Inferring the Underlying Structure of Information Cascades

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

Department of Computer Science
University of California at Santa Barbara
Santa Barbara, CA 93106-5110, USA
{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. Moreover, much less is known on how to infer cascades based on partial observations. 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 inference**

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 from 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 [4], [12].

A comprehensive understanding and analysis of cascades benefit various emerging applications in social networks [6], [16], viral marketing [1], [9], [27], and recommendation networks [24]. In order to model the propagation of information, various *cascade* models have been developed [8], [31], [33]. Among the most widely used models is the *independent cascade model* [16], 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 the entire cascade in practice, due to the noisy graphs with missing data, or data privacy policies [21], [29]. It is important to develop techniques that can *infer* the cascades using partial information. Consider the following example.

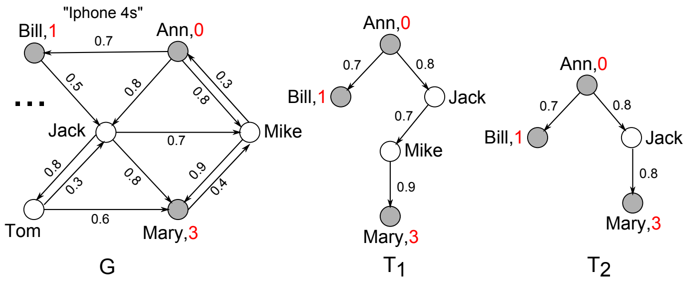


Figure 1: A cascade of an Ad (partially observed) in a social network G from user Ann, and its two possible tree representations T_1 and T_2.

Example 1: The graph G in Fig. 1 depicts a fraction of a social network (*e.g.*, Twitter), where each node is a user, and each edge represents an information exchange. For example, edge (Ann, Bill) with a weight 0.7 represents that a user Ann sends an advertisement (Ad) about a released product (*e.g.*, "Iphone 4s") with probability 0.7. To identify the impact of an Ad strategy, a company would like to know the complete cascade starting from their agent Ann. Due to data privacy policies, the observed information may be limited: (a) at time step 0, Ann posts an Ad about "Iphone 4s"; (b) **at** time step 1, Bill is **influenced** by Ann and retweets the Ad; (c) **by** time step 3, the Ad reaches Mary, and Mary retweets it. As seen, the information diffuses from one user to his or her neighbors with different probabilities, represented by the weighted edges in G. Note that the cascade unfolds as a **tree**, rooted at the node Ann.

To capture the entire topological information of the cascades, we need to make inferences in the graph-time domain. Given the above partially observed information, two such inferred cascades are shown as trees T_1 and T_2 in Fig. 1. T_1 illustrates a cascade where each path from the source Ann to each observed node has a length that exactly equals to the time step, at which the observed node is influenced, while T_2 illustrates a cascade where any path in T_2 from Ann to an observed node has a length no greater than the observed time step when the node is influenced, due to possible delay in observation, *e.g.*, Mary is known to be influenced by (instead of exactly at) time step 3. The inferred cascades provide useful information about the missing links and users that are important in the propagation of the information.

The above example highlights the need to make reasonable inference about the cascades, according to only the partial observations of influenced nodes and the time at or by which they are influenced. 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, including complexity bounds and approximation algorithms.

Contributions. We investigate the cascade inference problem, where cascades follow the widely used *independent cascade model*. To the best of our knowledge, this is the first work towards inferring cascades as *general trees* following independent cascade model, based on the partial observations.

(a) We introduce the notions of *(perfect and bounded) consistent trees* in Section II. These notions capture the inferred cascades by incorporating connectivity and time constraints in the partial observations. To provide a quantitative measure of the quality of inferred cascades, we also introduce two metrics in Section II, based on (i) the size of the consistent trees, and (ii) the likelihood when a diffusion function of the network graph is taken into account, respectively. These metrics give rise to two optimization problems, referred to as the *minimum consistent tree* problem and *minimum weighted consistent tree* problem.

(b) We investigate the problems of identifying perfect and bounded consistent trees, for given partial observations, in Section III and Section IV, respectively. These problems are variants of the inference problem.

(i) We show that these problems are all NP-complete. Worse still, the optimization problems are hard to approximate: unless P = NP, it is not possible to approximate the problems within any *constant* ratio.

(ii) Nevertheless, we provide approximation and heuristic algorithms for these problems. For bounded trees, the problems are $O(|X| * \frac{\log f_{min}}{\log f_{max}})$-approximable, where $|X|$ is the size of the partial observation, and f_{min} (resp. f_{max}) are the minimum (resp. maximum) probability on the graph edges. We provide such polynomial approximation algorithms. For perfect trees, we show that it is already NP-hard to even find a feasible solution. However, we provide an efficient heuristics using a greedy strategy. Finally, we address a practical special case for perfect tree problems, which are $O(d * \frac{\log f_{min}}{\log f_{max}})$-approximable, where d is the diameter of the graph, which is typically small in practice.

(c) We experimentally verify the effectiveness and the efficiency of our algorithms in Section V, using real-life data and synthetic data. We show that our inference algorithms can efficiently infer cascades with satisfactory accuracy.

Related work. We categorize related work as follows.

Cascade Models. To capture the behavior of cascades, a variety of cascade models have been proposed [2], [13], [15], [17], [18], such as *Suscepctible/Infected (SI) model* [2], *decreasing cascade model* [17], *triggering model* [16], *Shortest Path Model* [19], and the *Susceptible/Infected/Recover (SIR) model* [18]. In this paper, we assume that the cascades follow the *independent cascade model* [13], which is one of the most widely studied models (the shortest path model [19] is one of its special cases).

Cascade Prediction. There has been recent work on cascade prediction and inference, with the emphasis on global properties (*e.g.*, cascade nodes, width, size) [5], [11], [20], [23], [29], [31], [33] with the assumption of missing data and partial observations. The problem of identifying and ranking influenced nodes is addressed in [20], [23], but the topological inference of the cascades is not considered. Wang et al. [33] proposed a *diffusive logistic* model to capture the evolution of the density of active users at a given distance over time, and demonstrated the prediction ability of this model. Nevertheless, the structural information about the cascade is not addressed. Song et al. [31] studied the probability of a user being influenced by a given source. In contrast, we consider a more general inference problem where there are multiple observed users, who are influenced at different time steps from the source. Fei et al. [11] studied social behavior prediction and the effect of information content. In particular, their goal is to predict actions on an article based on the training dataset. Budak et al. [5] investigated the optimization problem of minimizing the number of the possible influencing nodes following a specified cascade model, instead of predicting cascades based on partial observations.

All the above works focus on predicting the nodes and their behavior in the cascades. In contrast, we propose approaches to infer both the nodes and the topology of the cascades in the graph-time domain.

Network Inference. Another host of work study network inference problem, which focuses on inferring network structures from observed cascades over the unknown network, instead of inferring cascade structures as trees [10], [14]. Manuel et al. [14] proposes techniques to infer the structure of a network where the cascades flow, based on the observation over the time each node is affected by a cascade. Similar network inference problem is addressed in [10], where the cascades are modeled as (Markov random walk) networks. The main difference between our work and theirs is (a) we use consistent trees to describe possible cascades allowing partial observations; (b) we focus on inferring the structure of cascades as trees instead of the backbone networks.

Closer to our work is the work by Sadikov et al. [29] that consider the prediction of the cascades modeled as k-trees, a

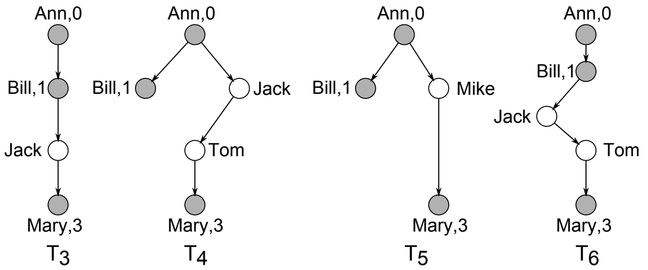


Figure 2: Tree representations of a partial observation $X = \{(\mathsf{Ann}, 0), (\mathsf{Bill}, 1), (\mathsf{Mary}, 3)\}$: T_3, T_4 and T_5 are consistent Trees, while T_6 is not.

balanced tree model. The global properties of cascades such as size and depth are predicted based on the incomplete cascade. In contrast to their work, (a) we model cascades as general trees instead of k-balanced trees, (b) while Sadikov et al. [29] assume the partial cascade is also a k-tree and predict only the properties of the original cascade, we 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. (c) The temporal information (*e.g.,* time steps) in the partial observations is not considered in [29].

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* $f : E \rightarrow R^+$ 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* [16]. We say an information propagates over a graph G 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) $\mathcal{T}$ is a function which 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 specified source node s to a set of influenced nodes.

Indeed, one may verify that any cascade from s following the independent cascade model is a tree rooted at s.

Example 2: The graph G in Fig. 1 depicts a social graph. The tree T_1 and T_2 are two possible cascades following the independent cascade model. For instance, after issuing an ad of "Iphone 4s", Ann at time 0 becomes "newly active". Bill and Jack retweet the ad at time 1. Ann becomes "active", while Bill and Jack are turned to "newly active". The process repeats until the ad reaches Mary at time step 3. The trace of the information propagation forms the cascade T_1.

As remarked earlier, it is often difficult to observe the entire structure of a cascade in practice. We model the observed information for a cascade as a *partial observation*.

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 easily 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 *equals to* 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 [22]; (b) the 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, t) indicates that node v is newly active at the time step $t' \leq t$, due to possible *delays* in the information propagation, as observed in [6].

Example 3: Recall the graph G in Fig. 1. The partial observation of a cascade in G is $X = \{(\mathsf{Ann}, 0), (\mathsf{Bill}, 1), (\mathsf{Mary}, 3)\}$. The tree T_1 is a perfect consistent tree *w.r.t.* X, where T_2 is a bounded consistent tree *w.r.t.* X.

Now consider the trees in Fig. 2. One may verify that (a) T_3, T_4 and T_5 are bounded consistent trees *w.r.t.* X; (b) T_3 and T_4 are perfect consistent trees *w.r.t.* X, where T_5 is not a perfect consistent tree. (c) T_6 is not a consistent tree, as there is no path from the source Ann to Mary with length no greater than 3 as constrained by the observation point $(\mathsf{Mary}, 3)$.

B. Cascade inference problem

We 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 the consistent trees that are most likely to be the real 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:

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

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 of the maximum likelihood in G *w.r.t.* X. Using log-likelihood, the *minimum weighted consistent tree* problem is to identify the consistent tree T_s with the minimum $-L_X(T_s)$, which in turn has the maximum likelihood.

Minimum consistent trees. Instead of weighted consistent trees, one may simply want to find the *minimum* structure that represents a cascade [25]. 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, the metric favors those consistent trees consist with the given partial observation 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 the related optimization problems using the two metrics. We investigate the problems for perfect consistent trees in Section III, and for bounded consistent trees in Section IV, respectively.

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 $\mathsf{PCT_{min}}$ (resp. $\mathsf{PCT_w}$) is to find the perfect consistent trees with minimum size (resp. weight) as the quality metric.

Though it is desirable to have efficient polynomial time algorithms to identify perfect consistent trees, the problems of searching $\mathsf{PCT_{min}}$ and $\mathsf{PCT_w}$ 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 [32], which is to determine whether there is a simple path of length $|V|-1$ in a graph G $=(V, E)$. Following this, one can verify that the $\mathsf{PCT_{min}}$ and $\mathsf{PCT_w}$ problems are NP-complete as an immediate result.

Proposition 1(b) shows that the $\mathsf{PCT_{min}}$ and $\mathsf{PCT_w}$ problems are hard to approximate. The APX class [32] consists of NP optimization problems that can be approximated by a polynomial time (PTIME) algorithm within *some* positive constant. The APX-hard problems are APX problems to which every APX problem can be reduced. Hence, the problem for computing a minimum (weighted) perfect consistent tree is among the hardest ones that allow PTIME algorithms with a constant approximation ratio.

It is known that if there is an *approximation preserving reduction (*AFP-reduction*)* [32] from a problem Π_1 to a problem Π_2, and if problem Π_1 is APX-hard, then Π_2 is APX-hard [32]. To see Proposition 1(b), we may construct an AFP-reduction from the minimum directed steiner tree (MST) problem. An instance of a directed steiner tree problem $I = \{G, V_r, V_s, r, w\}$ consists of a graph G, a set of *required* nodes V_r, a set of *steiner* nodes V_s, a source node r and a function w which assigns to each node a positive weight. The problem is to find a minimum weighted tree rooted at r, such that it contains all the nodes in V_r and a part of V_s. We show such a reduction exists. Since MST is APX-hard, $\mathsf{PCT_{min}}$ is APX-hard.

A. Bottom-up searching algorithm

Given the above intractability and approximation hardness result, we introduce a heuristic WPCT for the $\mathsf{PCT_w}$ 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

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. $T := T \cup \mathsf{PCT_l}(G_t, V_1 \cup V_2, V_3, i)$;
16. **if** T is a tree **then return** T;
17. **return** $\emptyset$;

Procedure $\mathsf{PCT_l}$
Input: A bipartite graph G_t,
node set V, node set V_s, a number t_i;
Output: a forest T_t.

1. $T_t = \emptyset$;
2. construct T_t as a minimum weighted steiner forest
   which cover V as the required nodes;
3. **for each** tree $T_i \in T_t$ **do**
4. $l(r) := t_i - 1$ where $r \in V_s$ is the root of T_i;
5. **return** T_t;

Figure 3: Algorithm WPCT: initialization, pruning and local searching

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: if for a node v' and each observed node v in a cascade with time step t, $\mathsf{dist}(s, v') + \mathsf{dist}(v', v) > t$, then v' and all the edges connected to v' can be removed from G_b.

Algorithm. Algorithm WPCT, as shown in Fig. 3, consists of the following steps:

Initialization (line 1). The algorithm WPCT starts by initializing a tree T, 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.

Pruning (lines 2-10). The algorithm WPCT 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). 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 with V_b and E_b at line 9. The partial observation X is also sorted *w.r.t.* the time step (line 10).

Bottom-up local searching (lines 11-17). Following a bottom-up greedy strategy, the algorithm WPCT processes each observation point as follows. 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 the observation points 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 union of the parents for the nodes in V_1 and V_2. (b) It constructs an edge set E_t which consists of the edges from the nodes in V_3 to the nodes in 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 $\mathsf{PCT_l}$ 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 (line 16). Otherwise, it returns $\emptyset$ (line 17). The above procedure is as illustrated in Fig. 4.

Procedure $\mathsf{PCT_l}$. Given a (bipartite) graph G_t, and two sets of nodes V and V_s in G_t, the procedure $\mathsf{PCT_l}$ computes for G_t a set of trees $T_t = \{T_1, \ldots, T_i\}$ with the minimum total weight (line 2), such that (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 contain all the nodes in V as leaves. For each T_i, $\mathsf{PCT_l}$ assigns its root r in V_s a level $l(r) = t_i - 1$ (line 4). T_t is then returned as a part of the entire perfect consistent tree (line 5). In practice, we may either employ linear programming, or an algorithm for MST problem (*e.g.*, [28]) to compute T_t.

Example 4: The cascade T_1 in Fig. 1, as a minimum weighted perfect consistent tree, can be inferred by algorithm WPCT as illustrated in Fig. 4. WPCT first initializes a tree T with the node Mary. It then constructs G_t as

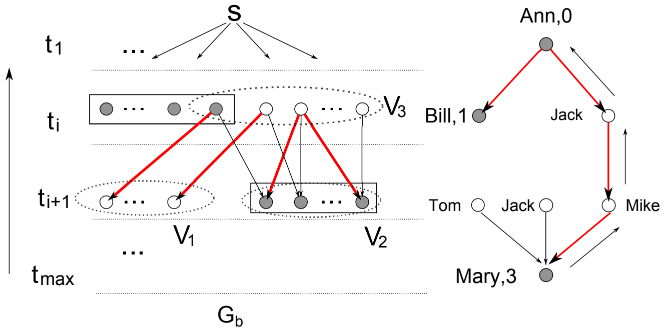


Figure 4: The bottom-up searching in the backbone network

the graph induced by edges (Tom, Mary), (Jack, Mary), and (Mike, Mary). Intuitively, the three nodes as the parents of Mary are the possible nodes which accepts the message at time step 2. It then selects the tree with the maximum probability, which is a single edge (Mike, Mary), and adds it to T. Following Mike, it keeps choosing the optimal tree structure for each level, and identifies nodes Jack. The process repeats until WPCT reaches the source Ann. It then returns the perfect consistent tree T as the inferred cascade from the partial observation X.

Correctness. 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.

Complexity. The algorithm WPCT is 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 PCT_l. Indeed, (a) the initialization and preprocessing phase (lines 1-9) takes $O(|V||E|)$ time, (b) the sorting phase is in $O(|X|^2)$ time, (c) the bottom-up construction is in $O(|t_{max} * \mathcal{A}|)$, which is further bounded by $O(|t_{max} * |V|^3)$ if an approximable algorithm is used [28]. In our experimental study, we utilize efficient linear programming to compute the *optimal* steiner forest.

The algorithm WPCT can easily be adapted to the problem of finding the minimum perfect consistent trees, where each edge has a unit weight.

Perfect consistent SP trees. The independent cascade model may be an overkill for real-life applications, as observed in [7], [19]. Instead, one may identify the consistent trees which follow the shortest path model [19], where cascades propagate following the 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 = dist(s, v); in other words, the path from s to v in the tree is the *shortest path* in G. The PCT_w (resp. 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 PCT_min and PCT_w problems for perfect sp trees are NP-hard and APX-hard; (c) the 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) probability by the diffusion function f.

We next provide an approximation algorithm to the PCT_w problem for sp trees. Given a graph G and a partial observation X, the algorithm, denoted as 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, WPCT_sp approximately computes an undirected minimum steiner tree T. If the directed counterpart T' of T in G_b is not a tree, 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 the shortest path, and (b) removes the redundant edges attached to v. It then returns T' as an sp tree.

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 [32]. Moreover, the algorithm WPCT_sp can be used for the problem 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 $(v, t) \in X$, dist$(s, v) \leq t$, where dist(s, v) is the distance from s to v in G.

Indeed, one may verify the following: (a) if there is a node $(v_i, t_i) \in X$ where dist$(s, v_i) > t_i$, there is no path satisfies the time constraint and T is empty; (b) if dist$(s, v_i) \leq t_i$ for each node $(v_i, t_i) \in X$, a BFS tree rooted at s with each node v_i in X as its internal node or leaf is a bounded consistent tree. Thus, to determine whether there is a bounded consistent tree is in $O(|E|)$ time, via a BFS traversal of G from s.

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

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 5: Algorithm WBCT: searching bounded consistent trees via top-down strategy

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) probability by the diffusion function f over G.

We can prove Theorem 1(a) as follows. First, the $\mathsf{BCT_w}$ problem, as a decision problem, is to determine whether there exists a bounded consistent tree T with $-L_X(T)$ no greater than a given bound B. The problem is obviously in NP. To show the lower bound, one may show there exists a polynomial time reduction from the exact 3-cover problem (X3C). Second, to see the approximation hardness, one may verify that there exists an AFP-reduction from the minimum directed steiner tree (MST) problem.

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

Algorithm. The algorithm WBCT is illustrated in Fig. 5. 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) [3] 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 $\emptyset$ as the tree T (line 5). Otherwise, the tree T is returned (line 8) after all the observation points in X are processed.

Correctness and complexity. One may verify that algorithm WBCT either correctly computes a bounded consistent tree T, or returns $\emptyset$. For each node in the observation point X, there is a path of weight selected using a greedy strategy, and the top-down strategy guarantees that the paths form a consistent tree. 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. We have considered the likelihood function as a quantitative metric for the quality of the bounded consistent trees. As remarked earlier, one may simply want to identify the bounded consistent trees of the *minimum* size. Given a social graph G and a partial observation X, the *minimum bounded consistent tree problem*, denoted as $\mathsf{BCT_{min}}$, is to identify the bounded consistent tree with the minimum size, *i.e.*, the total number of nodes and edges. The $\mathsf{BCT_{min}}$ problem is a special case of $\mathsf{BCT_w}$, and its main result is summarized as follows.

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

Proposition 4(a) and 4(b) can both be shown by constructing reductions from the MST problem, which is NP-complete and APX-complete [32].

Despite of the hardness, the problem can be approximated within $O(|X|)$ in polynomial time, by applying the algorithm WBCT over an instance where each edge has a unit weight. This completes the proof of Proposition 4(c).

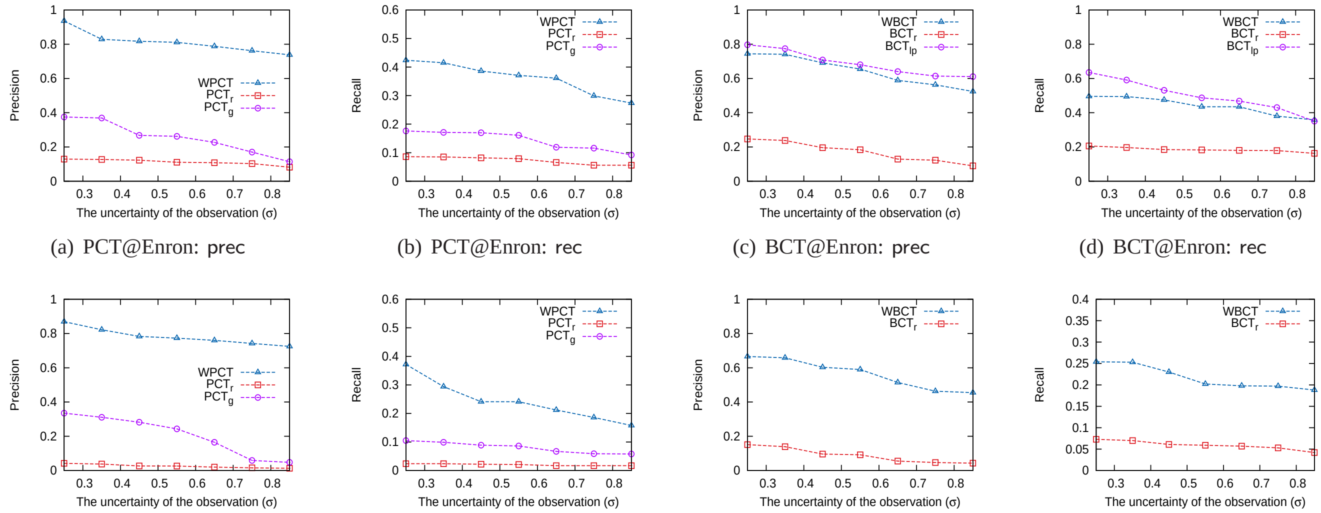
V. EXPERIMENTS

We next present an experimental study of our proposed methods. Using both real-life and synthetic data, we conduct three sets of experiments to evaluate (a) the effectiveness of the proposed algorithms, (b) the efficiency and the scalability of WPCT and WBCT.

Experimental setting. We used real-life data to evaluate the effectiveness of our methods, and synthetic data to conduct an in-depth analysis on scalability by varying the parameters of cascades and partial observations.

(a) Real-life graphs and cascades. We used the following real-life datasets. (i) *Enron email cascades*. The dataset of *Enron Emails* [1] consists of a social graph of $86,808$ nodes and $660,642$ edges, where a node is a user, and two nodes are connected if there is an email message between them. We tracked the *forwarded* messages of the same subjects and obtained 260 cascades of depth no less than 3 with more than 8 nodes. (ii) *Retweet cascades* (RT). The dataset of *Twitter Tweets* [2] [35] contains more than 470 million posts

[1]http://www.cs.cmu.edu/ enron/
[2]http://snap.stanford.edu/data/twitter7.html



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

(e) PCT@RT: prec (f) PCT@RT: rec (g) BCT@RT: prec (h) BCT@RT: rec

Figure 6: The prec and rec of the inference algorithms over Enron email cascades and Retweet cascades

from more than 17 million users, covering a period of 7 months from June 2009. We extracted the retweet cascades of the identified *hashtags* [35]. To guarantee that a cascade represents the propagation of a single hashtag, we removed those retweet cascades containing multiple hashtags. In the end, we obtain 321 cascades of depth more than 4, with node size ranging from 10 to 81. Moreover, we used the EM algorithm from [30] to estimate the diffusion function.

(b) Synthetic cascades. We generated a set of synthetic cascades unfolding in an anonymous Facebook social graph [3], which exhibits properties such as power-law degree distribution, high clustering coefficient and positive assortativity [34]. The diffusion function is constructed by randomly assigning real numbers between 0 and 1 to edges in the network. The generating process is controlled by size $|T|$. We randomly choose a node as the source of the cascade. By simulating the diffusion process following the independent cascade model, we then generated cascades *w.r.t.* $|T|$ and assigned time steps.

(c) Partial observation. For both real life and synthetic cascades, we define *uncertainty* of a cascade T as $\sigma = 1 - \frac{|X|}{|V_T|}$, where $|V_T|$ is the size of the nodes 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.

(d) Implementation. We have implemented the following in C++: (i) algorithms WPCT, and WBCT; (ii) two linear programming algorithms $\mathsf{PCT}_{\mathsf{lp}}$ and $\mathsf{BCT}_{\mathsf{lp}}$, which identify the optimal weighted bounded consistent trees and the optimal perfect consistent trees using linear programming, respectively; (iii) two randomized algorithms $\mathsf{PCT}_{\mathsf{r}}$ and $\mathsf{BCT}_{\mathsf{r}}$, which are developed to randomly choose trees

Algorithms	Precision	Enron		Twitter	
		d=3	d=4	d = 4	d = 5
WPCT	prec_v	100%	100%	97.2%	93.2%
	prec_e	78.2%	82.4%	86.1%	82.6%
WBCT	prec_v	100%	70.1%	73.6%	66.1%
	prec_e	69%	55.7%	60.6%	41.7%

Table I. prec_v and prec_e over real cascades

from given graphs. $\mathsf{PCT}_{\mathsf{r}}$ is developed using a similar strategy for WPCT, especially for each level the steiner forest is randomly selected (see Section III); as WBCT does, $\mathsf{BCT}_{\mathsf{r}}$ runs on bounded BFS directed acyclic graphs, but randomly selects edges. (iv) to verify various implementations of WPCT, an algorithm $\mathsf{PCT}_{\mathsf{g}}$ is developed by using a greedy strategy to choose the steiner forest for each level (see Section III). We used LP_solve 5.5 [4] as the linear programming solver.

We used a machine powered by an Intel(R) Core 2.8GHz CPU and 8GB of RAM, using Ubuntu 10.10. Each experiment was run by 10 times and the average is reported here.

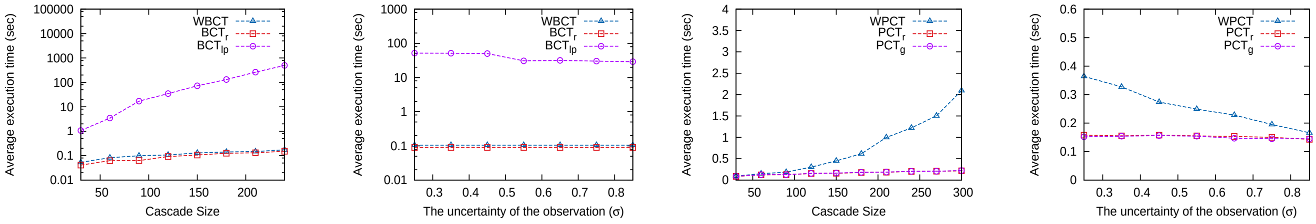
Experimental results. We next present our findings.

Effectiveness of consistent trees. In the first set of experiments, using real life cascades, we investigated the accuracy and the efficiency of our cascade inference algorithms.

(a) 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 $\mathsf{rec} = \frac{\Sigma(|(V_{T_i'} \cap V_{T_i}) \setminus V_X|)}{\Sigma(|V_{T_i} \setminus V_X|)}$. Intuitively, prec is the fraction of inferred nodes that are missing from T_i, while rec is the fraction of missing nodes that are inferred by T_i'.

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

[4]http://lpsolve.sourceforge.net/5.5/



(a) Varying $|T|$ (b) Varying σ (c) Varying $|T|$ (d) Varying σ

Figure 7: Efficiency and scalability over synthetic cascades

For Enron email cascades, Fig. 6(a) and Fig. 6(b) show the accuracy of WPCT, $\mathsf{PCT_g}$ and $\mathsf{PCT_r}$ for inferring cascades, while σ is varied from 0.25 to 0.85. $\mathsf{PCT_{lp}}$ does not scale over the Enron dataset and thus is not shown. (i) WPCT outperforms $\mathsf{PCT_g}$ and $\mathsf{PCT_r}$ on both prec and rec. (ii) 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, the performance of WBCT, $\mathsf{BCT_{lp}}$ and $\mathsf{BCT_r}$ are shown in Fig. 6(c) and Fig. 6(d), respectively. (i) Both $\mathsf{BCT_{lp}}$ and WBCT outperform $\mathsf{BCT_r}$, and their prec and rec decrease while the uncertainty increases. (ii) $\mathsf{BCT_{lp}}$ has better performance than WBCT. In particular, both $\mathsf{BCT_{lp}}$ and WBCT successfully infer the cascade nodes with the prec no less than 50% and with the rec no less than 25%, even when 85% of the nodes in the cascades are removed.

For retweet cascades, the prec and the rec of WPCT, $\mathsf{PCT_g}$ and $\mathsf{PCT_r}$ are shown in Fig. 6(e) and in Fig. 6(f), respectively. While the uncertainty increases from 0.25 to 0.85, (i) WPCT outperform $\mathsf{PCT_r}$ and $\mathsf{PCT_g}$, and (ii) the performance of all the algorithms decreases. In particular, WPCT successfully infers the nodes with the prec more than 80% and the rec more than 35%, while the uncertainty is 25%. Similarly, the prec and the rec of WBCT and $\mathsf{BCT_r}$ are presented in Fig. 6(g) and Fig. 6(h), respectively. As $\mathsf{BCT_{lp}}$ does not scale on retweet cascades, its performance is not shown. While the uncertainty σ increases, the prec and the rec of the algorithms decrease. For all σ, WBCT outperforms $\mathsf{BCT_r}$; in particular, WBCT correctly infers the nodes with prec no less than 60% and rec no less than 25%, when σ is 25%.

(b) To further evaluate the structural similarity of T_i and $T_i{}'$ as described in (a), we also evaluate (i) $\mathsf{prec}_v = \frac{|V''|}{|V'|}$ for nodes $V' = (V_{T_i{}'} \cap V_{T_i}) \setminus V_X$, where $V'' \in V'$ are the nodes with the same *topological order* in both T_i' and T_i, and (ii) $\mathsf{prec}_e = \frac{|E'|}{|E_{T_i{}'}|}$ for $E' = E_{T_i} \cap E_{T_i{}'}$, following the metric for measuring graph similarity [26]. The average results are as shown in Table I, for σ =50%, and the cascades of fixed depth. As shown in the table, for WPCT, the average prec_v is above 90%, and the average prec_e is above 75% over both datasets. Better still, the results hold even when we set $\sigma = 85\%$. For WBCT, prec_v and prec_e are above 65% and above 40%, respectively. For WPCT, prec_v and prec_e have almost consistent performance on both datasets; however, for WBCT, the prec_v and prec_e of the inferred Enron cascades are higher than those of the inferred retweet cascades. The gap might result from the different diffusion patterns between these two datasets: we observed that there are more than 70% of cascades in the Enron dataset whose structures are contained in the BFS directed acyclic graphs of WBCT, while in the Twitter Tweets there are less than 45% of retweet cascades following the assumed graph structures of WBCT.

Efficiency over real datasets. In all the tests over real datasets, $\mathsf{PCT_r}$, $\mathsf{BCT_r}$, $\mathsf{PCT_g}$ and WBCT take less than 1 second. $\mathsf{BCT_{lp}}$ does not scale for retweet cascades, while $\mathsf{PCT_{lp}}$ does not scale for both datasets. On the other hand, while WPCT takes less than 0.4 seconds in inferring all the Enron cascades, it takes less than 20 seconds to infer Twitter cascades where d=4, and 100 seconds when $d = 5$. Indeed, for Twitter network the average degree of the nodes is 20, while the average degree for Enron dataset is 7. As such, it takes more time for WPCT to infer Twitter cascades in the denser Twitter network. In our tests, the efficiency of all the algorithms are not sensitive *w.r.t.* the changes to σ.

Efficiency and scalability over synthetic datasets. In the second set of experiments, we evaluated the efficiency and the scalability of our algorithms using synthetic cascades.

(a) We first evaluate the efficiency and scalability of WPCT and compare WPCT with $\mathsf{PCT_r}$ and $\mathsf{PCT_g}$.

Fixing uncertainty $\sigma = 50\%$, we varied $|T|$ from 30 to 240. Fig. 7(c) shows that WPCT scales well with the size of the cascade. Indeed, it only takes 2 seconds to infer the cascades with 300 nodes.

Fixing size $|T| = 100$, we varied the uncertainty σ from 0.25 to 0.85. Fig. 7(d) illustrates that while all the three algorithms are more efficient with larger σ, WPCT is more sensitive. All the three algorithms scale well with σ.

As $\mathsf{PCT_{lp}}$ does not scale well, its performance is not shown in Fig. 7(c) and Fig. 7(d).

(b) Using the same setting, we evaluated the performance of WBCT, compared with $\mathsf{BCT_{lp}}$ and $\mathsf{BCT_r}$.

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	$O\lvert V^3\rvert$
$\mathsf{PCT}_{\mathsf{w}}$ (sp tree)	NP-c, APX-hard	$d * \frac{\log f_{max}}{\log f_{min}}$	$O\lvert V^3\rvert$
$\mathsf{PCT}_{\mathsf{min}}$	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 II. Summary: complexity and approximability

Fixing σ and varying $|T|$, the result is reported in Fig. 7(a). First, WBCT outperforms $\mathsf{BCT}_{\mathsf{lp}}$, and is almost as efficient as the randomized algorithm $\mathsf{BCT}_{\mathsf{r}}$. For the cascade of 240 nodes, WBCT takes less than 0.5 second to infer the structure, while $\mathsf{BCT}_{\mathsf{lp}}$ takes nearly 1000 seconds. Second, while WBCT is not sensitive to the change of $|T|$, $\mathsf{BCT}_{\mathsf{lp}}$ is much more sensitive.

Fixing $|T|$ and varying σ, Fig. 7(b) shows the performance of the three algorithms. The figure tells us that WBCT and $\mathsf{BCT}_{\mathsf{r}}$ are less sensitive to the change of σ than $\mathsf{BCT}_{\mathsf{lp}}$. This is because WBCT and $\mathsf{BCT}_{\mathsf{r}}$ identify bounded consistent tree by constructing shortest paths from the source to the observed nodes. When the maximum depth of the observation point is fixed, the total number of nodes and edges visited by WBCT and $\mathsf{BCT}_{\mathsf{r}}$ are not sensitive to σ.

Summary. We can summarize the results as follows. (a) Our inference algorithms can infer cascades effectively. For example, the original cascades and the ones inferred by WPCT have structural similarity (measured by prec_e) of higher than 75% in both real-life datasets. (b) Our algorithms scale well with the sizes of the cascades, and uncertainty. They seldom demonstrated their worst-case complexity. For example, even for cascades with 240 nodes, all of our algorithms take less than two seconds.

VI. Conclusion

In this paper, we investigated cascade inference problem based on partial observation. We proposed the notions of consistent trees for capturing the inferred cascades, namely, bounded consistent trees and perfect consistent trees, as well as quantitative metrics by minimizing either the size of the inferred structure or maximizing the overall likelihood. We have established the intractability and the hardness results for the optimization problems as summarized in Table II. Despite the hardness, we developed approximation and heuristic algorithms for these problems, with performance guarantees on inference quality, We verified the effectiveness and efficiency of our techniques using real life and synthetic cascades. Our experimental results have shown that our methods are able to efficiently and effectively infer the structure of information cascades.

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] N. Bailey. *The mathematical theory of infectious disease and its applications*. 1975.

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

[4] 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.

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

[6] 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.

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

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

[9] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, 2001.

[10] M. Eslami, H. R. Rabiee, and M. Salehi. Dne: A method for extracting cascaded diffusion networks from social networks. In *The Third IEEE International Conference on Social Computing*, 2011.

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

[12] 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.

[13] 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, 2001.

[14] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. *TKDD*, 5(4):21, 2012.

[15] M. Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6), 1978.

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

[17] D. Kempe, J. Kleinberg, and E. Tardos. Influential nodes in a diffusion model for social networks. In *ICALP*, 2005.

[18] W. O. Kermack and A. G. Mckendrick. A contribution to the mathematical theory of epidemics. *Proc R Soc Lond A*, 115:700–721, 1927.

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

[20] M. Kimura, K. Saito, R. Nakano, and H. Motoda. Finding influential nodes in a social network from information diffusion data. *Social Computing and Behavioral Modeling*, pages 1–8, 2009.

[21] G. Kossinets. Effects of missing data in social networks. *Social networks*, 28(3), 2006.

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

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

[24] 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.

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

[26] J. W. Raymond, E. J. Gardiner, and P. Willett. Rascal: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45:2002, 2002.

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

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

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

[30] K. Saito, R. Nakano, and M. Kimura. Prediction of informa-

tion diffusion probabilities for independent cascade model. In *KES*, 2008.
[31] X. Song, Y. Chi, K. Hino, and B. L. Tseng. Information flow modeling based on diffusion rate for prediction and ranking. In *WWW*, 2007.
[32] V. V. Vazirani. *Approximation Algorithms*. 2001.
[33] F. Wang, H. Wang, and K. Xu. Diffusive Logistic Model Towards Predicting Information Diffusion in Online Social Networks. *ArXiv e-prints*, 2011.
[34] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. In *EuroSys*, 2009.
[35] 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.