LinkProbe: Probabilistic Inference on Large-Scale Social Networks

Haiquan Chen[1] Wei-Shinn Ku[2] Haixun Wang[3] Liang Tang[2] Min-Te Sun[4]
[1]Valdosta State University, GA, USA
[2]Auburn University, AL, USA
[3]Microsoft Research Asia, Beijing, China
[4]National Central University, Taiwan
hachen@valdosta.edu,{weishinn,lzt}@auburn.edu,haixunw@microsoft.com,msun@csie.ncu.edu.tw

***Abstract*— As one of the most important Semantic Web applications, social network analysis has attracted more and more interest from researchers due to the rapidly increasing availability of massive social network data. A desired solution for social network analysis should address the following issues. First, in many real world applications, inference rules are partially correct. An ideal solution should be able to handle partially correct rules. Second, applications in practice often involve large amounts of data. The inference mechanism should scale up towards large-scale data. Third, inference methods should take into account probabilistic evidence data because these are domains abounding with uncertainty. Various solutions for social network analysis have existed for quite a few years; however, none of them support all the aforementioned features. In this paper, we design and implement LinkProbe, a prototype to quantitatively predict the existence of links among nodes in large-scale social networks, which are empowered by Markov Logic Networks (MLNs). MLN has been proved to be an effective inference model which can handle complex dependencies and partially correct rules. More importantly, although MLN has shown acceptable performance in prior works, it is also reported as impractical in handling large-scale data due to its highly demanding nature in terms of inference time and memory consumption. In order to overcome these limitations, LinkProbe retrieves the k-backbone graphs and conducts the MLN inference on both the most globally influencing nodes and most locally related nodes. Our extensive experiments show that LinkProbe manages to provide a tunable balance between MLN inference accuracy and inference efficiency.**

I. Introduction

As one of the most important Sematic Web applications, social network analysis [1], [2], [3], [4], [5] has attracted more and more interest from researchers due to the rapidly increasing availability of massive social network data for various Web 2.0 applications such as Facebook, YouTube, Flickr, and Wikipedia. In these applications, users are not only data consumers but also data producers. Friend-Of-A-Friend (FOAF) [6], [7] is an RDF/XML ontology especially designed to describe basic attributes of people and relationships among them, including name, mailbox, homepage URL, friends, interests, affiliations, etc. The friend relationships described in a FOAF data set can be depicted as a social graph, where each person is represented by a node and each friendship is denoted as an edge between two nodes as demonstrated in Figure 1. In this paper, we are interested in answering the following query: given two arbitrary nodes, x and y, what is the possible probability that a specific relationship (link) exists between x and y given G as evidence? In social network analysis, evaluating the existence of links is crucial for predicting relationships among people, inferring profiles, clustering people for community discovery, criminal network detection, etc.

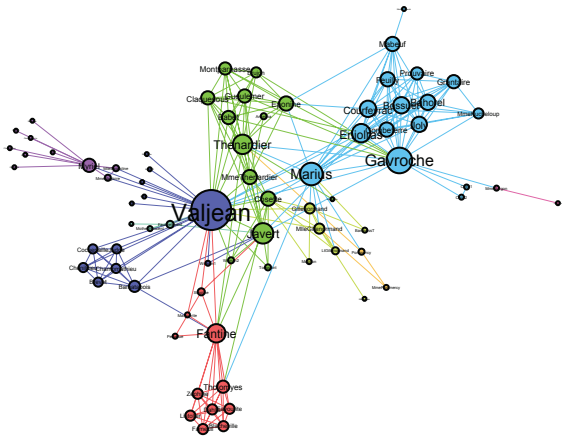
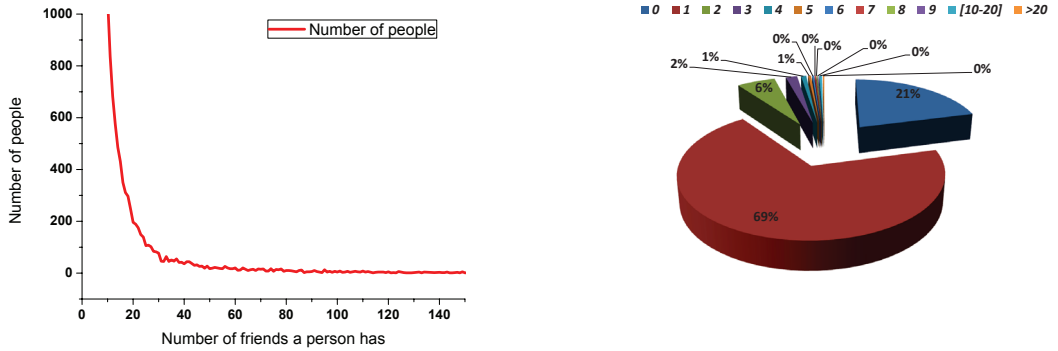


Fig. 1. An example social graph.

A. Challenges

However, to answer the aforementioned query is difficult.

- First, the desired solution should support partially correct inference rules. This is because most inference rules existing in social network analysis do not always hold in reality. For example, the transitive propriety in a friendship can be captured by the following rule: $\forall x, y, z \in \mathrm{P} : knows(x, y) \wedge knows(y, z) \longrightarrow knows(x, z)$, where P means the set of people. However, this rule is not always true, i.e., it can be violated in some cases. Although many inference mechanisms have been proposed in the literature, most of them fail to support inference on partially correct rules, i.e., they require that underlying rules must be satisfied all the time and cannot be violated during the entire inference procedure.
- Second, the ideal solution should scale well towards very large data volume, i.e., the inference cost should be affordable and tractable when coping with massive



(a) Power law curve. (b) Distribution of the number of friends.

Fig. 2. Statistics on the Billion Triple Challenge (BTC) 2009 data set.

data sets. This is because most social networks in practice often consist of a tremendous number of nodes and edges, featuring a huge amount of semi-structured data.

- Third, the desirable solution should be able to handle uncertain social data as evidence, where links between nodes are described probabilistically. In most real applications, social data (i.e., user profile and friend information) are often uploaded voluntarily by users themselves or obtained from various prediction techniques. As a consequence, impreciseness and uncertainty may arise in social data. Therefore, the desired method should support inference on such uncertain social data as well.

B. Observations

1) MLNs Support Partially Correct Inference Rules: As a unified inference method to combine probabilistic graph models and first-order logics, Markov Logic Networks (MLNs) [8], [9], [10] have shown their theoretical potentials in reasoning over partially correct rules [11] and modeling multi-variate structured dependency [12]. The inference on MLN is based on the inference over the resulting grounded Markov network (Markov random field) [13].

2) Social Graphs Tend to Be Sparse Globally and Dense Locally: Although MLNs are commonly employed to deal with partially correct inference rules, the major obstacle in applying MLNs in practice is that their grounded Markov network would be prohibitively huge. In other words, Markov logic networks become extremely inefficient in coping with large-scale data due to their highly demanding nature in terms of inference time and memory consumption [14]. This is because Markov logic networks require a complete grounding of rules in order to count the number of grounded rules (i.e., formula groundings) that are true given a particular possible world. Such counting becomes impractical when dealing with large-scale data because the number of formula groundings grows exponentially as the number of individuals in the investigated domain increases. However, unfortunately, social networks in reality usually contain a massive number of social relations, leading to the infeasibility of applying MLN directly on social networks in practice.

To tackle this issue, we observed that most social networks exhibit typical characteristics of scale-free networks. First, vertex degrees follow power law distributions. Second, vertices tend to cluster together. Third, the average length of all shortest paths between any two vertices is logarithmically small. We calculated statistics over a 1 % random sample on the Billion Triple Challenge (BTC) 2009 data set[1] with up to more than 13 million people and 35 million friend relations. Figure 2(a) illustrates the relationship between the number of friends (i.e., it can be interpreted as the degree of a vertex) and the number of people who have the number of friends in the BTC data set. The corresponding pie chart is demonstrated in Figure 2(b). Figures 2(a) and 2(b) reveal a fact that an overwhelming majority of people have a very limited number of friends while only a few number of people have a considerable number of friends (i.e., they are highly connected and can be considered as hubs in a given social graph). As depicted in Figure 2(b), 21% of the people do not have any friends, 69% of the people have exactly one friend, and 6% of the people have two friends, according to our sampled BTC data set.

In this paper, we attempt to take advantage of such characteristics to devise an MLN-driven inference engine with an affordable cost for searching massive social networks. We employ a k-backbone graph to discover the global topology of a given social network and conduct inference on both the most globally influencing nodes and the most locally related nodes.

C. Our Approach

Our paper presents LinkProbe, a prototype to estimate link existence in uncertain social networks based on probabilistic reasoning. LinkProbe is empowered by MLN inference but manages to provide a tunable balance between MLN inference accuracy and inference efficiency. First, in order to maintain the infrastructure of social graphs, we sort all the nodes by their degrees and construct the corresponding k-backbone graph. Then, we issue two independent runs of the random walk Metropolis sampling to explore local social graphs. Subsequently, LinkProbe applies MLN inference on the union of all the nodes in the k-backbone graph and all nodes discovered

[1] http://vmlion25.deri.ie/

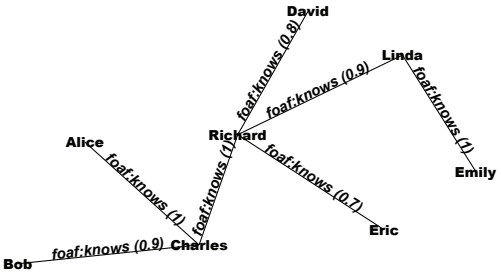


Fig. 3. An example of a probabilistic social graph.

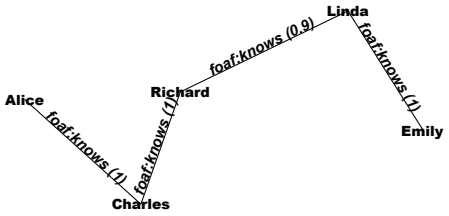


Fig. 4. 1-backbone.

locally. In addition, in order to support uncertain/probabilistic evidence data, LinkProbe adopts MC-SAT$^+$, a probabilistic extension of the well-known MLN inference method MC-SAT.

D. Our Contributions

We summarize our contributions as follows:

- We define probabilistic social graphs by injecting uncertainty into social graphs and formalizing the link queries on them. Afterwards, LinkProbe, a prototype for predicting probabilistic links, is proposed to answer such queries.
- We propose LinkProbe, which is driven by Markov logic networks and is often utilized in handling partially correct inference rules which appear commonly in social networks.
- In order to handle large data volumes, LinkProbe takes full advantage of the fact that most social graphs tend to be sparse globally and dense locally. LinkProbe conducts the MLN inference on both the most globally influencing nodes and the most locally related nodes. Compared to the "entire graph" based naive implementation, LinkProbe reduces the time and space costs by several orders of magnitude and, in the meanwhile, maintains the inference accuracy in an acceptable level. With a much higher scalability than the naive MLN implementation, LinkProbe is applicable to large social networks for probabilistic reasoning.

E. Paper Organization

The rest of this paper is organized as follows. The research problem is formally defined in Section II. In Section III we focus on the methods for constructing the k-backbone graphs. We explain how to explore d-local graphs and random walk in Section IV. The basics of Markov logic networks are introduced in Section V and the inference method in LinkProbe is discussed in Section VI. We illustrate the error analysis in Section VII. The experimental validation of our system is presented in Section VIII. Section IX surveys related works. Section X concludes the paper with future work.

II. Problem Formulation

A. Probabilistic Social Graphs

Definition 2.1 A *Probabilistic Social Graph* (PSG) is a social graph where each edge is associated with a probabilistic value to reflect the likelihood that a specified relation (e.g., friendship) exists among the two linked people.

Probabilistic social graphs can be stored in a database, denoted as $\mathcal{DB}$. $\mathcal{DB}$ may have the following schema: $< from, to, prob >$. Specifically, each pair of $from$ and to contains the two IDs of the vertices (i.e., involved people) associated to an edge (i.e., a friendship between two people). In addition, $prob$ denotes the weight of an edge, reflecting the probability of a friendship. Throughout our paper, without loss of generality, we consider such relation as a mutual friendship between people. For example, a record $< 11, 32, 0.8 >$ means that the person with ID 11 knows another person with ID 32 with a probability of 0.8.

B. Link Prediction

Definition 2.2 Given a probabilistic social graph G, the purpose of *link prediction* is to predict the probability that a specific link exists between two nodes in G. Take Figure 3 as an example. $Query(Bob, Emily, knows)$ may be launched to investigate how well Bob knows Emily, i.e., from the probabilistic view, to estimate the likelihood that a potential friendship edge exists from Bob to Emily.

III. The k-backbone Graphs

Definition 3.1 The *weighted degree* (WD) of a vertex u in a probabilistic social graph G, denoted as $WD^G(u)$, is defined to be the sum of the weights of all the edges incident to u. In other words, $WD^G(u)$ can be calculated as $WD^G(u) = \sum_{e \in \mathrm{E}(u)} W(e)$, where $\mathrm{E}(u)$ denotes the set of edges incident to u and $W(e)$ means the associated weight of edge e in G.

Definition 3.2 A *k-backbone graph* of a probabilistic social graph G is a subgraph of G which can be acquired by deleting from G all the vertices with the weighted degree less than k and all the associated edges.

Figures 4 and 5 demonstrate examples of 1-backbone and 2-backbone graphs. The detailed steps of k-backbone graph construction are formalized in Algorithm 1. For each vertex

Fig. 5. 2-backbone.

v_i, we add v_i to the k-backbone graph if $WD^G(v_i)$ is no less than k.

Algorithm 1 Retrieval of vertices in k-backbone graphs

Input: A probabilistic social graph G stored in database $\mathcal{DB}$
Output: All the vertices in the k-backbone graph, output as B

```
B = ∅
for each < from, to, prob > in DB do
    DB = DB ⋃ < to, from, prob >
    {We assume that relations are undirected here}
end for
for each vertex v_i in DB do
    Compute the weighted degree of v_i, WD^G(v_i)
    if WD^G(v_i) ≥ k then
        B = B ⋃ v_i
    end if
end for
```

IV. Random Walk

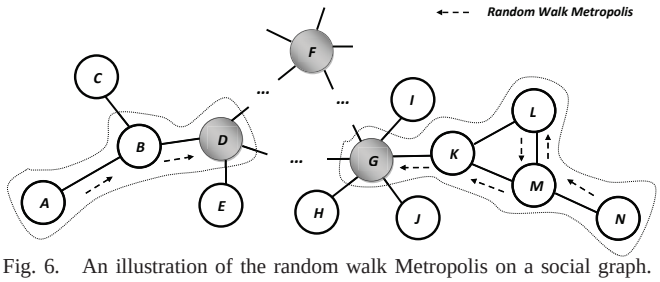


Fig. 6. An illustration of the random walk Metropolis on a social graph.

In the previous section, we illustrate how to explore the skeleton of a given social graph by discovering its k-backbones. In this section, we elaborate on how to focus our exploration on a relatively small part of the graph by retrieving d-local graphs and then employing a MCMC-based sampler, the Random Walk Metropolis (RWM), in order to connect d-local graphs with the derived k-backbone graph.

A. d-local Graphs

Definition A *d-local graph* of a node X in a probabilistic social graph G is a subgraph of G which can be acquired by deleting from G all the neighboring nodes connected with an edge with a probability less than d and all the associated edges. In a social graph, a vertex in the *d-local graph* of a node X means that a person knows X with a probability of no less than d.

B. Markov Chain Monte Carlo

A Markov chain is a stochastic process which consists of possible states of random variables. It can be denoted as a sequence states of X_1, X_2, X_3, ..., X_n, which satisfy

$$p(X_{n+1} = x | X_n = x_n, X_{n-1} = x_{n-1}, ..., X_1 = x_1) = p(X_{n+1} = x | X_n = x_n)$$

where $p(x|y)$ is the transition probability from state y to state x. Markov Chain Monte Carlo (MCMC) is a technique to generate samples from the state space by simulating a Markov chain [15], [16]. The formed Markov chain is constructed in such a way that the chain spends more time in the regions with higher importance, i.e., the stationary distribution of the Markov chain is the same as the target distribution. That is, the Markov chain can converge to the target distribution (the posterior) as its equilibrium distribution. From the perspective of Monte Carlo sampling, as the number of samples are sufficiently large, all the samples can become the fair samples from the posterior. Consequently, we are able to approximate the sophisticated target posterior based on deliberately constructing a Markov Chain of all the Monte Carlo samples.

C. Connecting Local Graphs with the Backbone Graphs

The Random Walk Metropolis (RWM) sampler is one of the most used MCMC-based samplers, which generates a sequence of random walks utilizing a proposal density and decides on whether to reject the proposed moves by employing the rejection sampling.

The random walk Metropolis algorithm simulates a Markov chain in which each state X_{t+1} only depends on the immediately previous state X_t. A new sample X' is proposed depending on the current state X_t. X' is accepted as the next state X_{t+1} with the probability of an acceptance rate α, which can be formalized as Equation 1. $P(X)$ is the probability of state X.

$$\alpha = P(X_{t+1} = X' | X_t) = \min\left\{1, \frac{P(X')}{P(X_t)}\right\} \quad (1)$$

$$P(u \longrightarrow w) = \min\left\{1, \frac{WD^G(w)}{WD^G(u)}\right\} \quad (2)$$

In our research problem, let u be an arbitrary vertex in a probabilistic graph G and $\mathrm{V}(u)$ denote the set of all the vertices adjacent to u in G. w is a random sample from $\mathrm{V}(u)$. If we use $WD^G(i)$ to represent the weighted degree of vertex i in G, Equation 1 can be rewritten as Equation 2, where $P(u \longrightarrow w)$ represents the probability that random walk moves from u to w. In addition, at each trial, the proposed state w for the current state u is uniformly drawn from $\mathrm{V}(u)$. By doing this, our random walk Metropolis algorithm starts from an initial vertex (usually a person whose friend relation is of interest), then moves along edges in G, and terminates once it reach a node in a k-backbone graph. Therefore, by launching the random walk Metropolis sampler, we obtain a Markov chain of random samples of nodes in G.

Here we illustrate our random walk Metropolis (RWM) algorithm by taking the graph in Figure 6 as an example. In Figure 6, grey nodes indicate the vertices in the k-backbone and dashed arrows represent a random walk by the RWM. Without loss of generality, assume that all the edges hold a weight of 1. Suppose we aim to investigate the probability that A knows N given the topology specified in Figure 6. We run the random walk Metropolis for A and N, respectively, to identify the set of their respective local closely related vertices, denoted as $local(A)$ and $local(N)$. At first, the RWM initiates a Markov chain with A as its starting point by adding A into

$local(A)$. Afterwards, the RWM proposes a new state by sampling uniformly over all the vertices adjacent to A. Because B is the only vertex directly connected to A, B will be suggested as the next state of the Markov chain. Then RWM accepts B with a probability of $min\{1, \frac{WD^G(B)}{WD^G(A)}\} = min\{1, \frac{3}{1}\} = 1$. Thus we insert B into $local(A)$. Subsequently, the RWM verifies that the termination condition is met, i.e., the RWM terminates if the current state is a k-backbone vertex. Because B is not a k-backbone vertex, the RWM continues and a new proposal state for B will be selected from $\{A, C, D\}$. Supposing D is chosen, the RWM accepts D with a probability of $min\{1, \frac{WD^G(D}{WD^G(B)}\} = min\{1, \frac{4}{3}\} = 1$. As a result, D is added into $local(A)$. Since D is a k-backbone vertex, the RWM for A terminates with D as the ending point. Similarly, we run the RWM for N and the resulting Markov chain is $local(N) = \{N, M, L, M, K, G\}$

The detailed steps of the RWM sampling method are summarized in Algorithm 2. In Algorithm 2, we initiate a Markov chain with u as the starting point. Afterwards, the RWM proposes a new state (vertex) to move to by sampling uniformly over all the adjacent vertices. Then, the RWM accepts the proposed state with a probability of the ratio of the weighted degree. After each move, the RWM verifies that the termination condition is met. The resulting Markov chain terminates when it moves to a vertex which belongs to the k-backbone graph.

By launching the RWM sampler from a particular vertex, for instance U, we can obtain a chain of vertices (nodes) locally related to U. As far as the prediction of a specific link is concerned, for example, $Query$(X, Y, $knows$), LinkProbe launches two independent runs of the RWM sampler for X and Y in order to connect their respective d-local graphs with the k-backbone graph.

Algorithm 2 The RWM sampling method (k, u, G)

Output: A Markov chain of vertices in G: $X = X_0, X_1, ..., X_i, ...$

$X_0 = u$
$L = \emptyset$
$L = L \bigcup X_0$
for i = 0 to max_tries **do**
  if X_i is a node in the k-backbone graph **then**
    Return
  else
    Obtain a random node X'_{i+1} from the adjacent list of X_i, $V(X_i)$
    Generate a random number between 0 and 1, $jitter$
    if $jitter \leq \min\left\{1, \frac{WD^G(X'_{i+1})}{WD^G(X_i)}\right\}$ **then**
      $X_{i+1} = X'_{i+1}$
    else
      $X_{i+1} = Xi$
    end if
    $L = L \bigcup X_{i+1}$
    $i++$
  end if
end for

D. Properties of the RWM Sampler

- The RWM sampler explores a probabilistic social graph locally by favoring the people who are more "influential" to the query results. First, the resulting Markov chain is ergodic so that each vertex has a certain probability to be visited. Second, the constructed Markov chain satisfies the detailed balance property. Therefore, a local vertex with a higher probabilistic degree holds a higher probability to be selected by the RWM sampler.
- The RWM algorithm is efficient in reaching the k-backbone graphs.The average length of all shortest paths between any two vertices is logarithmically small. On the other hand, the RWM algorithm tends to generate vertices with a higher probabilistic degree. Therefore, the RWM sampler can efficiently reach a k-backbone node which has a probabilistic degree of at least k.

V. Markov Logic Networks

A. Markov Networks

As one of the probabilistic graphical models, Markov networks (also called Markov random fields) can represent the full joint probability distribution D over a set of random variables $X = X_1, ...X_n$, as shown in Equation 3. In Equation 3, Φ_k and $x_{\{k\}}$ represent the potential function and the state of the k^{th} clique, respectively. Furthermore, Z is the normalizing constant and can be computed using Equation 4. Each variable can be represented as a node in the Markov network and the correlation among variables are reflected by the edges between nodes.

$$\mathrm{D} = P(X = x) = \frac{1}{Z} \prod_k \Phi_k(x_{\{k\}}) \tag{3}$$

$$Z = \sum_{x \in X} \prod_k \Phi_k(x_{\{k\}}) \tag{4}$$

Based on the full joint distribution D, we are able to answer any conditional probability query by summing the corresponding entries in D. To be specific, if we use $E = E_1, ..., E_m$ to denote the set of *evidence variables* and $Y = Y_1, ..., Y_k$ to represent the set of *query variables*, a conditional probability query of Y given the fact $E = e$ can be calculated as

$$\sum_H P(Y, H|E = e), \tag{5}$$

where $H = X - Y - E$ is a set of *hidden variables*.

B. Markov Blankets

In a Markov network, the *Markov Blanket* (MB) for node X, denoted as $MB(X)$, is the set of all the neighboring (i.e., directly connected) nodes of node X. The Markov blanket for an arbitrary node X maintains the following property: node X is conditionally independent of all other nodes in the network, given the Markov blanket for node X. This property can be represented using Equation 6. In Equation 6, $\overline{MB(X)}$ represents the Markov blanket for node X and $\overline{MB(X)}$

denotes all the nodes in the network which do not belong to the Markov blanket for node X.

$$P(X|MB(X), \overline{MB(X)}) = P(X|MB(X)) \tag{6}$$

C. Markov Logic Networks

Markov logic networks (MLN) are a powerful modeling tool which can provide the full expressiveness of probabilistic graphical models and first-order logic in a unifying representation. An MLN can be represented using a set of pairs (F_i, w_i), where F_i is a formula (a rule) in first-order logic and w_i is a real value representing the weight of the corresponding formula. The weight indicates how strong a formula is. The main idea behind MLN is that a possible world that violates a formula is not impossible but less probable.

The inference in an MLN is based on the inference in its resulting grounded Markov network. In other words, an MLN needs to be converted to a Markov network for inference by the means of creating a node for each grounded predicate and generating an edge between two nodes if the corresponding two grounded predicates appear in the same rule grounding.

In such a Markov network derived from an MLN, each node can be treated as a binary variable and each assignment of truth values to all the binary variables constitutes a possible world pw. The probability distribution of possible worlds (PW) in a ground Markov network can be defined as

$$P(PW = pw) = \frac{1}{Z} \exp(\sum_{i=1}^{F} w_i n_i(pw)), \tag{7}$$

where F is the number of formulas in the MLN, $n_i(pw)$ is the number of true groundings of F_i in the possible world pw, w_i is the weight of F_i, and Z is the normalizing constant.

D. Inference in Markov Logic Networks

As mentioned in the previous subsection, inference in an MLN is based on the inference in its derived Markov network. However, in many cases, generating an exact full joint distribution in a Markov network is *NP-Hard* and computationally intractable. In the worst case, the complexity of the algorithm in the case of n random variables is $\mathcal{O}(n2^n)$, requiring exponential time and space to construct the corresponding Markov network representation. Therefore, several approximate algorithms were developed to overcome this restriction.

1) MC-SAT: MC-SAT [8], [9] is a state-of-the-art approach to conduct approximate inference over Markov logic networks by combining *slice sampling* with satisfiability testing. Specifically, MC-SAT employs WalkSAT[2] as a procedure to sample from each slice, i.e., to sample a new state given the auxiliary variables. Furthermore, detailed balance among all the generated samples is preserved in MC-SAT so that inference calculations can be made only on the generated samples. MC-SAT has been proven to be orders of magnitude faster than standard MCMC methods, such as Gibbs sampling and simulated tempering, and is applicable to any model that can be expressed in Markov logic.

[2]Strictly speaking, a variant of WalkSAT, namely SampleSAT, is used in MC-SAT.

2) WalkSAT: Here we briefly introduce the WalkSAT solver, which is used in MC-SAT as a procedure to find a satisfying solution. *Satisfiability* is the problem of finding an assignment of truth values to all the variables that can satisfy all formulas in a knowledge base. Because *Satisfiability* is an NP-complete problem, WalkSAT serves as one of the most efficient approaches to *satisfiability* problems based on stochastic local search. Starting from a random initial state, WalkSAT repeatedly changes (flips) the truth value of a binary variable in a random unsatisfied formula. Specifically, with probability p, WalkSAT chooses the variable that maximizes the number of satisfied formula, and with probability $1 - p$, WalkSAT selects a random variable. WalkSAT has been proven to be able to solve *satisfiability* problems with thousands of random variables in a fraction of a second. The complete algorithm of a WalkSAT solver is formalized in Algorithm 3.

Algorithm 3 WalkSAT($\mathbb{C}$, max_tries)

Input: A set of logic formulas, $\mathbb{C}$, and the maximum number of tries, max_tries
Output: An assignment of all binary predicates satisfying each formula in $\mathbb{C}$

```
Randomly assign TRUE/FALSE to each binary predicate.
k = 0
for k ≤ max_tries do
    Randomly select a unsatisfied formula, F_i, in C
    With probability p, select a binary predicate in F_i and flip (change) the value of that predicate.
    With probability 1−p, select the binary predicate in F_i which can maximize the number of satisfied formulas and flip the value of that predicate.
    if all the formulas in C are satisfied then
        Return the current assignment of predicates as the solution.
    end if
    k++
end for
```

VI. Inference

A. MC-SAT$^+$

The inference in LinkProbe is based on the MC-SAT$^+$ method, which is a probabilistic extension of the well-known MC-SAT algorithm. In this section, we elaborate on how LinkProbe employs MC-SAT$^+$ to conduct the inference.

Notice that a possible world (a ground Markov network) is comprised of an assignment of truth values to all the nodes in an inference graph. Therefore, in order to instantiate a sample (a valid possible world), we need to assign truth values to all the involved nodes. Specifically, in each sampling cycle, MC-SAT$^+$ employs the WalkSAT solver and a biased sampling method to find values for *hidden nodes* and *probabilistic evidence nodes*, respectively. On the contrary, throughout all the sampling cycles, MC-SAT$^+$ keeps the values of *deterministic evidence nodes* fixed.

1) Searching for Values for Hidden Nodes: MC-SAT$^+$ launches the WalkSAT solver to find the truth values for all hidden nodes. Starting from a random initial state, WalkSAT repeatedly flips the truth value of a hidden node (can be treated

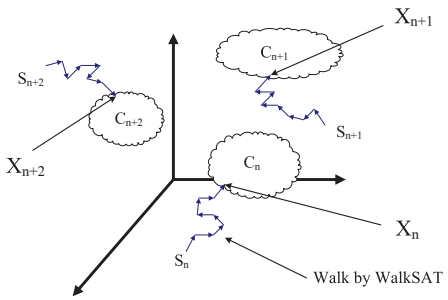


Fig. 7. An illustration of the sampling procedure in the MC-SAT$^+$ method.

as a binary random variable) in a random unsatisfied rule grounding. At each iteration in a sampling cycle of MC-SAT$^+$, WalkSAT selects a hidden variable v in a randomly chosen unsatisfied rule grounding to flip the value of v. The selection criterion is as follows. With probability p, WalkSAT chooses the hidden node that can minimize the number of unsatisfied rule groundings. With probability $1-p$, WalkSAT chooses a random hidden node in the referred rule grounding.

2) Generating Values for Probabilistic Evidence Nodes: We utilize inverse transform sampling to randomly generate values for probabilistic evidence triples based on their probability information specified in $\mathcal{DB}$. Inverse transform sampling is a basic method to draw random samples from any probability distribution given its cumulative distribution function [17]. Suppose we have a record $< from, to, prob >$ in $\mathcal{DB}$. When we need to assign values to the node representing this record, we randomly generate its value as $TRUE$ with a probability of $prob$ and as $FALSE$ with a probability of $1-prob$. Afterwards, we keep this randomly generated value unchanged during each sampling cycle of MC-SAT$^+$. The complete assignment operation of all the probabilistic evidence triples can be done by calling the procedure $biased\_sample()$.

3) Determining Values for Deterministic Evidence Nodes: For *deterministic evidence nodes*, i.e., a record $< from, to, prob >$ where $prob = 1$, we set their values as $TRUE$ and keep such values unchanged during the entire inference procedure.

4) Algorithm Description of MC-SAT$^+$: The complete algorithm of the MC-SAT$^+$ method is formalized in Algorithm 4. In Algorithm 4, each sample $x^{(i)}$ consists of *deterministic evidence nodes*, ${x_{de}}^{(i)}$, *probabilistic evidence nodes*, ${x_{pe}}^{(i)}$, and *hidden nodes*, ${x_h}^{(i)}$, i.e., $x^{(i)}$ = ${x_{de}}^{(i)} \cup {x_{pe}}^{(i)} \cup {x_h}^{(i)}$. In particular, since we keep the truth values of the *deterministic evidence nodes* unchanged during the entire sampling procedure, we rewrite ${x_{de}}^{(i)}$ as x_{de}. An illustration of the sampling in MC-SAT$^+$ is shown in Figure 7. As demonstrated in Figure 7, similar to MC-SAT [8], [9], MC-SAT$^+$ utilizes WalkSAT as a procedure to identify each valid solution X_n to each slice (constraint set) C_n from a random assignment S_n. The weight w_j for formula f_j can be approximated as $w_j = \log \frac{p_j}{1-p_j}$, where p_j is the probability that formula f_j holds. If p_j is 1, then f_j is a hard formula.

Algorithm 4 The MC-SAT$^+$ Inference Method

1: Assign truth values to x_{de}
2: ${x_{pe}}^{(0)} = biased\_sample()$
{Generate the truth values of the probabilistic evidence node}
3: ${x_h}^{(0)} = WalkSAT(hard\_formulas)$
{Find the truth values for the hidden nodes by applying WalkSAT with all hard rules as the constraints}
4: $counter$ = 0
5: **for** $i = 1$ to $num\_samples$ **do**
6: ${x_{pe}}^{(i)} = biased\_sample()$
7: C $= \emptyset$
8: **for** All $f_j \in$ formulas satisfied by $x^{(i-1)}$ **do**
9: With probability $1 - e^{-w_j}$, add f_j to C
{w_j is the weight of f_j}
10: **end for**
{Slice sampling}
11: ${x_h}^{(i)} = WalkSAT(\mathrm{C}, max\_tries)$
{Search for the valid truth values for hidden nodes by calling WalkSAT with C as the constraints. Consequently, we obtain $x^{(i)}$ as $x_{de} \cup {x_{pe}}^{(i)} \cup {x_h}^{(i)}$}
12: **if** The value of the query node in $x^{(i)}$ is true **then**
13: $counter++$
14: **end if**
15: **end for**

B. Probabilistic Inference in LinkProbe

Here we elaborate on the four steps of the inference procedure of LinkProbe for each submitted link query.

1) Deriving the k-backbone Graph: As discussed in Section III, we retrieve the k-backbone graph for a given social network G.

2) Retrieving the d-local Graphs: As discussed in Section IV, we retrieve the d-local graphs for a link query.

3) Connecting the d-local Graphs with the k-backbone Graph: For a link query, in particular $Query(X, Y, knows)$, we issue two runs of the RWM sampler from X and Y, respectively. LinkProbe collects all the discovered nodes during the RWM sampling.

4) Inferencing over Inference Subgraph: For a link query, $Query(X, Y, knows)$, its inference subgraph is comprised of the k-backbone graph, the respective d-local graphs of X and Y, and all the discovered nodes during the RWM sampling. As for each query, LinkProbe applies MC-SAT$^+$ over its inference subgraph instead of on the original graph.

C. Algorithm Description

Algorithm 5 shows how LinkProbe builds the inference subgraph for a given link query and then conducts joint inference over it. First, LinkProbe retrieves the k-backbone graph. Afterwards, LinkProbe identifies the d-local graphs. Next, LinkProbe launches random walk Metropolis samplers to connect the d-local graphs with the k-backbone graph in order to form the final inference subgraph. Subsequently, LinkProbe conducts MC-SAT$^+$ on the inference subgraph.

Algorithm 5 Inference Method in LinkProbe (d, k, X, Y)

Output: The probability p that X holds a certain relationship with Y, for example, knows(X,Y).

1: Retrieve all the vertices in the k-backbone graph of G as K
2: Identify the set of the vertices in the d-local graph of X as $\mathbb{X}$.
3: Identify the set of the vertices in the d-local graph of Y as $\mathbb{Y}$.
4: Generate a set of vertices $\mathbb{X}'$ by launching a RWM sampler from vertex X.
5: Generate a set of vertices $\mathbb{Y}'$ by launching a RWM sampler from vertex Y.
6: Construct the inference subgraph $I = K \bigcup \mathbb{X} \bigcup \mathbb{X}' \bigcup \mathbb{Y} \bigcup \mathbb{Y}'$
7: p = MC-SAT$^+$ (I, G)

VII. Error Analysis

$$\begin{cases} knows(X,Z) \wedge knows(Z,Y) & \longrightarrow knows(X,Y) \\ knows(X,T) \wedge knows(T,Y) & \longrightarrow knows(X,Y) \\ knows(X,U) \wedge knows(U,Y) & \longrightarrow knows(X,Y) \\ knows(X,V) \wedge knows(V,Y) & \longrightarrow knows(X,Y) \\ knows(X,W) \wedge knows(W,Y) & \longrightarrow knows(X,Y) \\ ... & \longrightarrow ... \\ ... & \longrightarrow ... \end{cases} \tag{8}$$

$$\begin{cases} knows(X,Z) \wedge knows(Z,Y) & \longrightarrow knows(X,Y) \\ knows(X,T) \wedge knows(T,Y) & \longrightarrow knows(X,Y) \end{cases} \tag{9}$$

In this section, we elaborate on the error induced by inferencing over the inference subgraph rather than over the original graph. We use $error$ to describe the error between our proposed inference graph-based MLN implementation, represented as $LinkProbe()$, and the original graph based MLN implementation, represented as $Naive()$, with the same number of samples drawn.

Without loss of generality, suppose we adopt the transitive property of friendship for inference. For simplicity of presentation, we focus the MLN inference on the Markov blanket of the grounding Markov network (the Markov blanket of a node contains all the variables that shield the node from the rest of the network). Our objective is to predict the acquaintance between two people, X and Y. In the naive "original graph" implementation, the constraint set for a certain slice can be represented as Equation 8, where X, Y, Z, T, U, V and W denote different people. If we have 10^6 people in the social graph, then the total number of formula groundings in Equation 8 could be very close to 10^6, which will lead to extreme inefficiency in terms of inference time and memory usage.

We aim to compare the inference result generated by the entire graph implementation $Naive()$ with that returned by the inference subgraph implementation $LinkProbe()$, with respect to the value of $knows(X,Y)$. Suppose X, Y, Z and T are the only nodes in the inference subgraph while others are not. Then, we can obtain Equation 9, which is only comprised of the nodes appearing in the inference subgraphs. For the referred slice during MC-SAT$^+$ inference, $Naive()$ implementation employs the constraint set Equation 8 while $LinkProbe()$ implementation utilizes the constraint set Equation 9. In addition, we use $\mathcal{N}$ to denote the set of the formula groundings in Equation 8 and $\mathcal{L}$ to represent the set of the formula groundings in Equation 9. Each predicate in Equation 8 and Equation 9 has three candidate values $True$, $False$ and $Unknown$.

A. Error Analysis

Case 1: All the left sides of the formula groundings in $\mathcal{N}$ are $FALSE$. In this case all the formula groundings in $\mathcal{N}$ are trivially satisfied. According to the LazySAT implementation of WalkSAT, any formula grounding trivially satisfied by evidence can be removed without affecting the final solution found by WalkSAT. Therefore, it is safe for us to delete all the formula groundings in $\mathcal{N} - \mathcal{L}$. Therefore, in this case Equation 8 and Equation 9 are equivalent with respect to the run of WalkSAT. As a result, $Naive()$ and $LinkProbe()$ will yield the same value of $knows(X,Y)$.

Case 2: At least one left side of the formula groundings in $\mathcal{N}$ is $True$ or $Unknown$.

- Case 2-A: $True$ or $Unknown$ only appears in the left side(s) of the formula groundings in $\mathcal{L}$. In this case the left sides of all the formula groundings in $\mathcal{N} - \mathcal{L}$ are $false$. As a result, it is safe to delete all the formula groundings in $\mathcal{N} - \mathcal{L}$ because they are all trivially satisfied. $Naive()$ and $LinkProbe()$ will return the same value of $knows(X,Y)$ in this case.
- Case 2-B: $True$ or $Unknown$ only appears in the left side(s) of the formula groundings in $\mathcal{N} - \mathcal{L}$. In this case all the left sides of all the formula groundings in $\mathcal{L}$ are $FALSE$. There could be an error between $Naive()$ and $LinkProbe()$ because some formula groundings in $\mathcal{N}-\mathcal{L}$ that need to be satisfied are dropped and all the formula groundings in $\mathcal{L}$ are trivially satisfied. However, because any person who knows X or Y with a probability no less than d on the probabilistic social graph G should be included in the inference subgraph (according to the definition of d-local graph), the probability that case 2-B occurs is less than $(1-d)^{(n+m)}$, where n and m are the numbers of friends on the inference subgraph. If we take d as 1, then case 2-B never happens.
- Case 2-C: $True$ or $Unknown$ appears in the left side(s) of the formula groundings in $\mathcal{L}$ and in $\mathcal{N} - \mathcal{L}$.
  - Case 2-C-1: At least one left side of the formula groundings in $\mathcal{L}$ is $TRUE$. In this case, because the formula groundings in $\mathcal{L}$ are able to determine the value of $knows(X,Y)$ as $TRUE$, it is safe to remove all the formula groundings in $\mathcal{N} - \mathcal{L}$ from $\mathcal{N}$. $Naive()$ and $LinkProbe()$ will return the same value of $knows(X,Y)$ as $TRUE$. Since $\mathcal{L}$ is comprised of predicates involving the most globally influencing people and most locally connected people, this case happens with a very high probability.
  - Case 2-C-2: All left sides of the formula groundings in $\mathcal{L}$ are $False$ or $Unknown$. In this case, the removal of all the formula groundings in $\mathcal{N} - \mathcal{L}$ may lead to a WalkSAT solution with a different value of $knows(X,Y)$, resulting in an error between

$Naive()$ and $LinkProbe()$ for the current slice sampling cycle. However, since the predicates in $\mathcal{L}$ describe the relationship between the persons of interest and their most locally related people and the most globally influencing people, each predicate in $\mathcal{L}$ is expected to be $True$ with a very high probability. Therefore, case 2-C-2 occurs only with a very low probability.

By reducing Equation 8 (corresponds to the entire graph) to Equation 9 (corresponds to the inference subgraph), LinkProbe decreases the number of formula groundings which need to be taken into account in each sampling cycle by several orders of magnitude, which leads to much higher inference efficiency and scalability without significant sacrifice in accuracy compared to the naive MLN implementation.

B. Error Upper Bound

During MC-SAT$^+$, LinkProbe first draws samples based on slice sampling, then calculates the frequency of $TRUE$ and $FALSE$ over all the samples to report the final inference result. Suppose N samples are drawn in total, then each incorrect count will induce the inference error of $\frac{1}{N}$. The error between $LinkProbe()$ and $Naive()$ can be described using Equation 10. In Equation 10, $P(k)$ is the probability that k incorrect counts occur during the inference. Because the incorrect count during the MC-SAT inference follows the Poisson Binomial distribution, the probability of having exactly k incorrect counts during N samples, $P(k)$, can be calculated as Equation 11, where p_i is the probability that case 2-C-2 occurs in the i^{th} sampling cycle, F_k is the set of all subsets of k integers that can be selected from {1,2,3,...,N}, and S^C is the complement of S. Because the mean of the Poisson Binomial distribution is $\sum_{i=1}^{N} p_i$, the error between $LinkProbe()$ and $Naive()$ can be bounded as shown in Equation 12. Notice that in Equation 12, N is very large and $\forall i, p_i$ is very small.

If we assume that the probability that case 2-C-2 occurs in each sampling cycle is identical, say P, then the incorrect count follows the Binomial distribution $B(N, P)$. In this case, Equation 10 can be rewritten as Equation 13.

$$error \leq \sum_{k=1}^{N} \frac{1}{N} \times k \times P(k) \tag{10}$$

$$P(k) = \sum_{S \in F_k} \times \prod_{i \in S} p_i \times \prod_{j \in S^C} (1 - p_j) \tag{11}$$

$$error \leq \frac{1}{N} \times \sum_{i=1}^{N} p_i \tag{12}$$

$$error \leq \sum_{k=1}^{N} \frac{1}{N} \times k \times \binom{N}{k} \times P^k \times (1 - P)^{N-k} \tag{13}$$

VIII. Experimental Validation

In order to validate the superiority of LinkProbe, we implemented LinkProbe and investigated its performance with two real-world data sets, LiveJournal (LJ) [18] and High Energy Physics (HEP) [19]. Table I shows the basic statistics of the above two data sets. We used the relation transitive property as the inference rule and the probability that the rule is true was approximated as the fraction of closed triangles. Each result on inference accuracy and efficiency is obtained by averaging 100 randomly generated link queries against the referred data set.

A. k-backbone Graphs

We retrieved 1500-backbone, 3000-backbone and 4500-backbone graphs for LJ and 200-backbone, 400-backbone and 600-backbone graphs for HEP. Table II lists the number of people involved in the above backbone graphs. Their respective social graphs are shown in Figure 8 and Figure 9, respectively. As illustrated in Figures 8 and 9, a higher k value leads to a sparser social graph, filtering out more nodes (more people) from the original data set.

B. Memory Consumption

LinkProbe is capable of inferencing over large-scale social graphs. Table III shows the approximate memory requirement of LinkProbe with data sets and k values specified. In addition, Table IV illustrates the approximate memory consumption by the MLN naive implementation.

C. Inference Accuracy

Definition *The Absolute Error* (AE): *The Absolute Error* is defined to be the difference between the predicted probability and the true probability regarding the existence of a particular link (i.e., a link query).

Definition *The Mean Absolute Error* (MAE): *The Mean Absolute Error* is defined to be the average value of the *the absolute error* over the involved link queries.

1) Social Networks: In this subsection, we investigate the performance of LinkProbe on the LiveJournal (LJ) data set in terms of inference accuracy. We retrieved the 1500-backbone graph, 3000-backbone graph and 4500-backbone graph of the LiveJournal data set. i.e., we set k as 1500, 3000, and 4500, respectively. We increased the number of samples drawn in LinkProbe to investigate its impact on inference accuracy. Figure 10 shows the results. In Figure 10, when we gradually raised the number of samples, the mean absolute error dropped accordingly. The reason is that when we draw more samples using MC-SAT$^+$, the distribution of samples (possible worlds) provided an approximation more close to the reality. On the

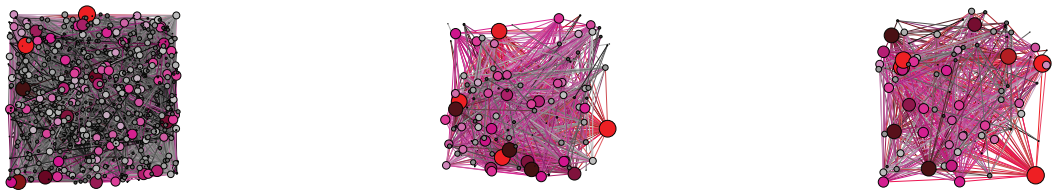
TABLE IV
Memory consumption by the MLN naive implementation.

	LJ	**HEP**
Required memory (GB)	$4.76 * 10^{11}$	$1.28 * 10^{7}$

TABLE I

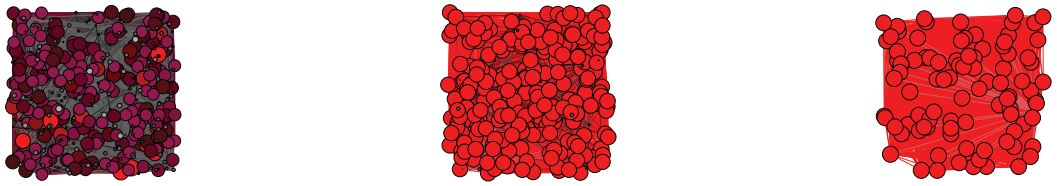
DATA SETS USED IN THE EXPERIMENTS.

Name	Description	Number of nodes	Number of edges	Average clustering coefficient	Fraction of closed triangles
LJ	social network	4,847,571	68,993,773	0.3123	28.820%
HEP	collaboration network	12,008	237,010	0.6115	65.950%



(a) k = 1500. (b) k = 3000. (c) k = 4500.

Fig. 8. The k-backbone graphs of LJ.



(a) k = 200. (b) k = 400. (c) k = 600.

Fig. 9. The k-backbone graphs of HEP.

TABLE II

NUMBER OF PEOPLE IN THE CORRESPONDING k-BACKBONE GRAPHS.

	LJ, k=1500	**LJ, k=3000**	**LJ, k=4500**	**HEP, k=200**	**HEP, k=400**	**HEP, k=600**
Number of people	652	124	46	425	252	95

TABLE III

MEMORY CONSUMPTION BY LINKPROBE.

	LJ, k=1500	**LJ, k=3000**	**LJ, k=4500**	**HEP, k=200**	**HEP, k=400**	**HEP, k=600**
Required memory (GB)	3.4638	0.3638	0.28813	2.51	0.485	0.1206

other hand, when we adopted a larger k value, the corresponding mean absolute error increased because the inference subgraph filtered out more nodes and contained a smaller fraction of the original graph.

2) Collaboration Networks: In this subsection, we investigate the performance of LinkProbe on the High Energy Physics (HEP) data set in terms of inference accuracy. We varied k from 200 to 400 to 600 and gradually increased the number of samples drawn in the inference to study their impact on inference accuracy. The experimental results are shown in Figure 11. In Figure 11, with the enlargement of the number of samples drawn, the mean absolute error is reduced accordingly. On the other hand, when we took a larger k value, the mean absolute error rose.

D. Inference Efficiency

Here we measure the end-to-end execution time (running time) counted from the time of query submission to the time returning the final inference results.

1) Social Networks: In this subsection, we investigate the performance of LinkProbe on the LiveJournal (LJ) data set in terms of inference efficiency. We varied k from 1500 to 3000 to 4500 and changed the number of samples. Figure 12 shows the results. In Figure 12, when we gradually increased the number of samples, the mean running time extended accordingly. The reason is that LinkProbe requires more time to come up with samples using MC-SAT$^+$. With a larger k value, the mean running time dropped because a smaller fraction of the original graph was included in the inference subgraph.

2) Collaboration Networks: In this subsection, we investigate the performance of LinkProbe on the High Energy Physics (HEP) data set in terms of inference accuracy. We increased k from 200 to 400 to 600 and varied the number of samples. The results are shown in Figure 13. In Figure 13, with the enlargement of the number of samples drawn in the inference, the mean running time extended. On the contrary, when we raised the k value, the mean running time dropped accordingly.

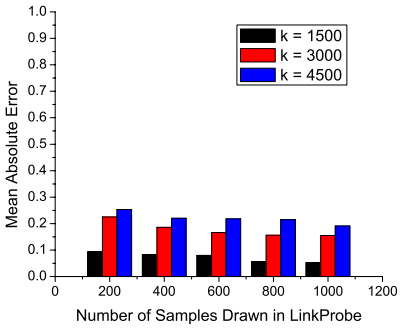


Fig. 10. The impact of number of samples on inference accuracy for LJ.

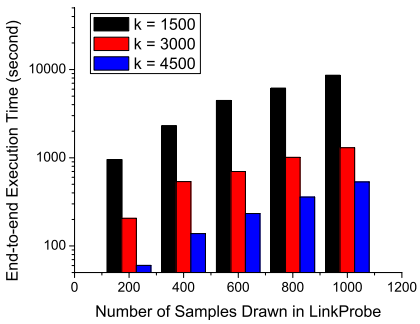


Fig. 12. The impact of number of samples on inference efficiency for LJ.

IX. Related Work

A. Sampling in Probabilistic Databases

Sampling-based approaches [17], [20], [21], [22] are proposed for managing incomplete and uncertain data in probabilistic databases [23], [24], [25], [26], [27]. The idea is simple and intuitive: we construct random samples while observing the prior statistical knowledge and constraints about the data. Thus, each sample is one possible realization (possible world) in the space of uncertainty, and the entire set of samples reveals the distribution of the uncertain data which we want to model. Queries and inferences are then conducted against this distribution. MCDB [17], for example, allows a user to define arbitrary variable generation functions that embody the database uncertainty. MCDB employs these functions to pseudorandomly generate realized values for the uncertain attributes, and evaluates queries over the realized values. However, compared to the statistical relational learning approaches, most of the above works are only focused on simpler probabilistic models for query processing.

B. Statistical Relational Learning

Techniques of utilizing statistical relational learning have attracted more and more interests from researchers due to the rapidly increasing applications on large-scale uncertain data (i.e., online data produced by users), for example, natural language processing, entity resolution, information extraction and social network analysis. Statistical relational learning is an emerging area that combines statistics, artificial intelligence and machine learning for addressing uncertainty with complex inherent structural constraints. In statistical relational learning, uncertainty is dealt with using statistical methods. Structural constraints can be represented using first-order logic. The inference in statistical relational learning is typically driven by using probabilistic graphical models, e.g., *Bayesian networks* and *Markov networks*. *Bayesian networks* represent a joint distribution of random variables as a directed acyclic graph. Its nodes are the random variables while the edges correspond to direct influence from one node to another. The BayesStore project [28] is one recent work based on *Bayesian networks* for probabilistic inference. Compared to *Bayesian networks*, *Markov networks* represent a joint probability distribution of random variables as an undirected graph, where the nodes represent the variables and the edges correspond to the direct probabilistic interaction between neighboring variables. Oliveira and Gomes [11] employed Markov logic networks for web reasoning over partially correct rules. However, one key issue of applying Markov Logic Networks in practice is its low efficiency, especially in domains involving many objects, due to the combinatorics. Some approaches were proposed to alleviate this problem. For example, Singla and Domingos [29] proposed a lazy implementation of Markov logic networks, named as LazySAT. Mihalkova and Richardson [30] presented a meta-inference algorithm that can speed up the inference by avoiding redundant computation. Shavlik and Natarajan [14] provided a preprocessing technique to reduce the effective size of inference networks. However, all the above approaches experimented only with very small-scale data sets, involving

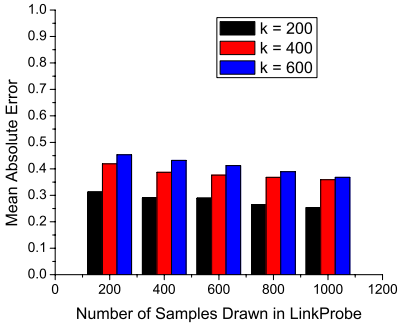


Fig. 11. The impact of number of samples on inference accuracy for HEP.

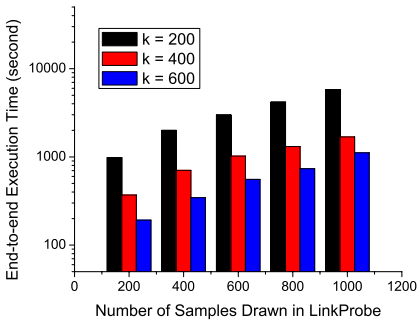


Fig. 13. The impact of number of samples on inference efficiency for HEP.

only a very limited number of objects. Our work aims at this challenge: how to apply statistical relational reasoning in large-scale problems, i.e., in a social graph which involves millions of people?

C. Link Prediction on Social Networks

As one of the core tasks in social networks, link prediction has been extensively studied. Liben-Nowell et al. [31] formalized the link prediction problem and developed approaches to link prediction based on measures for analyzing the proximity of nodes in a network. Unfortunately, they considered only the features that are based on the link structure of the network itself. Leroy et al. [32] introduced a framework to predict the structure of a social network when the network itself is totally missing while some other information regarding the nodes is available. They first generated a bootstrap probabilistic graph using any available feature and then applied the link prediction algorithms in [31] to the probabilistic graph. Backstrom et al. [33] proposed a supervised random walks based method for link prediction that performs a PageRank-like random walk on a social network so that it is more likely to visit the nodes to which new links will be created in the future. What distinguishes our effort from existing works in this trend is that our approach treats link prediction as a knowledge-based inference problem and utilizes Markov Logic Networks to capture complex and structural relation and interaction among social entities. Our system can be considered as an attempt of enabling statistical relational learning over large scale data for link prediction.

X. Conclusion and Future Work

LinkProbe is a novel prototype to predict link existence in large-scale social networks based on Markov Logic Networks. It aims to handle soft inference rules which are common in reality and was proven to scale well towards large social networks. Compared to the naive MLN implementation, LinkProbe decreases the number of formula groundings by several orders of magnitude, leading to a much higher inference efficiency and scalability without significant sacrifice in accuracy. Our extensive experiments with realistic data sets showed that LinkProbe manages to provide a tunable balance between MLN inference efficiency and inference accuracy.

Acknowledgments

This research has been funded in part by the National Science Foundation grants CNS-0831502 (CT) and CNS-0855251 (CRI) and the VSU Faculty Research Seed Grant (FRSG).

References

[1] C.-Y. Lin, N. Cao, S. Liu, S. Papadimitriou, J. Sun, and X. Yan, "SmallBlue: Social Network Analysis for Expertise Search and Collective Intelligence," in *ICDE*, 2009, pp. 1483–1486.

[2] Z. Wen and C.-Y. Lin, "How accurately can one's interests be inferred from friends," in *WWW*, 2010, pp. 1203–1204.

[3] U. Kuter and J. Golbeck, "Using probabilistic confidence models for trust inference in web-based social networks," *ACM Trans. Internet Techn.*, vol. 10, no. 2, 2010.

[4] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *KDD*, 2009, pp. 807–816.

[5] X. Song, C.-Y. Lin, B. L. Tseng, and M.-T. Sun, "Modeling and predicting personal information dissemination behavior," in *KDD*, 2005, pp. 479–488.

[6] L. Ding, L. Zhou, T. W. Finin, and A. Joshi, "How the Semantic Web is Being Used: An Analysis of FOAF Documents," in *HICSS*, 2005.

[7] J. Golbeck and M. Rothstein, "Linking Social Networks on the Web with FOAF: A Semantic Web Case Study," in *AAAI*, 2008, pp. 1138–1143.

[8] P. Domingos, "Markov logic: a unifying language for knowledge and information management," in *CIKM*, 2008, p. 519.

[9] P. Domingos, D. Lowd, S. Kok, H. Poon, M. Richardson, and P. Singla, "Just Add Weights: Markov Logic for the Semantic Web," in *URSW*, 2008, pp. 1–25.

[10] J. Wang and P. Domingos, "Hybrid Markov Logic Networks," in *AAAI*, 2008, pp. 1106–1111.

[11] P. Oliveira and P. Gomes, "Instance-based probabilistic reasoning in the semantic web," in *WWW*, 2009, pp. 1067–1068.

[12] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen, "Statsnowball: a statistical approach to extracting entity relationships," in *WWW*, 2009, pp. 101–110.

[13] J. Zhu, N. Lao, and E. P. Xing, "Grafting-light: fast, incremental feature selection and structure learning of markov random fields," in *KDD*, 2010, pp. 303–312.

[14] J. W. Shavlik and S. Natarajan, "Speeding Up Inference in Markov Logic Networks by Preprocessing to Reduce the Size of the Resulting Grounded Network," in *IJCAI*, 2009, pp. 1951–1956.

[15] S. M. Ross, *Introduction to Probability Models, Ninth Edition*. Academic Press, 2006.

[16] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An Introduction to MCMC for Machine Learning," *Machine Learning*, vol. 50, no. 1-2, pp. 5–43, 2003.

[17] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. Jermaine, and P. J. Haas, "MCDB: A Monte Carlo Approach to Managing Uncertain Data," in *SIGMOD*, 2008, pp. 687–700.

[18] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009.

[19] J. Leskovec, J. M. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *TKDD*, vol. 1, no. 1, 2007.

[20] J. Xie, J. Yang, Y. Chen, H. Wang, and P. S. Yu, "A Sampling-Based Approach to Information Recovery," in *ICDE*, 2008, pp. 476–485.

[21] H. Chen, W.-S. Ku, H. Wang, and M.-T. Sun, "Leveraging Spatio-temporal Redundancy for RFID Data Cleansing," in *SIGMOD Conference*, 2010, pp. 51–62.

[22] M. Yang, H. Wang, H. Chen, and W.-S. Ku, "Querying uncertain data with aggregate constraints," in *SIGMOD Conference*, 2011, pp. 817–828.

[23] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom, "Trio: A System for Data, Uncertainty, and Lineage," in *VLDB*, 2006, pp. 1151–1154.

[24] P. Andritsos, A. Fuxman, and R. J. Miller, "Clean Answers over Dirty Databases: A Probabilistic Approach," in *ICDE*, 2006, p. 30.

[25] L. Antova, C. Koch, and D. Olteanu, "Query Language Support for Incomplete Information in the MayBMS System," in *VLDB*, 2007, pp. 1422–1425.

[26] R. Cheng, S. Singh, and S. Prabhakar, "U-DBMS: A Database System for Managing Constantly-evolving Data," in *VLDB*, 2005, pp. 1271–1274.

[27] N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases," *The VLDB Journal*, vol. 16, no. 4, pp. 523–544, 2007.

[28] D. Z. Wang, E. Michelakis, M. N. Garofalakis, and J. M. Hellerstein, "Bayesstore: managing large, uncertain data repositories with probabilistic graphical models," *PVLDB*, vol. 1, no. 1, pp. 340–351, 2008.

[29] P. Singla and P. Domingos, "Memory-efficient inference in relational domains," in *AAAI*, 2006, pp. 488–493.

[30] L. Mihalkova and M. Richardson, "Speeding up inference in statistical relational learning by clustering similar query literals," in *ILP*, 2009, pp. 110–122.

[31] D. Liben-Nowell and J. M. Kleinberg, "The link prediction problem for social networks," in *CIKM*, 2003, pp. 556–559.

[32] V. Leroy, B. B. Cambazoglu, and F. Bonchi, "Cold start link prediction," in *KDD*, 2010, pp. 393–402.

[33] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *WSDM*, 2011, pp. 635–644.