent random walk simulations from v_i and in each random walk we stop when a node in B is encountered. Suppose these random walks' path lengths are $l_1,\ldots,l_c$. Then we use the average $\overline{\tilde{h}(v_i,B)} = \sum_{j=1}^{c} e^{-(l_j-1)}/c$ as the estimate of $\tilde{h}(v_i,B)$. However, this scheme is not a wise choice due to the following two reasons: 1) if we cannot reach any node in B from v_i, the random walk will never stop; 2) for a large scale graph, if we do not impose a maximum number of steps that a random walk can take, the sampling algorithm will be time consuming. In fact, since we adopt an exponentially damping factor (i.e. $e^{-(t-1)}$), the contribution of long paths are negligible.

With the above concerns, we adopt a sampling scheme as follows: we run c independent random walk simulations from v_i and in each random walk we stop when a node in B is visited or a maximum number of s steps is reached. Suppose out of c runs c_h random walks hit a node in B and the corresponding path lengths are $l_1,\ldots,l_{c_h}$, respectively. Let $\bar{c} = c - c_h$ be the number of random walks which reach s steps and do not hit any node in B. We can provide bounds for $\overline{\tilde{h}(v_i,B)}$.

THEOREM 2. *If we run c independent random walk simulations for estimating $\tilde{h}(v_i,B)$ and impose a maximum number of s steps for each random walk, we have $\sum_{j=1}^{c_h} e^{-(l_j-1)}/c \le \overline{\tilde{h}(v_i,B)} \le (\sum_{j=1}^{c_h} e^{-(l_j-1)} + e^{-s}\bar{c})/c$.*

PROOF. In the sampling scheme having the constraint of maximum number of steps, only c_h random walks have certain contribution to $\overline{\tilde{h}(v_i,B)}$. Hence, we turn to the bounds for the contribution of remaining $\bar{c}$ random walks which do not hit any node in B. For each of these $\bar{c}$ random walks, the contribution to $\overline{\tilde{h}(v_i,B)}$ is upper bounded by e^{-s} (i.e. hitting a node in B at $(s+1)$-th step). Aggregating those $\bar{c}$ random walks, we have $\overline{\tilde{h}(v_i,B)} \le (\sum_{j=1}^{c_h} e^{-(l_j-1)} + e^{-s}\bar{c})/c$. A lower bound for the contribution of those $\bar{c}$ random walks is 0. This leads to $\sum_{j=1}^{c_h} e^{-(l_j-1)}/c \le \tilde{h}(v_i,B)$. □

Algorithm 1: Sampling Approximation

Input: $\mathbf{P}$: transition probability matrix, v_i: start node, B: target node set, s: maximum # of walk steps, c: number of random walks

Output: low, up: lower and upper bounds of $\overline{\tilde{h}(v_i,B)}$

```
1  begin
2      low, up = 0
3      for k = 1 to c do
4          v = v_i, step = 0
5          while v ∉ B do
6              if step ≥ s then
7                  break
8              end
9              Randomly set v to its neighbors according to
               P.
10             step = step + 1
11         end
12         if v ∈ B then
13             low = low + e^{-(step-1)}
14         else
15             up = up + e^{-s}
16         end
17     end
18     up = (low + up)/c, low = low/c
19 end
```

We use $\tilde{h}'_{iB}$ and $\tilde{h}''_{iB}$ to represent the above lower and upper bounds for $\overline{\tilde{h}(v_i,B)}$, respectively. The following theorem provides the lower bound for the number of samples c required in order to obtain an ϵ-correct answer with respect to $[\tilde{h}'_{iB}, \tilde{h}''_{iB}]$ with probability $1-\delta$.

THEOREM 3. *Suppose we run c independent random walk simulations for estimating $\tilde{h}(v_i,B)$ and impose a maximum number of s steps for each random walk. In order to obtain $\Pr(\tilde{h}'_{iB} - \epsilon \le \tilde{h}(v_i,B) \le \tilde{h}''_{iB} + \epsilon) \ge 1-\delta$, the number of samples c should be at least $\frac{1}{2\epsilon^2}\ln(\frac{2}{\delta})$.*

PROOF. Omitted. □

We summarize the sampling algorithm in Algorithm 1.

4.4 Complexity

Suppose we use adjacency lists to store graphs and matrices. The space complexity of the iterative and sampling algorithms is $O(|E|)$. The major time costing parts of the iterative algorithm are the iterative matrix-vector multiplication and the construction of $\mathbf{P}_B$. The corresponding time complexity is $O(t|E|)$, where t is the number of iterations. For the sampling algorithm, the major time cost is the membership judgement for v to B (line 5 in Algorithm 1). We can either sort B and use binary search, or build index array for B. The corresponding time costs are $O(cs\log|B| + |B|\log|B|)$ and $O(cs+|V|)$, respectively.

5. ASSESSING STRUCTURAL CORRELATIONS

This section provides methods for efficient estimation of $\rho(V_q)$, $E[\rho(\widehat{V_q})]$ and $Var[\rho(\widehat{V_q})]$. We also present our method for estimating the deviation of $\rho(V_q)$ from $E[\rho(\widehat{V_q})]$.

5.1 Estimating $\rho(V_q)$

To compute $\rho(V_q)$, we need to compute $\tilde{h}(v_i, V_q \setminus \{v_i\})$ for all $v_i \in V_q$. However, for large scale graphs, V_q may also have a large size, posing a challenge for efficient computation of $\rho(V_q)$. Since those $\tilde{h}$'s form a finite population, we can use sampling techniques to efficiently estimate $\rho(V_q)$ [12]. Specifically, we randomly select c' nodes from V_q, denoted by $v_1, \ldots, v_{c'}$, to estimate their DHTs to the remaining nodes and take the average $\overline{\rho(V_q)}$ as an estimate of $\rho(V_q)$. Here we can use either iterative algorithm or sampling algorithm for estimating each $\tilde{h}(v_i, V_q \setminus \{v_i\})$. If the iterative algorithm is used, from Theorem 1 we obtain bounds for each $\tilde{h}(v_i, V_q \setminus \{v_i\})$ in the sample set. Aggregating those bounds, we can get lower and upper bounds for $\overline{\rho(V_q)}$. Following the same manner for the proof of Theorem 3, we can obtain the lower bound for c' in order to obtain an ϵ'-correct answer with respect to bounds for $\overline{\rho(V_q)}$. We omit the details due to space limitation. When the sampling algorithm is used, we provide the lower bound for c' in order to obtain an ϵ'-correct answer with respect to bounds for $\overline{\rho(V_q)}$ in the following theorem.

THEOREM 4. *Suppose we randomly select c' nodes from V_q to estimate their DHTs to the remaining nodes and take the average $\overline{\rho(V_q)}$ as an estimate of $\rho(V_q)$. For the sake of clarity, let $B_i = V_q \setminus \{v_i\}$. Suppose we have used Algorithm 1 to obtain an ϵ-correct answer for each $\tilde{h}(v_i, B_i)$ ($i = 1, \ldots, c'$) with respect to $[\tilde{h}'_{iB_i}, \tilde{h}''_{iB_i}]$ by setting the number of samples $c \geq \frac{1}{2\epsilon^2}\ln(\frac{2}{\delta})$. In order to obtain $\Pr(\sum_{i=1}^{c'} \tilde{h}'_{iB_i}/c' - \epsilon - \epsilon' \leq \rho(V_q) \leq \sum_{i=1}^{c'} \tilde{h}''_{iB_i}/c' + \epsilon + \epsilon') \geq 1 - \delta'$, the number of samples c' should satisfy $(1-\delta)^{c'}(1 - 2e^{-2c'\epsilon'^2}) \geq 1 - \delta'$.*

PROOF. We have $\Pr(\tilde{h}'_{iB_i} - \epsilon \leq \tilde{h}(v_i, B_i) \leq \tilde{h}''_{iB_i} + \epsilon) \geq 1 - \delta$ for $i = 1, \ldots, c'$. Notice $\overline{\rho(V_q)} = \sum_{i=1}^{c'} \tilde{h}(v_i, B_i)/c'$. Multiplying those probabilities inequalities together, we obtain $\Pr(\sum_{i=1}^{c'} \tilde{h}'_{iB_i}/c' - \epsilon \leq \overline{\rho(V_q)} \leq \sum_{i=1}^{c'} \tilde{h}''_{iB_i}/c' + \epsilon) \geq (1-\delta)^{c'}$. Since $0 \leq \tilde{h}(v_i, B_i) \leq 1$ for $i = 1, \ldots, c'$, according to Hoeffding's inequality for finite populations [12] we know $\Pr(|\overline{\rho(V_q)} - \rho(V_q)| \leq \epsilon') \geq 1 - 2e^{-2c'\epsilon'^2}$. Combining the above two probability inequalities, we have $\Pr(\sum_{i=1}^{c'} \tilde{h}'_{iB_i}/c' - \epsilon - \epsilon' \leq \rho(V_q) \leq \sum_{i=1}^{c'} \tilde{h}''_{iB_i}/c' + \epsilon + \epsilon') \geq (1-\delta)^{c'}(1 - 2e^{-2c'\epsilon'^2})$. Setting $(1-\delta)^{c'}(1 - 2e^{-2c'\epsilon'^2}) \geq 1 - \delta'$, we get the inequality c' should satisfy. Note δ should be large enough so that $(1-\delta)^{c'}(1-2e^{-2c'\epsilon'^2})$ can go beyond $1-\delta'$ as c' increases. □

5.2 Estimating the Deviation of $\rho(V_q)$ from Random Cases

As aforementioned in Section 3, we cannot just rely on $\rho(V_q)$ to measure the correlation between q and the graph structure. We should measure the deviation of $\rho(V_q)$ from the expected ρ value of a set of randomly selected m nodes ($m = |V_q|$), i.e., $E[\rho(\widehat{V_q})]$, in order to distinguish structural correlations from random results. In particular, we have

$$E[\rho(\widehat{V_q})] = \frac{\sum_{V_m \subseteq V} \rho(V_m)}{C_n^m}, \tag{9}$$

where V_m is any set of m nodes. The ideal solution is to obtain the distribution of $\rho(\widehat{V_q})$ and use the ratio of the number of node sets with size m whose ρ values are greater than or equal to $\rho(V_q)$ to C_n^m as the significance score for q. However, for a large scale graph it is very hard to get the distribution since C_n^m is very large. Here we propose an approximation method. Notice $\rho(\widehat{V_q})$ is defined as the average of $\tilde{h}(v_i, \widehat{V_q} \setminus \{v_i\})$ where $v_i \in \widehat{V_q}$. If we assume these $\tilde{h}$'s are independent, according to Central Limit Theorem, $\rho(\widehat{V_q})$ can be approximated by a normal distribution, where $Var[\rho(\widehat{V_q})] = Var[\tilde{h}(v_i, \widehat{V_q} \setminus \{v_i\})]/|\widehat{V_q}|$. If we obtain $E[\rho(\widehat{V_q})]$ and $Var[\rho(\widehat{V_q})]$, we can calculate the **adjusted structural correlation** $\tilde{\rho}$ for q as follows

$$\tilde{\rho}(V_q) = \frac{\rho(V_q) - E[\rho(\widehat{V_q})]}{\sqrt{Var[\rho(\widehat{V_q})]}}. \tag{10}$$

Note that this idea is similar to using Z scores to assess the significance of data mining results [11]. Eq. (10) can be used to derive the significance of q for a hypothesis that q is not randomly distributed over G. In the following we provide efficient methods for estimating $E[\rho(\widehat{V_q})]$ and $Var[\rho(\widehat{V_q})]$.

Intuitively, $E[\rho(\widehat{V_q})]$ would increase as m increases. We actually can prove the monotonic property of $E[\rho(\widehat{V_q})]$.

LEMMA 2. *Given an arbitrary node v_i and an arbitrary node set B ($v_i \notin B$), for any $v_k \in V \setminus (B \cup \{v_i\})$ we have $\tilde{h}(v_i, B) \leq \tilde{h}(v_i, B \cup \{v_k\})$.*

PROOF. We can interpret $\tilde{h}(v_i, B)$ as the sum of paths' probabilities weighted by corresponding $e^{-(\text{length}-1)}$ over all paths which start from v_i and first hit a node $v_j \in B$ at the end. We can divide the paths involved in the computation of $\tilde{h}(v_i, B)$ into two categories: 1) those containing v_k as intermediate nodes; 2) those not containing v_k. Paths in category 2 will be directly included in the computation of $\tilde{h}(v_i, B \cup \{v_k\})$ and their contribution will remain the same. For paths in category 1, when computing $\tilde{h}(v_i, B \cup \{v_k\})$ all category-1 paths will be reduced to their prefix paths down to the first occurrences of v_k. Due to the Markov property of random walks, the sum of probabilities of all paths sharing the same prefix equals to the probability of that prefix. Hence, the sum of probabilities of those prefix paths must be greater than or equal to that of all category-1 paths. Furthermore, now those prefix paths' probabilities are weighted by higher weights due to length reduction. Based on the above analysis, we can safely conclude $\tilde{h}(v_i, B) \leq \tilde{h}(v_i, B \cup \{v_k\})$. □

THEOREM 5. *Suppose $1 < m < n$. $E[\rho(\widehat{V_q})]$ will monotonically increase as $m = |\widehat{V_q}|$ increases.*

PROOF. We denote $\widehat{V_q}$ as $\widehat{V}_m$ using the number of nodes in $\widehat{V_q}$. We show that for an arbitrary m satisfying $1 < m < n$, we have $E[\rho(\widehat{V}_m)] \leq E[\rho(\widehat{V}_{m+1})]$. Recall that $E[\rho(\widehat{V}_m)] = \sum_{V_m} \rho(V_m)/C_n^m$. We can transform $E[\rho(\widehat{V}_m)]$ as follows

$$E[\rho(\widehat{V}_m)] = \frac{\sum_{V_m} \frac{\sum_{v_i \in V_m} \tilde{h}(v_i, V_m \setminus \{v_i\})}{m}}{C_n^m} = \frac{\sum_{v_i, V_{m-1}} \tilde{h}(v_i, V_{m-1})}{C_n^1 C_{n-1}^{m-1}} \tag{11}$$

where $v_i \notin V_{m-1}$. It means $E[\rho(\widehat{V}_m)]$ is the expected DHT from any v_i to any set of $m-1$ nodes which does not contain v_i. Similarly, we can rewrite $E[\rho(\widehat{V}_{m+1})]$ as

$$E[\rho(\widehat{V}_{m+1})] = \frac{\sum_{v_i, V_m} \tilde{h}(v_i, V_m)}{C_n^1 C_{n-1}^m}. \tag{12}$$

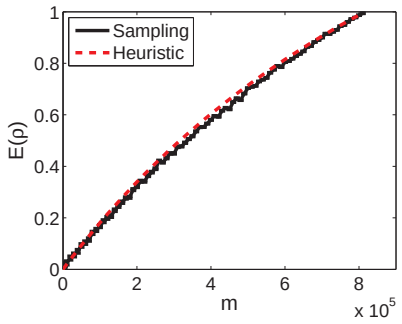

Figure 6: Comparison of sampling and geometric distribution heuristic for estimating $E[\rho(\widehat{V}_m)]$.

For each pair (v_i, V_{m-1}), we have $n-m$ corresponding pairs of the form $(v_i, V_{m-1} \cup \{v_k\})$ where $v_k \in V \setminus (V_{m-1} \cup \{v_i\})$. For each pair (v_i, V_m) there are m corresponding pairs of the form $(v_i, V_m \setminus \{v_k\})$ where $v_k \in V_m$. This is a many-to-many mapping. Multiplying $\frac{n-m}{n-m}$ and $\frac{m}{m}$ to (11) and (12) respectively, we can construct a one-to-one mapping between the two summations in the numerators of (11) and (12) where in the second pair there is one more target node. Notice that $(n-m)C_n^1 C_{n-1}^{m-1} = mC_n^1 C_{n-1}^m$. By Lemma 2, we conclude $E[\rho(\widehat{V}_m)] \leq E[\rho(\widehat{V}_{m+1})]$. □

We provide two methods for efficiently estimating $E[\rho(\widehat{V}_m)]$. The first one is a sampling method: we randomly select $c^\dagger$ different sets of m nodes from V and estimate their ρ values using the sampling method described in section 5.1. The average of these ρ values is taken as an estimate of $E[\rho(\widehat{V}_m)]$. By following the same manner for Theorem 4's proof, we can derive the lower bound for $c^\dagger$ in order to obtain an $\epsilon^\dagger$-correct answer with respect to bounds aggregated from the bounds of underlying sample ρ's with probability $1-\delta^\dagger$. Recall that from Eq. (11) we know $E[\rho(\widehat{V}_m)]$ is equal to the expected DHT from any v_i to any set V_{m-1} which does not contain v_i. Thus, we can also directly sampling (v_i, V_{m-1}) pairs and take the average DHT among those pairs as an estimate of $E[\rho(\widehat{V}_m)]$. For a fixed graph, we can pre-compute $E[\rho(\widehat{V}_m)]$ for a number of different m values and employ interpolation to estimate $E[\rho(\widehat{V}_m)]$ for arbitrary m.

Alternatively, we can derive an approximation method for $E[\rho(\widehat{V}_m)]$ by a *geometric distribution*. This approximation method is empirical. A geometric distribution is a probability distribution of the number t of Bernoulli trials needed to get one success. When we randomly generate $\widehat{V}_m$, each node of G has probability $\frac{m}{n}$ to be chosen. In the following discussion, we assume each node of G is chosen independently with probability $\frac{m}{n}$. With this relaxation, $|\widehat{V}_m|$ becomes a binomial random variable with m as its expected value. Let us consider starting from a node $v_i \in \widehat{V}_m$ to hit the remaining nodes in $\widehat{V}_m$. Let $p = \frac{m-1}{n-1}$ be the probability of each node other than v_i being in $\widehat{V}_m$. The probability that we first hit (i.e. stop) a target node after one step is $\sum_j p_{ij} p = p$. The probability that we stop after two steps is $\sum_{j,k} p_{ij} p_{jk} (1-p) p = (1-p)p$. We do not consider cases where the surfer comes back to v_i in this approximation. This forms a geometric distribution where the probability that we "succeed" after t steps is $(1-p)^{t-1}p$. By the definition of DHT (i.e. Eq. (6)), $\tilde{h}(v_i, \widehat{V}_m \setminus \{v_i\})$ is actually the expectation of $e^{-(t-1)}$ under the geometric distribution described above:

$$\tilde{h}(v_i, \widehat{V}_m \setminus \{v_i\}) = \sum_{t=1}^{\infty} e^{-(t-1)}(1-p)^{t-1}p = \frac{p}{1-e^{-1}(1-p)}. \quad (13)$$

Since v_i is an arbitrary node in $\widehat{V}_m$, we have

$$\rho(\widehat{V}_m) = \frac{p}{1-e^{-1}(1-p)}. \quad (14)$$

Since we assume each node of G is chosen independently, the obtained $\rho(\widehat{V}_m)$ is an approximation of $E[\rho(\widehat{V}_m)]$. In case the graph contain 0-degree nodes, we just need to multiply Eq. (14) by the probability that a randomly selected node is not a 0-degree node. We empirically compare this heuristic approximation method with the sampling method on the DBLP co-author network. The results are shown in Figure 6. Regarding the sampling method, we sample 1500 (v_i, V_{m-1}) pairs for each m and use Algorithm 1 to estimate DHT. The error bars on the curve of the sampling method represent lower and upper bounds for the estimates of $E[\rho(\widehat{V}_m)]$. We can see that results obtained by sampling roughly fit the curve of the heuristic method. Therefore, we can either use sampling method and interpolation or the heuristic method to estimate $E[\rho(\widehat{V}_m)]$. In experiments we employ the heuristic method.

Regarding $Var[\rho(\widehat{V}_m)]$, we also propose a sampling method. Directly sampling $Var[\rho(\widehat{V}_m)]$ requires computing ρ for each sampled V_m and is time consuming since m can be large. If we assume $\tilde{h}(v_i, \widehat{V}_m \setminus \{v_i\})$'s in the numerator of the definition of $\rho(\widehat{V}_m)$ are independent, we can approximate $Var[\rho(\widehat{V}_m)]$ by $Var[\tilde{h}(v_i, \widehat{V}_m \setminus \{v_i\})]/m$. For a given m, we just sample (v_i, V_{m-1}) pairs and take the sample variance of the corresponding DHTs divided by m as an estimate of $Var[\rho(\widehat{V}_m)]$. Again, pre-computation and interpolation can be used here to estimate $Var[\rho(\widehat{V}_m)]$ for arbitrary m.

Table 1: Applying gScore on examples in Figure 1.

Example	p-value	Approximate Z score ($\tilde{\rho}$)
Fig. 1(A)	0.0191	2.0776
Fig. 1(B)	0.0309	1.9015
Fig. 1(C)	0.7418	-0.7117

5.3 Illustrative Examples

In order to show intuitively the capability of our measure, we report exact p-values and approximate Z scores (i.e. $\tilde{\rho}$) for the three toy examples of Figure 1 in Table 1. If we adopt significance level $\alpha = 0.05$, according to our measure, events in Figure 1(A) and (B) are deemed to be correlated with the network, indicating our measure can capture these two different distributions of correlation. While the event in Figure 1(C) does not have a correlation. The approximate Z scores ($\tilde{\rho}$) reflect relative correlation degree and can be used to rank structural correlations. Since distributions of $\rho(\widehat{V}_m)$ are not exactly normal, we cannot simply convert these scores to p-values to determine the existence of correlations. In practice gScore is suitable for ranking correlations since it is hard to obtain p-values.

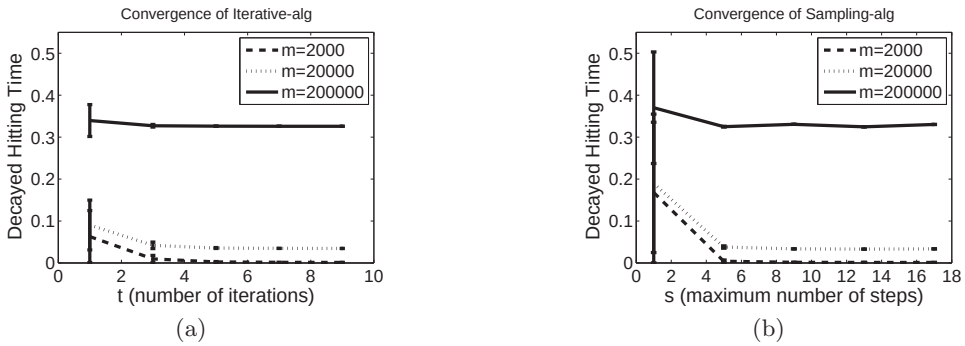

Figure 7: Exploring the convergence of (a) Iterative-alg with respect to the number of iterations, and (b) Sampling-alg with respect to the maximum number of steps a random walk performs.

6. EMPIRICAL STUDIES

This section present experimental results on three real world datasets: DBLP, TaoBao and Twitter. We report the top-k interesting events discovered by our approach and also perform some manual verification. We provide an interesting study of correlation changes of product sales and research topics over time. We also examine our measure on synthetic events and under structure alteration. Finally, we analyze the scalability of gScore with a Twitter network. All experiments are run on a PC with Intel Core i7 CPU and 12GB memory. The source code of gScore can be downloaded at "http://www.cs.ucsb.edu/∼xyan/software/gScore.html".

6.1 Datasets

We empirically evaluate our correlation measure and gScore on three real world datasets: DBLP, TaoBao and Twitter.

DBLP The DBLP snapshot was downloaded on Oct. 5th, 2010 (http://www.informatik.uni-trier.de/∼ley/db). Its paper records were parsed to obtain the co-author social graph. Keywords in paper titles are treated as events associated with nodes (authors) on the graph. The first time an author used a keyword was also recorded. It contains 815,940 nodes, 2,857,960 edges and 171,614 events.

TaoBao The TaoBao dataset was derived from China's most famous customer-to-customer shopping Website named TaoBao (http://www.taobao.com). By the end of 2009, TaoBao has about 170 million users and 1 billion products. We extracted users from three cities (Beijing, Shanghai and Hangzhou), their product purchase history, and constructed the friend social graph among them. It consists of 794,001 nodes, 1,370,284 edges. We selected 100 typical products from TaoBao to show the effectiveness of our measure.

Twitter The Twitter dataset has about 40 million nodes and 1.4 billion edges (http://twitter.com). We do not have events for this dataset. It is mainly used to test the scalability of gScore.

6.2 Performance of DHT Approximation

We investigate the convergence and running time of the two DHT approximation algorithms proposed in Section 4. DBLP is used as the test bed. We use *Iterative-alg* and *Sampling-alg* to denote the iterative algorithm and sampling algorithm, respectively. Iterative-alg has one parameter (number of iterations t) and Sampling-alg has two parameters (maximum number of steps s and number of random walks c). For Iterative-alg, we investigate its converging speed with respect to t. For Sampling-alg, we find when $c > 600$, increasing c hardly improves the obtained bounds. Thus, we set $c = 600$ and investigate the converging speed of Sampling-alg with respect to s. The results are shown in Figure 7 with various m values (the number of nodes that have the same event). For each m value, we randomly select a node v and a set B of $m-1$ nodes and apply the two algorithms to estimate $\tilde{h}(v, B)$. This process is repeated 50 times and the averaged results are reported. As shown in Figure 7, both algorithms converge quickly after about 5 iterations. Note that Iterative-alg gives lower and upper bounds for $\tilde{h}$, while Sampling-alg gives bounds for an estimate of $\tilde{h}$, i.e. $\overline{\tilde{h}}$. Comparing Figure 7(a) and 7(b), one can find that the two algorithms converge to roughly the same values. It means empirically Sampling-alg provides a good estimation of $\tilde{h}$.

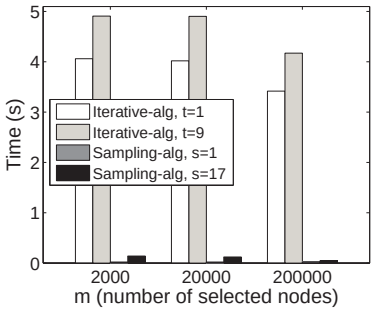

Figure 8: Comparison of Iterative-alg and Sampling-alg with respect to the time used to estimate one DHT.

The running time of Iterative-alg and Sampling-alg for estimating one DHT under different m values is shown in Figure 8. For Iterative-alg, we report the running time for $t = 1$ and $t = 9$ and for Sampling-alg, $s = 1$ and $s = 17$. It shows that Sampling-alg is much faster than Iterative-alg. Note that regarding Iterative-alg, the time cost of "s=9" is not 9 times of that of "s=1". This is because not only matrix-vector multiplication but also the construction of $\mathbf{P}_B$ account for time cost. In fact, Iterative-alg runs even faster when m increases: less rows of $\mathbf{P}$ are needed to con-

Table 2: Correlation scores for top five products in category "Laptops and tablets" in TaoBao.

#	Product	Bounds for $\tilde{\rho}$	ρ ($\times 10^{-2}$)	$\lvert V_q \rvert$
1	ThinkPad T400	[554.43, 554.47]	[6.2396, 6.2400]	47
2	Apple iPad	[227.56, 227.57]	[6.7979, 6.7984]	698
3	ThinkPad X200	[91.39, 91.42]	[1.0799, 1.0802]	60
4	Toshiba L600	[20.36, 20.41]	[0.2009, 0.2014]	31
5	ThinkPad T410	[−1.13, −1.09]	[0.0004, 0.0009]	72

Table 3: Correlation scores for top five products in category "Mobile and handheld devices" in TaoBao.

#	Product	Bounds for $\tilde{\rho}$	$\lvert V_q \rvert$
1	iPod Touch 3	[92.06, 92.09]	484
2	Nokia 6300	[90.97, 90.99]	188
3	iPhone 4	[69.07, 69.09]	520
4	Nokia N82	[53.20, 53.24]	84
5	HTC G3	[36.48, 36.49]	732

struct the desired matrix. Since Sampling-alg is much faster than Iterative-alg and also provides reasonable estimates for DHTs, for the following experiments we employ Sampling-alg to estimate DHT. gScore is also referred to Sampling-alg. We set $s = 12$ and $c = 600$.

6.3 Structural Correlation Estimation

We employ our adjusted correlation measure $\tilde{\rho}$ (Eq. (10)) to rank products and keywords in TaoBao and DBLP, respectively. Using Eq. (10), we obtain an estimate (lower and upper bounds) of $\tilde{\rho}$ for each product and keyword. A ranked list of events can be generated according to these bounds. If the bounds of two events overlap, we increase sample numbers and the maximum steps to break a tie.

Before presenting the results, we would like to emphasize that our correlation findings are just for the specific social networks involved in this study. For example, if a product has no correlation with the TaoBao social network, it does not mean the product sales is not influenced by other social channels. For the sake of clarity, products in TaoBao are divided into three categories: *Laptops and tablets*, *Mobile and handheld devices* and *Other*. Due to space limitation, we show the results for top five correlated products in each category.

Table 2 and Table 3 show the ranked lists for top five products in the former two categories, respectively. We also show ρ values in Table 2. ThinkPad and Apple products usually have high correlation with the underlying network, indicating there are fan communities for these brands. An interesting exception is ThinkPad T410 (Table 2), which is a new version of Thinkpad T400. In comparison with T400, its correlation score is very close to that of random cases. The reason may be people in the fan community already bought T400 and they would not further buy a new version for T400 since they are quite similar and not cheap.

Table 4: Correlation scores for top five products in category "Other" in TaoBao.

#	Product	Bounds for $\tilde{\rho}$	$\lvert V_q \rvert$
1	Mamy Poko baby diapers	[238.50, 238.51]	4892
2	Beingmate Infant milk powder	[227.71, 227.72]	163
3	EVE game cards	[198.56, 198.58]	374
4	Mabinogi game cards	[189.56, 189.58]	446
5	Gerber cookies	[149.51, 149.52]	1491

The ranked list for top five products from category "Other" is shown in Table 4. Here "EVE" and "Mabinogi" are two online games and players in China must buy *game cards* to obtain gaming time. We find products for infants, like diaper and powder tend to be correlated with the network. This indicates people tend to follow friends' recommendation when choosing this kind of products. Game card is another kind of highly correlated products. Intuitively playing with friends is an important attractive feature of online games.

Finally, we show the correlation scores for the five most uncorrelated products from category "Other" in Table 5. These products' scores are very close to those of random cases (some scores deviate a little from random cases due to estimation errors in variance). This indicates for clothing and accessories people usually follow their own preference.

Table 5: Correlation scores for the five most uncorrelated products in category "Other" in TaoBao.

#	Product	Bounds for $\tilde{\rho}$	$\lvert V_q \rvert$
1	Tiffany rings	[2.71,2.72]	1092
2	Jack&Jones suits	[−0.48, −0.46]	311
3	Ray-Ban sunglasses	[−0.78, −0.77]	4958
4	Swarovski anklets	[−0.88, −0.84]	72
5	Jack&Jones shirts	[−3.28, −3.27]	1606

Table 6: Manual investigation of consumers for three products. "k-hop friend relationship" means the distance between two consumers is k.

Products	# k-hop friend relationships			$\lvert V_q \rvert$
	$k=1$	$k=2$	$k=3$	
ThinkPad T400	4	7	10	47
ThinkPad T410	0	1	19	72
Swarovski anklets	0	1	32	72

To further verify the effectiveness of gScore, we manually examine the consumers for three products: ThinkPad T400, ThinkPad T410 and Swarovski anklets. The numbers of k-hop friend relationships among the corresponding consumers are reported in Table 6 for $k = 1, 2, 3$. A k-hop friend relationship means the distance between two consumers is k. It is easy to see the consumers of ThinkPad T400 are closer than those for the other two products. Our measure correctly reflects the situation described by Table 6. We find for ThinkPad T400, there is a small community which contains 4 people and 4 edges. By investigating their profiles in TaoBao, we find they are geographically near each other and all like buying electronic products. It is this small community that makes the correlation score of ThinkPad T400 high.

For DBLP, due to space limitation, we only show the results of 12 representative keywords (Table 7), together with the numbers of authors who published papers with these keywords. It is observed that keywords representing general topics (e.g. Computation) are less correlated with the underlying co-author network than those representing more focused topics (e.g. Hadoop, Microarray). General keywords often randomly occur in the research network, while dedi-

Table 7: Correlation ranking for keywords in DBLP.

#	Keyword	Bounds for $\tilde{\rho}$	$\lvert V_q \rvert$
1	Hadoop	[1225.22, 1225.46]	57
2	Microarray	[958.64, 958.67]	4738
3	OLTP	[912.52, 912.68]	105
4	AJAX	[857.61, 857.74]	179
5	Virus	[825.14, 825.21]	905
6	E-Learning	[811.85, 811.91]	3552
7	Database	[775.80, 775.83]	19522
8	Mining	[760.33, 760.36]	15371
9	Relational	[697.17, 697.22]	6225
10	Retrieval	[647.60, 647.64]	13996
11	Indexing	[624.94, 625.00]	5069
12	Computation	[586.58, 586.63]	11187

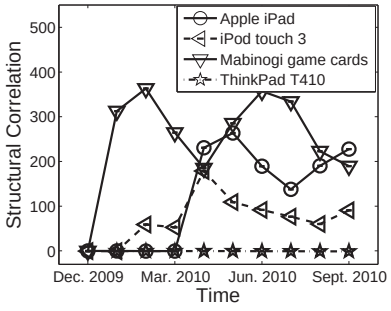

Figure 9: Correlation evolution in TaoBao.

cated keywords are usually topics pursued by a relatively small community of researchers. Scores on DBLP are generally higher than those on TaoBao. This is because if a keyword appears in the title of a paper, it will appear on all authors of that paper between whom there is a fully connected subgraph. Notice that the ranked list obtained by our measure is not just a ranking of keyword popularity. For example, comparing "Database" and "Indexing", we can see "Database" is not only more popular than "Indexing" (19522 vs. 5069), but also more correlated with the network structure. This is intuitive because "Indexing" is more general than "Database" and researchers from different communities would pursue it.

6.4 Evolution of Structural Correlation

When incorporating the time factor, our structural correlation measure can be used to investigate the evolution of an event's correlation with the underlying structure, which will shed light on users' behaviors. We compute correlation scores for an event at different time points. At each time point, all nodes on which an event q had occurred before that time point are used to form V_q. Intuitively, structural correlation of an event q would increase if newly engaging nodes for q are close neighbors of those where q had already occurred, and decrease if newly engaging nodes are random nodes (that is, these nodes are not influenced by network structure). We call these two types of newly engaging nodes *link-following newly engaging nodes* and *random newly engaging nodes*, respectively.

The TaoBao dataset contains purchase transactions between Nov. 2009 and Oct. 2010. Hence, we take month as the time unit. Figure 9 presents the evolution curves for four

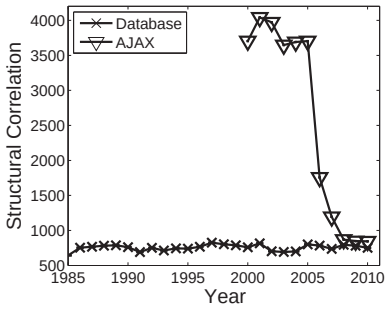

Figure 10: Correlation evolution in DBLP.

products. It was observed that for almost all the products which have correlations close to random cases in our previous experiments, their curves remain around 0 (e.g., "Thinkpad T410" in Figure 9). We suspect that it would be difficult to promote those products (e.g. viral marketing) using the underlying social network. For the other three products shown in Figure 9, at the beginning, purchases just occur randomly (i.e. the curve is close to 0); after a while one or more peaks will appear, indicating many link-following newly engaging nodes appear. The sales of these products is likely influenced by social links, thus suitable for promotion. The correlation may drop since people may found the product from other channels. An interesting observation is that the first peak of Mabinogi game cards appears between January and March of 2010, which collides with Spring Festival, the most important festival in China. The peak implies players successfully invited their friends to play this online game in holidays.

For the DBLP dataset, we use year as the time unit and investigate the varying trends of keywords' structural correlations from 1985 to 2010. Figure 10 shows evolution patterns for two keywords, "Database" and "AJAX". For relatively newer and more focused topics (e.g. "AJAX"), the correlation curve decreases abruptly in the evolution process. This represents the topic dispersion phenomenon: at the beginning, several researchers proposed the topic; then more and more different researchers (random newly engaging nodes) started to pursue this topic, making the keyword less correlated with the social network. For long standing and more general research topics like "Database", the correlation remains relatively stable: every year professors engaged in the topic would hire new students to pursue that topic and some random researchers would also publish papers on that topic (accounting for link-following and random newly engaging nodes, respectively).

6.5 Results on Synthetic Data

We apply our measure on synthetic events and graphs with changing neighborhood properties. DBLP graph is used. We create synthetic events using the cascade model for influence spread [17]: at first a random set of 100 nodes is chosen as the initial V_q; then in each iteration nodes joining V_q in the last iteration can activate each currently inactive neighbor with probability p_{ac}; we stop when $|V_q| > 10000$. p_{ac} can be regarded as representing the level of participation in an event. Intuitively, higher p_{ac} would lead to higher correlation. The results are shown in Figure 11. "Random" means we expand the initial 100 random nodes with a number of randomly selected nodes from the remaining nodes in order

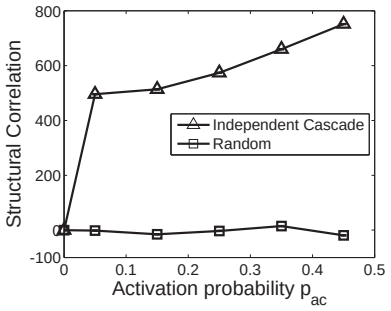

Figure 11: Applying gScore on synthetic events.

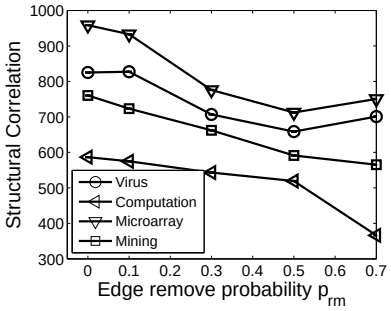

Figure 12: Variation of structural correlation when randomly removing edges.

to match the corresponding event sizes of cascade model. We can see as p_{ac} increases, the curve of cascade model goes up, while that of "Random" remains around 0.

For the graph alteration experiment, we randomly remove edges from the DBLP graph (each edge is removed with probability p_{rm}). The structural correlations of four keywords under different p_{rm} are reported in Figure 12. It is observed that randomly removing edges from the network affects the correlation scores of events. In general, the correlation scores decrease with increasing p_{rm} since edge removal would damage the correlation structure of an event and make the event less correlated. Occasionally, the curves go up. This is because edge removal also affects $E[\rho(\widehat{V_q})]$ and $Var[\rho(\widehat{V_q})]$.

6.6 Scalability of Sampling-alg

Finally, we investigate the scalability of Sampling-alg when the graph size n increases. The Twitter graph is used to perform this experiment. We extract subgraphs with different sizes (i.e. n) and for each n, different values of m are tested. Results are averaged over 50 sampled DHTs. Figure 13 shows that Sampling-alg is scalable and only need 0.17s to estimate one DHT on a graph with 10 million nodes. Although the time cost of Sampling-alg is linear in n, it only involves creating an index array of size n in memory. Regarding ρ, the estimation time is only 8.5s on a graph with 10 million nodes if we set the number of samples $c' = 50$. Note that this can also be regarded as the time used for computing one adjusted correlation $\tilde{\rho}$ since $E(\rho)$ and $Var(\rho)$ can be obtained from pre-computed results. Intuitively, when n is fixed and m increases, the running time should decrease since it is easier to hit a target node (most random walks do not need to reach the maximum steps, s). This is the reason

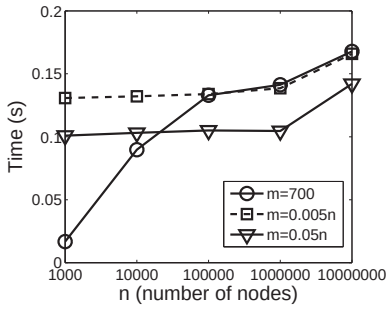

Figure 13: Running times of Sampling-alg for estimating one DHT when varying the graph size.

that the curve of $m = 0.05n$ is below that of $m = 0.005n$. Since we only store the adjacency list, the memory cost is linear in the number of edges in the graph. We do not show the curve here due to space limitation.

7. RELATED WORK

Broadly speaking, our work is related to graph clustering [28, 31, 9, 13] and dense subgraph mining [10, 25, 32]. A graph cluster shall contain many internal edges but relatively few edges to other clusters. Dense subgraph mining is to find subgraphs with high intra-connectivity and low inter-connectivity. However, these problems only aim at finding closely related nodes on graphs and they do not consider the attributes (events) associated with nodes.

Recently, researchers have investigated node attributes in graph mining. For example, Ester *et al.* found attribute data can be used to improve clustering performance [8]. Zhou *et al.* constructed clustering algorithms based on structural and attribute similarities [33]. Silva *et al.* [29] proposed a novel structural correlation pattern mining problem which aims to find pairs (S, V) in which S is a frequent attribute set and V is a dense subgraph where each node contains all the attributes in S. The key difference between this problem and ours is that we study the correlation between an attribute and the entire network structure globally. In [22], the authors examined how to find dense subgraphs where nodes had homogeneous attributes. Again, they also focused on local patterns but not global structural correlations studied in this work.

Our work is also related to research in social influence. In social networks, behaviors of two people tend to be related if they are friends. Anagnostopoulos *et al.* [2] studied the problem of distinguishing social influence from other sources of correlation using time series of people behaviors. La Fond and Neville [19] presented a randomization technique for distinguishing social influence and homophily for temporal network data. Our work is different from theirs in two aspects. First, these studies assume the existence of correlations, while we try to determine if there is a correlation. Second, they are concerned with direct friendship, while our structural correlation is defined in a more general graph proximity notion.

There are many graph proximity measures proposed in the literature. Here we name a few. *Common neighbors* and *Jaccard's coefficient* are two measures based on node neighborhood [23]. Common neighbors computes the number of common neighbors of two nodes. Jaccard's coefficient

is defined as the number of common neighbors divided by the number of all distinct neighbors of two nodes. *Katz* [15] defined a measure which sums over all paths between two nodes, exponentially damped by their length to make short paths more important. *Hitting time* [20] and *personalized PageRank* [24] are random walk based graph proximity measures. Our structural correlation framework can adopt any graph proximity measure. We use hitting time in this work since it is a more holistic measure.

The randomization and sampling techniques used in this work have been studied extensively. Gionis *et al.* used swap randomization to assess data mining results [11], while [18] provided a rigorous bound for identifying statistically significant frequent itemsets. We successfully extended related techniques to the graph domain and showed that these techniques with appropriate modifications are scalable in large-scale graphs.

8. CONCLUSIONS

In this paper, we studied the problem of measuring how strongly an event that took place in a graph is correlated to the graph structure. A novel measure, called structural correlation, was introduced to assess this correlation. It can also be used to derive statistical significance to test if an event is randomly distributed over a graph or not. We proposed using hitting time to instantiate our framework and derived a set of sampling and approximation algorithms so that the correlation score can be estimated very quickly in large-scale graphs. By comparing the score with the situation where the event is randomly distributed in the same network, our method is able to discover the events of nodes that are highly correlated with the graph structure. Our method is scalable and successfully applied to the co-author DBLP network and social networks extracted from TaoBao.com, the largest online shopping network in China, with many exciting discoveries.

9. ACKNOWLEDGMENTS

This research was sponsored in part by the U.S. National Science Foundation under grant IIS-0905084 and by the Army Research Laboratory under cooperative agreement W911NF-09-2-0053 (NS-CTA). X. Yan was supported in part by the Institute for Collaborative Biotechnologies through Grant, No. DFR3A-8-447850-23002 from the U.S. Army Research Office. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

10. REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216, 1993.

[2] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *SIGKDD*, pages 7–15, 2008.

[3] H. Bao and E. Y. Chang. Adheat: an influence-based diffusion model for propagating hints to match ads. In *WWW*, pages 71–80, 2010.

[4] M. Brand. A random walks perspective on maximizing satisfaction and profit. In *SDM*, 2005.

[5] J. J. Brown and P. H. Reingen. Social ties and word-of-mouth referral behavior. *J. of Consumer Research*, 14(3):350–362, 1987.

[6] P. Chebotarev and E. V. Shamis. On proximity measures for graph vertices. *Automation and remote control*, 59(10):1443–1459, 1998.

[7] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *SIGKDD*, 2010.

[8] M. Ester, R. Ge, B. J. Gao, Z. Hu, and B. Ben-Moshe. Joint cluster analysis of attribute data and relationship data: the connected k-center problem. In *SDM*, pages 25–46, 2006.

[9] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *SIGKDD*, pages 150–160. ACM, 2000.

[10] D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *VLDB*, pages 721–732, 2005.

[11] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. In *SIGKDD*, pages 167–176, 2006.

[12] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. of the American Statistical Association*, 58(301):13–30, 1963.

[13] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Natural communities in large linked networks. In *SIGKDD*, pages 541–546, 2003.

[14] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, pages 271–279. ACM, 2003.

[15] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[16] H. Kautz, B. Selman, and M. Shah. Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

[17] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, pages 137–146. ACM, 2003.

[18] A. Kirsch, M. Mitzenmacher, A. Pietracaprina, G. Pucci, E. Upfal, and F. Vandin. An efficient rigorous approach for identifying statistically significant frequent itemsets. In *PODS*, 2009.

[19] T. La Fond and J. Neville. Randomization tests for distinguishing social influence and homophily effects. In *WWW*, pages 601–610, 2010.

[20] L. Lovász. Random walks on graphs: A survey. *Bolyai Soc. Math. Studies*, 2(2):1–46, 1993.

[21] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM*, 2008.

[22] F. Moser, R. Colak, A. Rafiey, and M. Ester. Mining cohesive patterns from graphs with feature vectors. In *SDM*, 2009.

[23] D. L. Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559, 2003.

[24] L. Page, S. Brin, R. Motwani, and T. Winograd. The Pagerank citation algorithm: bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[25] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *SIGKDD*, pages 228–238, 2005.

[26] P. Sarkar and A. Moore. A tractable approach to finding closest truncated-commute-time neighbors in large graphs. In *UAI*, 2007.

[27] P. Sarkar, A. W. Moore, and A. Prakash. Fast incremental proximity search in large graphs. In *ICML*, pages 896–903, 2008.

[28] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[29] A. Silva, W. Meira Jr, and M. J. Zaki. Structural correlation pattern mining for large graphs. In *Proc. of the 8th Workshop on Mining and Learning with Graphs*, pages 119–126, 2010.

[30] R. Srikant and R. Agrawal. Mining generalized association rules. *Future Generation Computer Systems*, 13(2-3):161–180, 1995.

[31] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[32] Z. Zeng, J. Wang, L. Zhou, and G. Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *SIGKDD*, pages 797–802, 2006.

[33] Y. Zhou, H. Cheng, and J. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.