2012 IEEE 12th International Conference on Data Mining

Inferring the Underlying Structure of Information Cascades

Bo Zong, Yinghui Wu, Ambuj K. Singh, and Xifeng Yan

University of California at Santa Barbara
{bzong, yinghui, ambuj, xyan}@cs.ucsb.edu

***Abstract*—In social networks, information and influence diffuse among users as cascades. While the importance of studying cascades has been recognized in various applications, it is difficult to observe the complete structure of cascades in practice. In this paper we study the cascade inference problem following the independent cascade model, and provide a full treatment from complexity to algorithms: (a) We propose the idea of consistent trees as the inferred structures for cascades; these trees connect source nodes and observed nodes with paths satisfying the constraints from the observed temporal information. (b) We introduce metrics to measure the likelihood of consistent trees as inferred cascades, as well as several optimization problems for finding them. (c) We show that the decision problems for consistent trees are in general NP-complete, and that the optimization problems are hard to approximate. (d) We provide approximation algorithms with *performance guarantees* on the quality of the inferred cascades, as well as heuristics. We experimentally verify the efficiency and effectiveness of our inference algorithms, using real and synthetic data.**

***Keywords*-information diffusion; cascade prediction.**

I. Introduction

In various real-life networks, users frequently exchange information and influence each other. The information (*e.g.,* messages, articles, recommendation links) is typically created by a user and spreads via links among users, leaving a trace of its propagation. Such traces are typically represented as trees, namely, *information cascades*, where (a) each node in a cascade is associated with the time step at which it receives the information, and (b) an edge from a node to another indicates that a user propagates the information to and *influences* its neighbor [3], [10], [15].

A comprehensive understanding and analysis of cascades benefits various emerging applications in social networks [5], [11], viral marketing [1], [18], and recommendation networks [16]. In order to model the propagation of information, various *cascade models* have been developed [8], [22], [24]. Among the most widely used models is the *independent cascade model* [11], where each node has only one chance to influence its inactive neighbors, and each node is influenced by at most one of its neighbors independently. Nevertheless, it is typically difficult to observe entire cascades in practice, due to the noisy graphs with missing data, or data privacy policies [20]. This calls for techniques that can *infer* the cascades using partial information. Although cascade models and a set of related problems, *e.g.,* influence maximization, have been widely studied, much less is known on how to infer the cascade structures based on partial information, including complexity bounds and approximation algorithms.

In this paper we investigate the cascade inference problem, where cascades follow the widely used *independent cascade model*. The paper is organized as follows. In Section II, we introduce the notions of (*perfect* and *bounded*) *consistent trees* to capture the missing structures of partial cascades, as well as a measure for their quality. We investigate the problems of identifying perfect and bounded consistent trees, given partial observations, in Section III and Section IV, respectively. We show that these problems are all NP-complete, and hard to approximate. Nevertheless, we provide heuristics for these problems. We experimentally verify the effectiveness and efficiency of our inference algorithms in Section V.

Related work. There has been recent work on cascade prediction and inference, with the emphasis on global properties (*e.g.,* cascade nodes, width, size) [4], [7], [9], [14], [20], [22], [24]. All the above works focus on predicting the nodes and their behavior in cascades. In contrast, we propose approaches to infer both the nodes and the topology of the cascades in the graph-time domain. Closer to our work is the work by Sadikov et al. [20] that considers cascade prediction on k-trees. In contrast, we model cascades as general trees, and infer the nodes as well as topology of the cascades only from a set of nodes and their activation time, using much less available information. In addition, the temporal information in the partial observations is not considered in [20]. More related work is discussed in [26].

II. Consistent Trees

We start by introducing several notions.

Diffusion graph. We denote a social network as a *directed* graph $G = (V, E, f)$, where (a) V is a finite set of nodes, and each node $u \in V$ denotes a user; (b) $E \subseteq V \times V$ is a finite set of edges, where each edge $(u, v) \in E$ denotes a social connection via which the information may diffuse from u to v; and (c) a *diffusion function* $f : E \to R^+$ which assigns for each edge $(u, v) \in E$ a value $f(u, v) \in [0, 1]$, as the probability that node u *influences* v.

Cascades. We first review the *independent cascade model* [11]. We say a cascade unfolds over a graph G

1550-4786 2012

DOI 10.1109/ICDM.2012.100

following the *independent cascade model* if (a) at any time step, each node in G is exactly one of the three states {*active*, *newly active*, *inactive*}; (b) a cascade starts from a *source node* s being *newly active* at time step 0; (c) a *newly active* node u at time step t has only one chance to influence its *inactive* neighbors, such that at time $t+1$, (i) if v is an inactive neighbor of u, v becomes *newly active* with probability $f(u,v)$; and (ii) the state of u changes from *newly active* to *active*, and cannot influence any neighbors afterwards; and (d) each *inactive* node v can be influenced by at most one of its *newly active* neighbors independently, and the neighbors' attempts are sequenced in an arbitrary order. Once a node is *active*, it cannot change its state.

Based on the independent cascade model, we define a *cascade* C over graph $G = (V, E, f)$ as a *directed tree* $(V_c, E_c, s, \mathcal{T})$ where (a) $V_c \subseteq V$, $E_c \subseteq E$; (b) $s \in V_c$ is the *source node* from which the information starts to propagate; and (c) function $\mathcal{T}$ assigns for each node $v_i \in V_c$ a *time step* t_i, which represents that v_i is *newly active* at time step t_i.

Intuitively, a cascade is a tree representation of the "trace" of the information propagation from a source node s to a set of influenced nodes. Indeed, one may verify that any cascade from s following the model is a tree rooted at s.

Partial observation. Given a cascade $C = (V_c, E_c, s, \mathcal{T})$, a pair (v_i, t_i) is an *observation point*, if $v_i \in V$ is known (observed) to be *newly active at* or *by* time step t_i. A *partial observation* X is a set of observation points. Specifically, X is a *complete observation* if for any $v \in V_c$, there is an observation point $(v, t) \in X$. To simplify the discussion, we also assume that pair $(s, 0) \in X$ where s is the source node. The techniques developed in this paper can be adapted to the case where the source node is unknown.

We are now ready to introduce the idea of consistent trees.

A. Consistent trees

Given a partial observation X of a graph $G = (V, E, f)$, a *bounded consistent tree* $T_s = (V_{T_s}, E_{T_s}, s)$ *w.r.t.* X is a directed subtree of G with root $s \in V$, such that for *every* $(v_i, t_i) \in X$, $v_i \in V_{T_s}$, and s reaches v_i by t_i hops, *i.e.,* there exists a path of length *at most* t_i from s to v_i. Specifically, we say a consistent tree is a *perfect consistent tree* if for every $(v_i, t_i) \in X$ and $v_i \in V_{T_s}$, there is a path of length *exactly* t_i from s to v_i.

Intuitively, consistent trees represent possible cascades which conform to the independent cascade model, as well as the partial observation. Note the following: (a) the path from the root s to a node v_i in a bounded consistent tree T_s is not necessarily a shortest path from s to v_i in G, as observed in [13]; (b) perfect consistent trees model cascades when the partial observation is accurate, *i.e.,* each time t_i in an observation point (v_i, t_i) is exactly the time when v_i is newly active; in contrast, in bounded consistent trees, an observation point (v_i, t_i) indicates that node v_i is newly active at the time step $t_i' \leq t_i$, due to possible *delays* during information propagation, as observed in [5].

B. Cascade inference problem

We now introduce the general cascade inference problem. Given a social graph G and a partial observation X, the *cascade inference problem* is to determine whether there exists a consistent tree T *w.r.t.* X in G.

There may be multiple consistent trees for a partial observation, so one often wants to identify the best consistent tree. We next provide two quantitative metrics to measure the quality of the inferred cascades. Let $G = (V, E, f)$ be a social graph, and X be a partial observation.

Minimum weighted consistent trees. In practice, one often wants to identify consistent trees that are close to the actual cascades. Recall that each edge $(u, v) \in E$ in a given network G carries a value assigned by a diffusion function $f(u, v)$, which indicates the probability that u influences v. Based on $f(u, v)$, we introduce a *likelihood function* as a quantitative metric for consistent trees.

Likelihood function. Given a graph $G = (V, E, f)$, a partial observation X and a consistent tree $T_s = (V_{T_s}, E_{T_s}, s)$, the *likelihood* of T_s, denoted as $L_X(T_s)$, is defined as:

$$L_X(T_s) = \mathbb{P}(X \mid T_s) = \prod_{(u,v) \in E_{T_s}} f(u, v). \qquad (1)$$

Following common practice, we opt to use the log-likelihood metric, where

$$L_X(T_s) = \sum_{(u,v) \in E_{T_s}} \log f(u, v)$$

Given G and X, a natural problem is to find the consistent tree with maximum likelihood. Using log-likelihood, the *minimum weighted consistent tree* problem is to identify the consistent tree T_s with the minimum $-L_X(T_s)$.

Minimum consistent trees. In some cases, one may simply want to find the *minimum* structure that represents a cascade [17]. The minimum consistent tree, as a special case of the minimum weighted consistent tree, depicts the smallest cascades with the fewest communication steps to pass the information to all the observed nodes. In other words, this metric favors consistent trees with the fewest edges.

Given G and X, the *minimum consistent tree* problem is to find the minimum consistent trees in G *w.r.t.* X.

In the following sections, we investigate the cascade inference problem, and their optimization problems.

III. Cascades as perfect trees

As remarked earlier, when the partial observation X is accurate, one may want to infer the cascade structure via *perfect consistent trees*. The *minimum* (resp. *weighted*) *perfect consistent tree* problem, denoted as $\mathsf{PCT}_{\mathsf{min}}$ (resp.

$\mathsf{PCT_w}$) is to find the perfect consistent trees with minimum size (resp. maximum log-likelihood) as the quality metric.

It is desirable that the $\mathsf{PCT_{min}}$ and $\mathsf{PCT_w}$ problems can be solved in polynomial time. Nevertheless, the following result shows that these problems are nontrivial.

Proposition 1: Given a graph G and a partial observation X, (a) it is NP-complete to determine whether there is a perfect consistent tree *w.r.t.* X in G; and (b) the $\mathsf{PCT_{min}}$ and $\mathsf{PCT_w}$ problems are NP-complete and APX-hard.

One may verify Proposition 1(a) by a reduction from the Hamiltonian path problem [23]. We prove Proposition 1(b) by constructing an *approximation preserving reduction (*AFP-reduction*)* [23] from the minimum directed steiner tree (MST) problem to $\mathsf{PCT_{min}}$. The APX-hard problems are APX problems to which every APX problem can be reduced [23].

A. Bottom-up searching algorithm

Given the above intractability and approximation hardness result, we introduce a heuristic WPCT for $\mathsf{PCT_w}$ problem. The idea is to (a) generate a "backbone network" G_b of G which contains all the nodes and edges that are possible to form a perfect consistent tree, using a set of *pruning rules*, and also rank the observed nodes in G_b with the descending order of their time step in X, and (b) perform a bottom-up evaluation for each time step in G_b using a local-optimal strategy, following the descending order of the time step.

Backbone network. We consider pruning strategies to reduce the nodes and the edges that are not possible to be in any perfect consistent trees, given a graph $G = (V, E, f)$ and a partial observation $X = \{(v_1, t_1), \ldots, (v_k, t_k)\}$. We define a backbone network $G_b = (V_b, E_b)$, where

- $V_b = \bigcup\{v_j | \mathsf{dist}(s, v_j) + \mathsf{dist}(v_j, v_i) \leq t_i\}$ for each $(v_i, t_i) \in X$; and
- $E_b = \{(v', v) | v' \in V_b, v \in V_b, (v', v) \in E\}$

Intuitively, G_b includes all the possible nodes and edges that may appear in a perfect consistent tree for a given partial observation. In order to construct G_b, a set of *pruning rules* can be developed as follows: for a node v' in G, if every observed node v in a cascade with time step t has $\mathsf{dist}(s, v') + \mathsf{dist}(v', v) > t$, then v' and all the edges connected to v' can be safely pruned.

Algorithm. Algorithm WPCT is as shown in Fig. 1. It starts by initializing a tree T (line 1), by inserting all the observation points into T. Each node v in T is assigned with a *level* $l(v)$ equal to its time step as in X. The edge set is set to empty. It then constructs a backbone network G_b with the pruning rules (lines 2-9). It initializes a node set V_b within t_{max} hop of the source node s, where t_{max} is the maximum time step in X (line 2). If there exists some node $v \in X$ that is not in V_b, the algorithm returns $\emptyset$, since there is no path from s reaching v with t steps for $(v, t) \in X$ (line 3).

Input: graph G and partial observation X.
Output: a perfect consistent tree T in G.

1. tree $T = (V_T, E_T)$, where $V_T := \{v | (v, t) \in X\}$, set level $l(v) := t$ for each $(v, t) \in X$, $E := \emptyset$;
2. set $V_b := \{v_b | \mathsf{dist}(s, v_b) \leq t_{max}\}$;
3. **if** there is a node v in X and $v \notin V_b$ **then return** $\emptyset$;
4. set $E_b := \{(v', v) | (v', v) \in E, v' \in V_b, v \in V_b)\}$;
5. **for each** $v \in V_b$ **do**
6. **if** there is no $(v_i, t_i) \in X$ that $\mathsf{dist}(s, v) + \mathsf{dist}(v, v_i) \leq t_i$ **then**
7. $V_b = V_b \setminus \{v\}$;
8. $E_b = E_b \setminus \{(v_1, v_2)\}$ where $v_1 = v$ or $v_2 = v$;
9. graph $G_b := (V_b, E_b)$;
10. list $L := \{(v_1, t_1), \ldots, (v_k, t_k)\}$ where $t_i \leq t_{i+1}$, $(v_i, t_i) \in X$, $i \in [1, k-1]$;
11. **for each** $i \in [1, t_{max}]$ following descending order **do**
12. $V_t := V_1 \cup V_2 \cup V_3$, $V_1 := \{v_i | (v, t_i) \in X\}$; $V_2 := \{v | v \in V_T, l(v) = t_i\}$; $V_3 := \{v' | (v', v) \in E_b, v \in V_1 \cup V_2, v' \notin V_T\}$;
13. $E_t := \{(v', v) | v' \in V_3, v \in V_1 \cup V_2, (v', v) \in E_b\}$;
14. construct $G_t = (V_t, E_t)$;
15. $T := T \cup \mathsf{PCT_I}(G_t, V_1 \cup V_2, V_3, i)$;
16. **if** T is a tree **then return** T;
17. **return** $\emptyset$;

Figure 1: Algorithm WPCT: pruning and local searching

It further removes the redundant nodes and edges that are not in any perfect trees, using the pruning rules (lines 5-8). The network G_b is then constructed (line 9). The partial observation X is also sorted *w.r.t.* the time step (line 10).

Following a bottom-up greedy strategy, WPCT then processes each observation point (lines 11-17). For each i in $[1, t_{max}]$, it generates a (bipartite) graph G_t. (a) It initializes a node set V_t as the union of three sets of nodes V_1, V_2 and V_3 (line 12), where (i) V_1 is the nodes in X with time step t_i, (ii) V_2 is the nodes v in the current perfect consistent tree T with level $l(v) = t_i$, and (iii) V_3 is the set of possible parents for the nodes in V_1 and V_2. (b) It constructs an edge set E_t which consists of the edges from V_3 to V_1 and V_2. (c) It then generates G_t with V_t and the edge set E_t, which is a bipartite graph. After G_t is constructed, the algorithm WPCT invokes procedure $\mathsf{PCT_I}$ to compute a "part" of the perfect tree T, which is an *optimal* solution for G_t, a part of the graph G_b which contains all the observed nodes with time step t_i. It expands T with the returned partial tree (line 15). The above process (lines 11-15) repeats for each $i \in [1, t_{max}]$ until all the nodes in X are processed. Algorithm WPCT then checks if the constructed T is a tree. If so, it returns T as a perfect tree; otherwise, it returns $\emptyset$ (line 17).

Procedure $\mathsf{PCT_I}$. Given a (bipartite) graph G_t, and two sets of nodes V and V_s in G_t, the procedure $\mathsf{PCT_I}$ [26] computes for G_t a set of trees $T_t = \{T_1, \ldots, T_L\}$ with the minimum total weight, where (a) each T_i is a 2-level tree with a root in V_s and leaves in V, (b) the leaves of any two trees in T_t are disjoint, and (c) the trees cover all the nodes in V. The weight of T_i consists of the edge weights, and the root weight which is estimated by the minimum-cost path from

the cascade root s to T_i's root. For each T_i, $\mathsf{PCT_I}$ assigns its root r in V_s a level $l(r) = t_i - 1$. T_t is then returned as a part of the entire perfect consistent tree. In practice, we may either employ linear programming, or an algorithm for MST problem (*e.g.,* [19]) to compute T_t.

Discussion. The algorithm WPCT either returns $\emptyset$, or correctly computes a perfect consistent tree *w.r.t.* the partial observation X. Indeed, one may verify that (a) the pruning rules only remove the nodes and edges that are not in any perfect consistent tree *w.r.t.* X, and (b) WPCT has the loop invariant that at each iteration i (lines 11-15), it always constructs a part of a perfect tree as a forest. One may further verify that WPCT runs in time $O(|V||E|+|X|^2+t_{max}*\mathcal{A})$, where t_{max} is the maximum time step in X, and $\mathcal{A}$ is the time complexity of procedure $\mathsf{PCT_I}$. The bottom-up construction runs in $O(|t_{max} * \mathcal{A}|)$, which is further bounded by $O(|t_{max} * |V|^3)$ if an approximable algorithm is used [19]. In our experimental study, we utilize efficient linear programming to compute the *optimal* steiner forest.

The algorithm WPCT can easily be adapted for $\mathsf{PCT_{min}}$ problem for minimum perfect consistent trees, treating the weight on each edge as unit weight.

Perfect consistent SP trees. The independent cascade model may be an overkill for real-life applications, as observed in [6], [12]. Instead, one may identify the consistent trees which follow the shortest path model [12], where cascades propagate following shortest paths. We define a *perfect shortest path (*sp*) tree* rooted at a given source node s as a perfect consistent tree, such that for each observation point $(v, t) \in X$ of the tree, $t = \mathsf{dist}(s, v)$; in other word, the path from s to v in the tree is a *shortest path* in G. The $\mathsf{PCT_w}$ (resp. $\mathsf{PCT_{min}}$) problem for sp trees is to identify the sp trees with the maximum likelihood (resp. minimum size).

Proposition 2: Given a graph G and a partial observation X, (a) it is in PTIME to find a sp tree *w.r.t.* X; (b) the $\mathsf{PCT_{min}}$ and $\mathsf{PCT_w}$ problems for perfect sp trees are NP-hard and APX-hard; (c) the $\mathsf{PCT_w}$ problem is approximable within $O(d * \frac{\log f_{min}}{\log f_{max}})$, where d is the diameter of G, and f_{max} (resp. f_{min}) is the maximum (resp. minimum) value defined by the diffusion function f.

We next provide an approximation algorithm for the $\mathsf{PCT_w}$ problem on sp trees. Given a graph G and a partial observation X, the algorithm, denoted as $\mathsf{WPCT_{sp}}$ (not shown), first constructs the backbone graph G_b as in the algorithm WPCT. It then constructs node sets $V_r = \{v|(v, t) \in X\}$, and $V_s = V \setminus V_r$. Treating V_r as required nodes, V_s as steiner nodes, and the log-likelihood function as the weight function, $\mathsf{WPCT_{sp}}$ approximately computes an undirected minimum steiner tree T. If the directed counterpart T' of T in G_b is not a tree, $\mathsf{WPCT_{sp}}$ transforms T' to a tree: for each node v in T' with more than one parent, it (a) connects s and v via a shortest path, and (b) removes redundant edges attached to v. It then returns T' as an sp tree.

Input: graph G and partial observation X.
Output: a bounded consistent tree T in G.

1. tree $T = (V_t, E_t)$, where $V_t := \{s|(s, 0) \in X\}$, $E_t := \emptyset$;
2. compute t_k bounded BFS DAG G_d of s in G;
3. **for each** $t_i \in [t_1, t_k]$ **do**
4. **for each** node v where $(v, t_i) \in X$ and $l(v) = i$ **do**
5. **if** $i > t_i$ **then return** $\emptyset$;
6. find a path ρ from s to v with the minimum weight $w(\rho) = -\Sigma \log f(e)$ for each $e \in \rho$;
7. $T = T \cup \rho$;
8. **return** T as a bounded consistent tree;

Figure 2: Algorithm WBCT

One may verify that (a) T' is a perfect sp tree *w.r.t.* X, (b) the weight $-L_X(T')$ is bounded by $O(d * \frac{\log f_{min}}{\log f_{max}})$ times of the optimal weight, and (c) the algorithm runs in $O(|V^3|)$ time, leveraging the approximation algorithm for the steiner tree problem [23]. Moreover, $\mathsf{WPCT_{sp}}$ can be used for the problem $\mathsf{PCT_{min}}$ for sp trees, where each edge in G has the same weight. This achieves an approximation ratio of d.

IV. Cascades as bounded trees

In this section, we investigate the cascade inference problems for bounded consistent trees. In contrast to the intractable counterpart in Proposition 1(a), the problem of finding a bounded consistent tree for a given graph and a partial observation is in PTIME.

Proposition 3: For a given graph G and a partial observation X, there is a bounded consistent tree in G *w.r.t.* X if and only if for each $(v, t) \in X$, $\mathsf{dist}(s, v) \leq t$, where $\mathsf{dist}(s, v)$ is the distance from s to v in G.

Given a graph G and a partial observation X, the *minimum weighted bounded consistent tree* problem, denoted as $\mathsf{BCT_w}$, is to identify the bounded consistent tree T_s^* *w.r.t.* X with the maximum likelihood $L_X(T_s^*)$. Using log-likelihood measurement, the $\mathsf{BCT_w}$ problem is to identify T_s^* with the minimum $-\log L_X(T_s^*)$ (see Section II).

Theorem 1: Given a graph G and a partial observation X, the $\mathsf{BCT_w}$ problem is
(a) NP-complete and APX-hard; and
(b) approximable within $O(|X| * \frac{\log f_{min}}{\log f_{max}})$, where f_{max} (resp. f_{min}) is the maximum (resp. minimum) value defined by the diffusion function f over G.

One may show the hardness result in Theorem 1(a) by constructing (1) a polynomial time reduction from the exact 3-cover problem (X3C), and (2) an AFP-reduction from the minimum directed steiner tree (MST) problem.

We next provide an algorithm for the $\mathsf{BCT_w}$ problem, which runs in *linear time w.r.t.* the size of G, and with performance guarantee as in Theorem 1(b).

Algorithm. The algorithm, denoted as WBCT, is illustrated in Fig. 2. Given a graph G and a partial observation X, the algorithm first initializes a tree $T = (V_t, E_t)$ with the single

source node s (line 1). It then computes the t_k bounded BFS directed acyclic graph (DAG) [2] G_d of the source node s, where t_k is the maximum time step of the observation points in X, and G_d is a DAG induced by the nodes and edges visited by a BFS traversal of G from s (line 2). Following a top-down strategy, for each node v of $(v,t) \in X$, WBCT then (a) selects a path ρ with the minimum $-\Sigma \log f(e)$ from s to v, and (b) extends the current tree T with the path ρ (lines 3-7). If for some observation point $(v,t) \in T$, $\mathsf{dist}(s,v) > t$, then WBCT returns $\emptyset$ (line 5). Otherwise, the tree T is returned (line 8).

Correctness and complexity. One may verify that algorithm WBCT either correctly computes a bounded consistent tree T, or returns $\emptyset$. The algorithm runs in time $O(|E|)$, since it visits each edges at most once following a BFS traversal.

We next show the approximation ratio in Theorem 1(b). Observe that for a single node v in X, (a) the total weight of the path w from s to v is no greater than $|w| \log f_{min}$, where $|w|$ is the length of w; and (b) the weight of the counterpart of w in T^*, denoted as w', is no less than $|w^*| \log f_{max}$. Also observe that $|w| \leq |w^*|$. Thus, $w/w^* \leq \frac{\log f_{min}}{\log f_{max}}$. As there are in total $|X|$ such nodes, $L_X(T)/L_X(T^*) \leq |X| \frac{w}{w^*} \leq |X| \frac{\log f_{min}}{\log f_{max}}$. Theorem 1(b) thus follows.

Minimum bounded consistent tree. As remarked earlier, one may simply want to identify a bounded consistent tree with minimum total number of nodes and edges. Given a social graph G and a partial observation X, the *minimum bounded consistent tree problem*, denoted as $\mathsf{BCT_{min}}$, is to identify such a tree in G *w.r.t.* X.

The $\mathsf{BCT_{min}}$ problem is a special case of $\mathsf{BCT_w}$. We summarize the main result as follows.

Proposition 4: The $\mathsf{BCT_{min}}$ problem is (a) NP-complete, (b) APX-hard, and (c) approximable within $O(|X|)$.

V. Experiments

We next present an experimental study to evaluate the effectiveness and efficiency of the proposed algorithms.

Experimental setting. We used both real-life and synthetic datasets. For real life data we used (1) the *Enron email cascades*[1], a social graph of $86,808$ users and $660,642$ edges as email forwarding relation, and (2) *Retweet cascades* (RT)[2] [25]. For Enron dataset, We tracked the *forwarded* messages of the same subjects and obtained 260 cascades of depth no less than 3 with more than 8 nodes. For Retweet dataset, we obtained 321 cascades of depth more than 4, with node size ranging from 10 to 81, by extracting the retweet cascades of the identified *hashtags* [25]. We used the EM algorithm from [21] to estimate the diffusion function. We also randomly generated a set of synthetic cascades unfolding in an anonymous Facebook social graph[3]. For all the datasets, we define *uncertainty* of a cascade T as $\sigma = 1 - \frac{|X|}{|V_T|}$, where $|V_T|$ is the node number in T, and $|X|$ is the size of the partial observation X. We remove the nodes from the given cascades until the uncertainty is satisfied, and collect the remaining nodes and their time steps as X.

We have implemented the following in C++: (i) algorithms WPCT, and WBCT; (ii) two algorithms $\mathsf{PCT_{lp}}$ and $\mathsf{BCT_{lp}}$, where $\mathsf{PCT_{lp}}$ (resp. $\mathsf{BCT_{lp}}$) identifies the optimal weighted bounded (resp. perfect) consistent trees using linear programming, respectively; (iii) two randomized algorithms $\mathsf{PCT_r}$ and $\mathsf{BCT_r}$, where $\mathsf{PCT_r}$ is similar to WPCT except for randomly selecting the steiner forest at each level (see Section III); as WBCT does, $\mathsf{BCT_r}$ runs on bounded BFS directed acyclic graphs, but randomly selects edges; and (iv) an algorithm $\mathsf{PCT_g}$ which uses a greedy strategy to select the steiner forest at each level (see Section III). We used a machine powered by an Intel(R) Core 2.8GHz CPU and 8GB of RAM, using Ubuntu 10.10.

Experimental results. We next present our findings.

Effectiveness of consistent trees. Given a set of real life cascade $\mathbf{T} = \{T_1, \ldots, T_k\}$, for each cascade $T_i = (V_{T_i}, E_{T_i}) \in \mathbf{T}$, we computed an inferred cascade $T_i' = (V_{T_i'}, E_{T_i'})$ according to a partial observation with uncertainty σ. Denote the nodes in the partial observation as V_X. We evaluated the *precision* as $\mathsf{prec} = \frac{\Sigma(|(V_{T_i'} \cap V_{T_i}) \setminus V_X|)}{\Sigma(|V_{T_i'} \setminus V_X)|}$, and $\mathsf{rec} = \frac{\Sigma(|(V_{T_i'} \cap V_{T_i}) \setminus V_X|)}{\Sigma(|V_{T_i} \setminus V_X)|}$. For Enron dataset, we found that WPCT outperforms $\mathsf{PCT_g}$ and $\mathsf{PCT_r}$ on both prec and rec, as shown in Fig. 3(a) and Fig. 3(b), respectively. When the uncertainty increases, both the prec and rec of the three algorithms decrease. In particular, WPCT successfully infers cascade nodes with prec no less than 70% and rec no less than 25% even when 85% of the nodes in the cascades are removed.

Using the same setting, both $\mathsf{BCT_{lp}}$ and WBCT outperform $\mathsf{BCT_r}$, and their prec and rec decrease while the uncertainty increases, as shown in Fig. 3(c) and Fig. 3(d), respectively. The performance of these algorithms over retweet cascades [26] verifies our observations.

Efficiency. In all the tests over real-life datasets, $\mathsf{PCT_r}$, $\mathsf{BCT_r}$, $\mathsf{PCT_g}$ and WBCT take less than 1 second. $\mathsf{BCT_{lp}}$ and $\mathsf{PCT_{lp}}$ do not scale for these datasets. In our tests, the efficiency of all the algorithms are not sensitive *w.r.t.* the changes to σ. The results over synthetic data also verify the effectiveness and scalability of our methods.

In summary, our inference algorithms can infer cascades effectively based on partial observation, and scale well with the sizes of the cascades. For more details, please refer to [26].

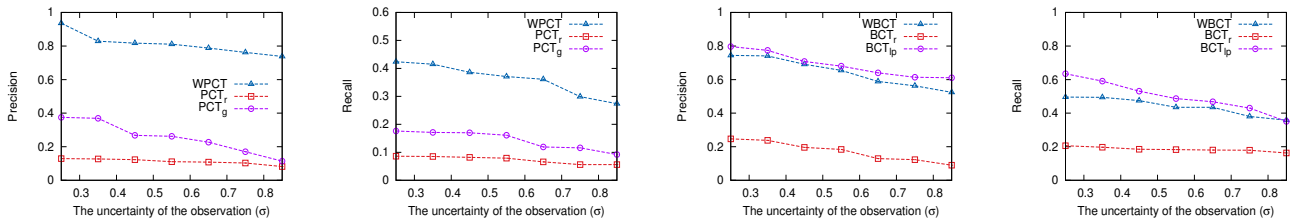
VI. Conclusion

In this paper, we investigated cascade inference problem based on partial observation. We proposed consistent trees

[1] http://www.cs.cmu.edu/∼enron

[2] http://snap.stanford.edu/data/twitter7.html

[3] http://current.cs.ucsb.edu/socialnets



(a) PCT@Enron: prec (b) PCT@Enron: rec (c) BCT@Enron: prec (d) BCT@Enron: rec

Figure 3: The precision and recall of the inference algorithms over Enron cascades

Problem	Complexity	Approximation	time
$\mathsf{BCT}_{\mathsf{min}}$	NP-c, APX-hard	$\lvert X\rvert$	$O(\lvert E\rvert)$
$\mathsf{BCT}_{\mathsf{w}}$	NP-c,APX-hard	$\lvert X\rvert * \frac{\log f_{max}}{\log f_{min}}$	$O(\lvert E\rvert))$
$\mathsf{PCT}_{\mathsf{min}}$ (sp tree)	NP-c, APX-hard	d	$\lvert V^3\rvert$
$\mathsf{PCT}_{\mathsf{w}}$ (sp tree)	NP-c, APX-hard	$d * \frac{\log f_{max}}{\log f_{min}}$	$\lvert V^3\rvert$
$\mathsf{PCT}_{\mathsf{min}}$	NP-c, APX-hard	–	$O(\lvert t_{max} * \lvert V\rvert^3)$
$\mathsf{PCT}_{\mathsf{w}}$	NP-c, APX-hard	–	$O(\lvert t_{max} * \lvert V\rvert^3)$

Table I. Summary of the results

as well as quantitative metrics for capturing the inferred cascades. We have established the hardness results for the optimization problems as summarized in Table I. Despite the hardness, we provide heuristics for these problems, with performance guarantees on inference quality. Experimental results show that our methods are able to efficiently and effectively infer partially observed cascades.

Acknowledgment. This research was sponsored in part by the U.S. National Science Foundation under grant IIS-1219254 and by the Army Research Laboratory under cooperative agreement W911NF-09-2-0053 (NS-CTA). 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.

REFERENCES

[1] D. Arthur, R. Motwani, A. Sharma, and Y. Xu. Pricing strategies for viral marketing on social networks. *Internet and Network Economics*, pages 101–112, 2009.

[2] J. Bang-Jensen and G. Z. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, 2008.

[3] S. Bikhchandani, D. Hirshleifer, and I. Welch. A theory of fads, fashion, custom, and cultural change as informational cascades. *Journal of political Economy*, pages 992–1026, 1992.

[4] C. Budak, D. Agrawal, and A. El Abbadi. Limiting the spread of misinformation in social networks. In *WWW*, 2011.

[5] M. Cha, F. Benevenuto, Y.-Y. Ahn, and P. K. Gummadi. Delayed information cascades in flickr: Measurement, analysis, and modeling. *Computer Networks*, 56(3):1066–1076, 2012.

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

[7] F. Chierichetti, J. M. Kleinberg, and D. Liben-Nowell. Reconstructing patterns of information diffusion from incomplete observations. In *NIPS*, pages 792–800, 2011.

[8] K. Dave, R. Bhatt, and V. Varma. Modelling action cascades in social networks. In *AAAI*, 2011.

[9] H. Fei, R. Jiang, Y. Yang, B. Luo, and J. Huan. Content based social behavior prediction: a multi-task learning approach. In *CIKM*, 2011.

[10] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, pages 211–223, August 2001.

[11] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

[12] M. Kimura and K. Saito. Tractable models for information diffusion in social networks. *PKDD*, pages 259–271, 2006.

[13] G. Kossinets, J. Kleinberg, and D. Watts. The structure of information pathways in a social communication network. In *SIGKDD*, 2008.

[14] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila. Finding effectors in social networks. In *KDD*, pages 1059–1068, 2010.

[15] J. Leskovec, M. McGlohon, C. Faloutsos, N. S. Glance, and M. Hurst. Patterns of cascading behavior in large blog graphs. In *SDM*, 2007.

[16] J. Leskovec, A. Singh, and J. Kleinberg. Patterns of influence in a recommendation network. *Advances in Knowledge Discovery and Data Mining*, pages 380–389, 2006.

[17] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In *SIGKDD*, pages 529–537, 2011.

[18] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *SIGKDD*, 2002.

[19] G. Robins and A. Zelikovsky. Tighter bounds for graph steiner tree approximation. *SIAM J. Discrete Math.*, 19(1), 2005.

[20] E. Sadikov, M. Medina, J. Leskovec, and H. Garcia-Molina. Correcting for missing data in information cascades. In *WSDM*, 2011.

[21] K. Saito, R. Nakano, and M. Kimura. Prediction of information diffusion probabilities for independent cascade model. In *KES*, 2008.

[22] X. Song, Y. Chi, K. Hino, and B. L. Tseng. Information flow modeling based on diffusion rate for prediction and ranking. In *WWW*, 2007.

[23] V. V. Vazirani. *Approximation Algorithms*. 2001.

[24] F. Wang, H. Wang, and K. Xu. Diffusive Logistic Model Towards Predicting Information Diffusion in Online Social Networks. *ArXiv e-prints*, 2011.

[25] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM, 2011.

[26] B. Zong, Y. Wu, A. K. Singh, and X. Yan. Inferring the underlying structure of information cascades. *arXiv preprint arXiv:1210.3587*, 2012.