Interestingness-Driven Diffusion Process Summarization in Dynamic Networks

Qiang Qu[1], Siyuan Liu[2], Christian S. Jensen[3], Feida Zhu[4], and Christos Faloutsos[5]

[1] Department of Computer Science, Aarhus University
[2] Heinz College, Carnegie Mellon University
[3] Department of Computer Science, Aalborg University
[4] School of Information Systems, Singapore Management University
[5] School of Computer Science, Carnegie Mellon University
qu@cs.au.dk, siyuan@cmu.edu, csj@cs.aau.dk, fdzhu@smu.edu.sg, christos@cs.cmu.edu

Abstract. The widespread use of social networks enables the rapid diffusion of information, e.g., news, among users in very large communities. It is a substantial challenge to be able to observe and understand such diffusion processes, which may be modeled as networks that are both large and dynamic. A key tool in this regard is data summarization. However, few existing studies aim to summarize graphs/networks for dynamics. Dynamic networks raise new challenges not found in static settings, including time sensitivity and the needs for online interestingness evaluation and summary traceability, which render existing techniques inapplicable. We study the topic of dynamic network summarization: how to summarize dynamic networks with millions of nodes by only capturing the few most interesting nodes or edges over time, and we address the problem by finding interestingness-driven diffusion processes. Based on the concepts of diffusion radius and scope, we define interestingness measures for dynamic networks, and we propose OSNet, an online summarization framework for dynamic networks. We report on extensive experiments with both synthetic and real-life data. The study offers insight into the effectiveness and design properties of OSNet.

1 Introduction

The summarization of networks or graphs continues to be an important research problem due in part to the ever-increasing sizes of real-world networks. While most studies consider the summarization of static networks according to criteria such as compression ratio, network representation, minimum loss, and visualization friendliness [15,20], recent developments in social network mining and analysis as well as in location-based services [6, 13] and bioinformatics [20] give prominence to the study of a new kind of dynamic network [9, 10] that captures information diffusion processes in an underlying network. These developments offer new challenges to network summarization.

An information diffusion process in a network can be represented by a stream of timestamped pairs of nodes from the underlying network, where a timestamped pair indicates that information was sent from one node to the other at the given time. This stream can be modeled as a *dynamic network*. An example of an information diffusion

T. Calders et al. (Eds.): ECML PKDD 2014, Part II, LNCS 8725, pp. 597–613, 2014.

process is the spread of news items among Twitter users by means of the network's "reply/(re)tweet" functionality.

While the network that may be created from a completed diffusion process by assembling the node pairs that represent the process is a static network, the summarization task for a diffusion process distinguishes itself from that of a static network. The critical difference lies in that, for a dynamic diffusion process, it is most valuable to capture each "interesting" development as the process evolves, in an online fashion. This problem is termed *dynamic network summarization* (DNS) and has many applications. We highlight several as follows.

In information visualization, massive dynamic networks are hard to visualize due to their size and evolution [14]. With DNS, it is possible to create online, time-labeled summaries in the form of "trajectories" such that it is possible to view the change in a diffusion process as it evolves. In social graph studies, DNS enables the identification of interesting dynamics in the form of "backbones" that describe how information propagates and that can help capture the evolving roles of different participants in diffusion processes. This is useful for tasks such as change detection [18] and trend mining [4]. In road traffic analysis, DNS can capture major traffic flows. Summaries for given periods can be projected onto the road network to detect traffic thoroughfares, provide better road planning services, or analyze how people move in a city [11].

One approach is to compute a summary from the evolving diffusion process periodically. Thus, the process is represented by a sequence of summaries of static networks. Each network aggregates edges and nodes in a time interval of size $\triangle t$ [14]. However, this approach is costly when networks are large. Further, parameter $\triangle t$ is fundamentally hard to set: if it is too small, performance deteriorates, while if it is too large, important diffusion dynamics may be missed. Even if given a $\triangle t$, most of the previous methods show difficulty in producing results that capture interesting dynamics, because their specific criteria and goals do not target dynamics.

As suggested by the application examples, DNS faces unique challenges.

(1) Time Sensitivity. Diffusion processes often represent vast, viral, and unpredictable processes, e.g., breaking news and bursty events [21]. As a result, the rate of diffusion can vary drastically over a short period of time. It is a difficult challenge to respond adaptively to the changing dynamics and to achieve timely summarizations.

(2) Online Interestingness Evaluation. A key challenge is to capture the most interesting nodes and edges in summarizations. Compared with traditional network summarization, interestingness evaluation in DNS assumes an extra degree of difficulty because of the partial view of the network at any time of the evaluation.

(3) Summary Traceability. An important goal is to enable a better understanding of the evolution of the diffusion process throughout its life cycle. A good summary should reveal the flow of the dynamics so that interesting developments can be traced.

To tackle the DNS problem, we propose OSNet, a framework for Online Summarization of Dynamic Networks that aims to produce concise, interestingness-driven summaries that capture the evolution of diffusion processes. Our contribution is fivefold: 1) Unlike previous proposals that apply optimization criteria in offline settings, we consider a setting where network summarization occurs online, as a diffusion process

evolves. 2) Based on the concepts of propagation radius **proRadius** and propagation scope **proScope**, we formalize the problem of characterizing the interesting dynamics of an evolving diffusion process in a traceable manner. 3) We propose OSNet that encompasses online and incremental dynamic network summarization algorithms on a spreading tree model. In terms of entropy, OSNet archives the best summaries with respect to informativeness. 4) A generalization of OSNet is presented. 5) Extensive experiments are conducted with both synthetic and real-life datasets.

2 Problem Definition

The input to the problem is a stream of time ordered interactions (i.e., diffusion processes) on a network G. We define a network as a labeled graph $G = (V, E, l_G)$, where V is a set of nodes, $E \subset V \times V$ is a set of undirected edges, and l_G is a labeling function. Given a set $\Sigma = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_k\}$ of labels, labeling function $l_G : V(G) \mapsto \Sigma$ maps nodes to labels.

A diffusion process on a network G, denoted by $\mathcal{D}(G)$, is a stream of time-ordered interactions. An interaction $x = (\delta, u, v, t) \in \mathcal{D}(G)$ indicates that a specific story is diffused from node u to node v at time $t \in \mathcal{T}$. A story is defined by a textual keyword list used to describe an event, such as breaking news in Twitter. The diffusion from u to v captures that node v receives the story from u. We also say that u is an infector of v while v is an infectee of u. We call time t the infection time of node v. Note that a diffusion process of a story can be initiated by different nodes that are regarded as seeds or roots. For each interaction x, we further define δ to be a three-tuple as a canonical identifier, i.e., $\delta = (storyID, v_r, t')$, where $storyID$ is the identity of the diffusing story, v_r represents the seed node starting the diffusion, and t' is the infection time of the infector u. The diffusion process from a seed over a time period forms a time-stamped graph, known as a network cascade C [9, 18] where each interaction is a directed edge from the infector to the infectee.

Definition 1. [Cascade C] *A cascade C is a directed graph $C = (V_C, E_C, l_{V_C}, l_{E_C})$, representing a diffusion process $\mathcal{D}(G) = \{x = (\delta, u_i, v_i, t_i)\}$ diffusing from a seed on a story during a time period $\mathcal{T}$. The node set is $V_C = \cup u_i + \cup v_i$, and the edge set is $E_C = \cup_{u_i, v_i \in x}(u_i, v_i)$. A node pair (u_i, v_i) for each x is considered as a directed edge from u_i to v_i. $l_{V_C} : V_C \mapsto \Sigma$ is node labeling function, and $l_{E_C} : E_C \mapsto \mathcal{T}$ is an edge labeling function.*

A network G with diffusion processes is termed a diffusion network or a dynamic network, which, for simplicity, we also denote by G. Given a diffusion network G, a set $\mathcal{I}(G) \in V(G)$ is given that contains the seed nodes from which a diffusion starts. The infection time of a seed v_r is given as t_{v_r}. We use $\deg^+(u)$ to denote the number of infectees of a node u in a cascade C.

Before we present the definition of interestingness, two measures are introduced to evaluate nodes in a dynamical process by i) how far the information can travel (Measure 1: depth) and ii) how many infectees a node can have (Measure 2: breadth). These two measures can be used for capturing the **interestingness** of a diffusion process for three reasons : 1) The two measures agree with intuition. 2) The two measures capture the cascade, enabling reconstruction with little more information. 3) The two measures

offer a foundation for computing different properties of a cascade. In addition, we observe that other studies also suggest that the two measures can characterize diffusion processes [2, 22].

Measure 1. [Propagation Radius (proRadius)] *The propagation radius of a node v in a cascade C, denoted by $y(v)$, is the length of the path $l(v)$ from the root of C to v, $|l(v)|$. The maximum propagation radius of a node in C is the diameter of C: $d(C) = \max(y(v))$. Note that the propagation radius of the root is 0.*

Measure 2. [Propagation Scope (proScope)] *The* **proScope***, $w(v) = deg^{+}(v)$, of a node v for a cascade C is the number of infectees of v in C.*

Definition 2. [Interestingness] *We represent a node v by a vector $(y(v), w(v))$ and use Equation 1 to quantify the total interestingness of the node. As the degree distribution of many networks follows a power-law, we use a log value of the* ***proScope****:*

$$\xi(v) = \alpha \log w(v) + (1 - \alpha) y(v), \tag{1}$$

where $\alpha \in [0, 1]$ balances the two measures. We set $\log w(v) = 0$ if $w(v) = 0$. Note that cascades evolve over time as interactions arrive in the stream. We thus use $\xi_t(v)$ to denote the interestingness of a node v at time t, which is calculated using the values of ***proScope*** *and* ***proRadius*** *of v at t.*

Definition 3. [Interesting Summary $S(C)$] *Given a cascade C and a threshold τ, an interesting summary $S(C)$ is a subgraph of C satisfying that for any node $v_i \in S(C)$, $\xi_t(v_i) > \tau$ holds; for two nodes u and v in $V(S(C))$, the edge $e' = (u, v)$ exists in $S(C)$ if and only if $e = (u, v)$ exists in C. Labels of the edges and nodes in $S(C)$ retain the labels they have in C.*

Definition 4. [Traceable Interesting Summary $\mathbf{S}(C)$] *Given an interesting summary $S(C) \subset C$, a traceable interesting summary $\mathbf{S}(C)$ is a super-graph of $S(C)$, denoted $S(C) \subset \mathbf{S}(C)$. A node v_i in C is in $\mathbf{S}(C)$ if: v_i is the seed, or $\xi_t(v_i) > \tau \vee (\exists v_j \in C, (v_i \in l(v_j) \wedge \xi_t(v_j) > \tau))$.*

As some nodes are removed from an interesting summary (Definition 3), remaining interesting nodes may become disconnected. Definition 4 includes the missed nodes on the paths from the seed to the remaining interesting nodes. A traceable interesting summary thus is possible to reveal the flow of dynamics and interesting developments can be traced throughout their life cycle. To explain the evolution in a traceable interesting summary, we next introduce the concepts **diffusion rise** and **diffusion decay**, defined by the notion of acceleration intensity. In the rest of the paper, we use a summary (summaries) to indicate a traceable interestingness summary (summaries) for simplicity.

Definition 5. [Acceleration Intensity ϱ] *Given a node v_i as an infector of a node v_j in a cascade C, the acceleration intensity is defined based on the diffusion path from v_i to v_j ($l(v_i, v_j)$) in C as*

$$\varrho(l(v_i, v_j)) = \frac{\xi_{t_j}(v_j) - \xi_{t_i}(v_i)}{|t_j - t_i|}, \tag{2}$$

where t_i and t_j are the infection times of v_i and v_j, respectively.

We can now define the rise and decay of a diffusion process: When $\varrho > 0$, the propagation process from v_i to v_j is a diffusion rise process; otherwise, it is a diffusion decay process.

The goal of the DNS problem is to better understand network dynamics. A summary thus needs to be informative with respect to the original data. There are several methods to evaluate informativeness. Among these, we propose to use Entropy. A review of Shannon Entropy and details are presented in Section 3.2. Here we denote the entropy of a traceable interesting summary $\mathbf{S}(C)$ by $H(\mathbf{S}(C))$. Recall that the entropy gains when its value decreases. We thus aim to find a summary with minimal entropy to achieve the best informativeness. The problem is stated as follows:

Problem Statement (Interestingness-driven Diffusion Process Compression). *Given a diffusion network G with seed sets $\cup\mathcal{I}(G)$, stories diffuse from each seed over time. The dynamic process is represented by a stream of interactions, which forms a set of cascades $\{\ldots, C_i, \ldots\}$. The output of the problem is a set of traceable interesting summaries $\mathbf{S}(G) = \{\ldots, S_i(C_i), \ldots\}$ ($|S_i(C_i)| > 0$). The entropy ($H(\mathbf{S}(C))$) of each summary $S_i(C_i)$, which reveals diffusion rise and decay, is minimized subject to the balancing parameter $0 \leq \alpha \leq 1$ of the aggregate score and the interestingness threshold $\tau \geq 0$.*

To solve the problem, two sub-problems have to be solved: i) How to model the dynamics on the top of graphs? Is the cascade model suitable? The diffusion processes we discuss are evolving over time. And all the cascades on a node are merged. This may cause problems for the summarization because the interestingness of a node is associated with time stamps and stories as node instances. This requires to design a labeling function to distinguish the node instances, which is ineffective. ii) How to set proper values for α and τ for different diffusion processes? Given an α in the range $[0, 1]$, each connected subgraph of a cascade C over time can be a summary, which yields a hard graph decomposition problem. On the other hand, the scale of a summary mostly depends on the threshold τ. A proper value is necessary because we intend to find all interesting developments. We proceed to develop the OSNet framework that encompasses new and incremental techniques capable of continuously summarizing dynamics based on a spreading tree model in step with the evolution of diffusion processes.

3 Our Method

3.1 Spreading Tree Model

Although network cascades can model diffusion processes. several issues of dynamics challenge the effectiveness of network cascades. First, the interactions on a node are merged in cascades [9]. However, in dynamic networks, a node may become interesting only at a specific interaction, which would require extra efforts in designing labels to distinguish different interactions and cascades. Furthermore, as cascades are directed graphs, there exist backward and forward edges or even cycles. This makes a cascade hard to interpret and navigate. Second, the cascade model is a graph model. Summary search can then be regarded as subgraph search. However, graph search is usually time-consuming since it involves isomorphism checking. Third, since cascades are merged into one directed graph, the graph search space grows exponentially, which makes the

problem even harder. We propose to instead use a **Spreading Tree** model. First, spreading trees are constructed directly by interactions without any other efforts. The model distinguishes interactions and cascades by itself. Next, tree search is relatively efficient. Numerous proposals of efficient tree operations exist. Third, there are no backward and forward edges in spreading trees. The tree structure is not as complex as a cascade. The search space is proportional to the scale of the interactions.

Definition 6. [Spreading Tree T] *A spreading tree $T = (v_r, V', E', l_{V'}, l_{E'})$, is a rooted and labeled n-ary tree, where $v_r \in V'$ is the root, V' is a set of nodes, $E' \subseteq V' \times V'$ is a set of edges, $l_{V'} : V' \mapsto \Sigma$ is node labeling function, and $l_{E'} : E' \mapsto \mathcal{T}$ is an edge labeling function.*

Intuitively, a node represents a specific user in a network, and the node's label is the name of the user; an edge in a spreading tree connects an infector node with an infectee node, and the edge's label is the infection time of the infectee node. A non-root node has one infector. A non-leaf node has one or more infectees, and a leaf node has no infectees.

Given a diffusion network G, each seed $v_r \in V(G)$ forms the root of a spreading tree. When an interaction $x = (\delta, u, v, t) \in \mathcal{D}(G)$ arrives, the spreading tree for δ is updated by inserting a **new** node labeled v and an edge labeled t from an **existing** node labeled u to v. Note that both u and v are labels of the nodes. To find the existing node u, we search the tree in breadth-first order starting from the root until a node with label u and infection time $\delta.t'$ is found. Therefore, although multiple nodes have the same label, the three-tuple δ can determine from which node to insert the edge to the new infectee.

From the above, the spreading tree model achieves the following properties: 1) cascades can be equally modeled as spreading trees, such that the summarization on cascades equals the task on spreading trees; 2) the trees are separated by seeds; 3) a node can be duplicated in a spreading tree, which shows the model distinguishes node instances; 4) the size of trees is proportional to the scale of interactions; 5) infection occurs top-down, and diffusion always occurs from a parent node to a child node.

3.2 Parameter Relief

Although using fixed values for parameters is simple for implementation, two main issues demand better approaches. First, for a single diffusion process, prediction of the network statistics (arrival rate, number of infectees, propagating range, etc.) is usually difficult. Thus, it is hard to find parameter settings that can best capture the dynamics. Second, different diffusion processes vary substantially in range and scope. Thus, the same settings are not likely to work across different processes. Our study aims to provide a self-tuning mechanism that adapts to differences in the summarization.

Alpha Estimation. Recall that the entropy H of a random variable E with possible values $\{e_1, \ldots, e_n\}$ is defined as

$$H(E) = -\sum_{i}^{n} p(e_i) log_2 p(e_i), \tag{3}$$

where $p(e_i)$ is the probability mass function of outcome e_i. $H(E)$ is close to 0 if the distribution is highly skewed and informative.

We measure the entropy of a summary $\mathbf{S}(C)$ and aim to maximize the informativeness of $\mathbf{S}(C)$ to have the maximum possible information out of T. Given a set of continuous interactions $\mathcal{D}(G)$ by time t (denoted by $\mathcal{D}(G)_t$), the probability of diffusing story i from a node v_j is regarded as a conditional probability:

$$p_{(i,t)}(v_j) = p_{(i,t)}(v_i) \times \frac{\sum^{\mathcal{D}(G)_t} f_{(i,t)}(v_j, x)}{|\mathcal{D}(G)_t|},$$

where $p_{(i,t)}(v_i)$ is the probability of the node that infects v_j. Note that for a seed node, the probability of its infector is 1 in order to guarantee that a root is infected. The function $f_{(i,t)}(v_j, x)$ is an indicator that is 1 if v_j is an infectee in x when $storyID = i$ by time t, otherwise 0. Then we use the entropy $H_{p(i,t)}(\mathbf{S}(C))$ as an informativeness measure of a summary $\mathbf{S}(C)$ with respect to T:

$$H_{p(i,t)}(\mathbf{S}(C)) = -\sum_{j=1}^{|\mathbf{S}(C)|} p_{(i,t)}(v_j) \log p_{(i,t)}(v_j). \tag{4}$$

Thus, $\mathbf{S}(C)$ is the most informative by time t with respect to T if the value of its entropy $H_{p(i,t)}$ is minimized. Before we present the details of the estimation, Lemma 1 is introduced as a property of a summary's entropy.

Lemma 1. *If two summaries $\mathbf{S}(T)$ and $\mathcal{S}'(T)$ satisfy $d(\mathbf{S}(T)) > d(\mathcal{S}'(T))$, $V'(\mathcal{S}'(T) \backslash \mathbf{S}(T)) = \emptyset$, and $|l(v)| > d(\mathcal{S}'(T))$ where $v \in \mathbf{S}(T) \backslash \mathcal{S}'(T)$, then we have $H_{p(i,t)}(\mathbf{S}(T)) \leq H_{p(i,t)}(\mathcal{S}'(T))$ holds.*

The proof is omitted due to the space limitation. It shows that the entropy is smaller for those summaries with greater depth. By Equation 1, to achieve the smallest entropy, we need to minimize α because a smaller α yields a higher weight for depth such that deep summaries are preferred. In the remainder of the section, we present the bounds onα followed by our estimation based on entropy.

Lemma 2. *The depth of a summary $\mathbf{S}(T)$ is bounded by the parameter α as $d(T) \geq \tau/(1-\alpha)$.*

Proof. By Measure 1, we have $\max(y(v)) = d(T), v \in \mathbf{S}(T)$. Given such a node v, we have $(1-\alpha)d(T) \geq \tau$ when we set $w(v) = 0$.

Theorem 1. *Let n as the maximal number of nodes in a summary $\mathbf{S}(T)$ with a threshold τ. The parameter α is bounded as*

$$\tau / \sqrt[d(T)]{\frac{n}{d(T)+1}} \leq \alpha \leq 1 - \tau/d(T). \tag{5}$$

Proof. Lemma 2 confirms the right part of Equation 5. The left part is achieved as follows. Similar to Lemma 2, the maximum fanout of $\mathbf{S}(T)$ is τ/α as a positive integer and larger than 1. Then we obtain $(\frac{\tau}{\alpha})^{d(T)} + \ldots + \frac{\tau}{\alpha} + 1 = \sum_i^{d(T)+1} (\frac{\tau}{\alpha})^i \leq n$. Since $\sum_i^{d(T)+1} (\frac{\tau}{\alpha})^i \leq (d(T)+1)(\frac{\tau}{\alpha})^{d(T)}$, the given $\mathbf{S}(T)$ has at most n nodes, i.e., $(\frac{\tau}{\alpha})^{d(T)} + \ldots + \frac{\tau}{\alpha} + 1 \leq n$, if $(d(T)+1)(\frac{\tau}{\alpha})^{d(T)} \leq n$. By transforming the inequality, the left part follows.

Several studies [9, 22] have shown that most diffusion processes are within 3 hops in social networks. Without loss of generality, we assume that the lower bound of the depth of $\mathbf{S}(T)$ is 3. The bound yields the maximum lower-bounded α. As we know, the minimum α turns out to produce the most informative summaries. Thus, by Equation 5 we have the estimation for α as:

$$\alpha = \tau \sqrt[3]{\frac{4}{n}}, \tag{6}$$

to obtain the minimum entropy. The estimation is therefore able to facilitate summarization regardless of varying dynamics.

Threshold Selection. The goal is to find the most interesting developments of dynamics over time as summaries. This naturally requires OSNet to only focus on the small set of the interesting nodes and edges in a spreading tree T. Our goal is to find a proper threshold that can make the summarization converge fast and produce a small sized summary over time. However, the changes and differences of dynamics challenge the setting of such a threshold. Therefore, a selection mechanism adapting to the trends of dynamics (i.e., rise and fall) is necessary.

The idea of the proposed solution is to maintain a variable τ' for each spreading tree T, which is the maximum value (MAX) of $\xi_{t'}(v_i)$, $v_i \in T$ by time t'. During the summarization, we compare a new interestingness score $\xi_t(v_j)$ with τ': if $\xi_t(v_j) > \tau'$, then $\tau' = \xi_t(v_j)$, and v_j is inserted into the corresponding $\mathbf{S}(T)$. If we have a value of τ' that is large enough, OSNet converges to a relatively steady state until there is a more interesting node, e.g., far away from the seed and with many infectees, to exhibit another rise of the diffusion. Thus, in a summary $\mathbf{S}(T)$ based on MAX, the interesting nodes (by the first condition in Definition 3) in deeper levels always show diffusion rises from those in lower levels. From an interesting node to a node recovered for the next interesting node, the flow is always a diffusion decay.

Other methods than MAX would be possible, e.g., average value (AVG) of $\xi_{t'}(v_i)$ as $\sum_{v_i \in V} \xi_{t'}(v_i)/|V|$. We compare these alternatives experimentally in Section 4.

3.3 Algorithmic Framework and Details

Framework Overview. An overview of OSNet is shown in Figure 1. The input is a diffusion process $\mathcal{D}(G)$ that is captured by a set of indexed spreading trees. There are indexes on storyID and seeds, such that we can insert an interaction into a spreading tree T_i efficiently. By Equation 1, the interestingness-based operator is to evaluate the interestingness of nodes in spreading trees with two parameters, α and τ. We evaluate the interestingness of a node v when it infects new nodes (i.e., $w(v)$ increases). If v has $\xi_t(v) > \tau$, it is inserted into a summary $\mathbf{S}(T_i)$. The summaries are also indexed in the same way as T. We thus insert v into $\mathbf{S}(T_i)$ by searching storyID and seed. Once a node v is inserted into tree T, it is tagged with its branch such that a node cannot be reinserted into the summary $\mathbf{S}(T_i)$. We only insert new nodes and edges into a tree over time, and it is not necessary to rebuild any part of T or $\mathbf{S}(T)$.

When a node v of T_i is to be inserted into $\mathbf{S}(T_i)$ at time t, there may be three cases: 1) $\mathbf{S}(T_i)$ does not exist and v_i is not a seed ($v \notin \mathcal{I}(G)$); 2) $\mathbf{S}(T_i)$ exists and the infector of v in T_i is already in $\mathbf{S}(T_i)$; 3) $S(T_i)$ exists and the infector of v in T_i is not in $\mathbf{S}(T_i)$. Cases 1) and 2) are straightforward. We can create a new tree for case 1); and for case 2),

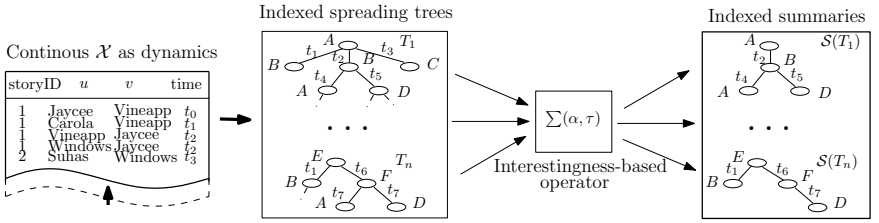

Fig. 1. Overview of the OSNet framework

we insert v as a child of its infector in $\mathbf{S}(T)$. In case 3), the insertion of v renders $\mathbf{S}(T_i)$ disconnected, and the process thus cannot be traced from the seed to v. A solution is to recover all the nodes in the path from the root to v_i. We call this problem the *Recovery Problem*.

Path Recovery. An efficient way in a tree-based data model to solve the *Recovery Problem* is to construct $\mathbf{S}(T)$ as a search tree. The basic idea is that all the siblings at each level of $\mathbf{S}(T)$ are ordered. The canonical ordering is based on timestamps of tree branches (edges) and node labels. If a node v_j gets infected from v_i at time t_i, v_j is inserted in the approach: the timestamps of edge labels of all the siblings on the left are no later than t_i, and the node labels on the left are not lexicographic larger than v_j. Lemma 3 presents the worst case search cost of the search tree.

Lemma 3. *Let a tree T have n nodes and fanout d. The worst case search cost when $d(T)$ is minimum is:*

$$O(\log_d^{(n(d-1)+1)}(\log_d^{(n(d-1)+1)} - 1)\log_2 d).$$

The proof is omitted due to the space limitation.

Algorithm Details. There exist two essential components of OSNet depicted by Algorithm 1: 1) Constructing spreading trees (from lines 3 to 5); 2) Summarizing the most interesting dynamics into $\mathbf{S}(T)$ (from lines 8 to 10). Specifically, the following explains details. We allow users to terminate the summarization process through variable breakFlag in line 2. According to applications, one can also bound the size of $\mathbf{S}(T)$ to abort the algorithm. Note that we have no limitation on the size n. Once a new interaction $x(\delta, v_i, v_j, t)$ arrives (line 3), we call mapT in line 4 to retrieve the T of story δ. Next, branchOut in line 5 is an insertion, which inserts an infectee v_j from v_i with an edge labeled by t into T. We implement each $\mathbf{S}(T)$ as a search tree. From line 6, we summarize the updated node according to Equation 1. If the node's interestingness exceeds the threshold, it shows a diffusion rise, and the node is inserted into $\mathbf{S}(T)$. Parameters are automatically adjusted in line 7 based on discussion in Section 3.2.

Before inserting a node into $\mathbf{S}(T)$, we retrieve the path from the root in line 8 by iteratively pushing an infector (function Push) into list path. We then insert the missed nodes and edges into $\mathbf{S}(T)$ in line 10. These nodes show diffusion decays from the last interesting node, but rises to the next. Summaries are returned if necessary in line 11.

Algorithm 1. Algorithmic description of the OSNet

Input : Network G, seed set $\mathcal{I}(G)$.
Output: A set of summarized spreading trees, $\mathbf{S}(G)$.

begin
  Threshold $\tau \leftarrow 0$; weight parameter $\alpha \leftarrow 0$;
  Boolean beakFlag $\leftarrow false$;
  List path $\leftarrow null$;
  Spreading tree set Set(T) rooted by seeds in $\mathcal{I}(G)$;
  if *beakFlag* $== false$ **then**
    if $x(\delta, v_i, v_j, t) \leftarrow \mathcal{D}(G)[t_{ij}]$ *exists* **then**
      $T \leftarrow$ mapT(Set$(T), \delta)$
      `/* add` x `onto` T. `*/`
      $v_i \leftarrow$ Search(T, v_i) branchOut(v_i, v_j, t)
      if $\xi(v_i) > \tau$ **then**
        $\tau \leftarrow \xi(v_i), \alpha \leftarrow \tau \sqrt[3]{\frac{4}{n}}$ `/* retrieve path from` T. `*/`
        while $v_i.getInfector(T) \notin \mathcal{I}(G)$ **do**
          path.Push$(v_i.getinfector(T))$
        $\mathbf{S}(T) \leftarrow$ getST(Set$(T), T)$ insertPath$(\mathbf{S}(T)$, path$)$
  return *Set*(T)

4 Evaluation

4.1 Experimental Methodology and Settings

The study probes into three questions: 1) **Sense-making Evaluation**: Compared with the state-of-the-art, do summaries generated by OSNet make sense and achieve the goal of capturing interesting dynamics? 2) **Parameter Study**: Can we use fixed parameters? What are the effects of the parameters? Does OSNet converge fast, using MAX or AVG? 3) **Real-life Data**: How does OSNet work on real-life data?

The experiments on synthetic data are used to test whether our methods produce expected results in a controlled environment. We first generate an underlying structure G_0 containing 10,000 nodes. With a random seed set $\mathcal{I}(G_0)$, we then start the propagation for each seed in a breadth-first manner. The number of infectees of a node v obeys the following models to simulate different dynamics: I) Gaussian distribution (G); II) Poisson distribution (P); III) Zipf distribution (Z), which is an approximate power law probability distribution. We define the **modeled number of nodes** to be the number of nodes we choose for a dataset, and we require that their numbers of infectees obey one of the three distributions. To simulate continuous dynamics, we generate the interactions as a data stream with an arrival rate of 1 per millisecond.

Experiments were conducted on a 3.2 GHz Intel Core i5 with 16GB 1600 MHZ DDR3 main memory and running OSX 10.8.5. Algorithms were implemented in JDK 1.6.

4.2 Sense-Making Evaluation

We compare OSNet with several existing algorithms using synthetic data. To enable existing methods to support diffusion processes, we generate a graph sequence for each

dataset, in which each graph aggregates all edges and nodes in a time interval Δt. Due to the space limitation, we only report results for several time intervals. Similar findings apply to other intervals. We compare our techniques against the following state-of-the-art algorithms:

- **DisSim-Alg:** This is a graph compression algorithm that abstracts a large graph into a smaller graph that contains approximately the same information. It is developed based on the notion of dissimilarity between the decompression graph and the original graph. We use an existing implementation [20] and set the weight of an edge to 1 if the adjacent nodes diffuse infection by time t: otherwise, edge weights are set to 0.
- **MDL-Alg:** MDL is a successful and popular technique for graph compression. We compare against a recent study by Navlakha et al. [15] where a graph is compressed and represented as a graph summary and a set of corrections. We use the original GREEDY algorithm that offers the best compression and lowest cost [15]. To enable cliques to be merged into a single supernode, we add self-edges to each node before applying the algorithm.

(**OSNet**): Figures 2 to 4 show an example from t_1 to t_3 using data generated by applying Zipf distribution. The infector as the central node of each group is labeled with a canonical identifier for ease of explanation. The node with identifier 0 is the seed of the propagation. In the figures, the red and darker nodes are the nodes that are already infected; the grey and lighter nodes are other nodes in the synthetic networks. To facilitate visualization, we remove the background nodes and edges in the underlying networks that are not involved in the diffusion process. Figures 5 to 7 present the summaries by OSNet from t_1 to t_3. The results show the incrementality of the summaries. The intuitively interesting nodes are captured, and the summaries are traceable and connected paths, such that we can spot the dynamics from the start to the nodes i) that infect nodes in great quantity; and ii) that are far from the seed. We observe that the summaries in Figures 5 and 6 are the same. Although in the original diffusion process from t_1 to t_2, the diffusion reached nodes 86 and 87 at t_2, the number of infectees is quite few. Compared with the other nodes in $\mathbf{S}(T)$, 86 and 87 are thus not interesting enough to be summarized. This shows that from t_1 to t_2 the diffusion process is not rising according to Definition 5, and OSNet adapts to the changes in diffusion. In contrast at time t_3, both 108 and 109 have many infectees and they are far away from the seed 0. They again expedite the diffusion process such that we capture the two as interesting nodes. If we only summarize the two without including nodes 86 and 87, we loose the connections that allow us to interpret how information propagates. Thus, 86 and 87 are recovered and included. The findings show that OSNet is capable of finding a small set of connected interesting nodes that meaningfully capture the diffusion process.

(**DisSim-Alg**): We vary Δt to generate graph sequences and try various values for the internal compression ratio parameter. We report three of representatives at t_1 in Figure 8. The findings show that the summaries vary a lot w.r.t. compression ratio. Comparing (a) and (c) where (c) is with a higher compression ratio, the graph size of (c) is much smaller but it is with less information of the propagation because *DisSim-Alg* aims to minimize the dissimilarity according to edge weights. To maintain a smaller dissimilarity, some edges or superedges are removed (e.g., (c)). Figure 8(b) shows a summary

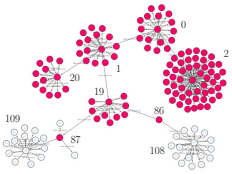

Fig. 2. A diffusion process $\mathcal{D}(G)$ at t_1

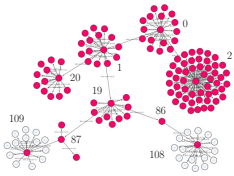

Fig. 3. The snapshot of $\mathcal{D}(G)$ at t_2

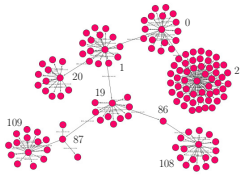

Fig. 4. The snapshot of $\mathcal{D}(G)$ at t_3

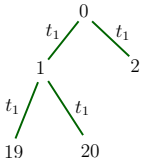

Fig. 5. The summary $\mathbf{S}(T)$ of $\mathcal{D}(G)$ at t_1

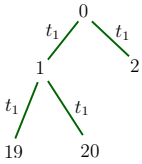

Fig. 6. The summary $\mathbf{S}(T)$ of $\mathcal{D}(G)$ at t_2

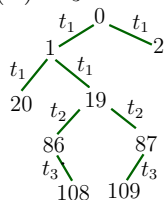

Fig. 7. $\mathbf{S}(T)$ at t_3 with recovery of 86 and 87

caused by using a smaller Δt that is larger than that of (a). This occurs because when the compression ratio is achieved, although new edges and nodes arrive, the algorithm only considers the dissimilarity and does not attempt further compression. As a result, the algorithm does not adapt to dynamics and capture traceable flows well.

(**MDL-Alg**): Figure 9 shows summaries obtained by *MDL-Alg* on the same diffusion process. *MDL-Alg* does not require the users to supply parameters. It computes the best cliques to merge in order to maintain a low cost. The findings show that *MDL-Alg* generates separate cliques, which makes little sense for diffusion process.

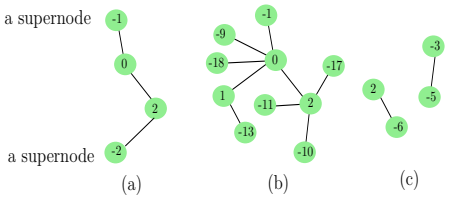

Fig. 8. Summaries by DisSim-Alg at t_1

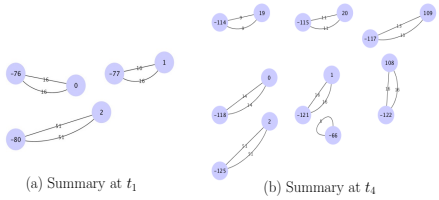

Fig. 9. Summaries by MDL-Alg

4.3 Parameter Study and Understanding

(**Weight α on proScope**): We increase α from 0 to 1 in steps of 0.1. For synthetic data, the depth of the spreading trees is 100, and the modeled number of nodes is 1000. For Gaussian (G) datasets, we set the mean to 100 and the standard deviation to 20. The expect value for Poisson distribution (P) is 50. The maximum $deg^+(v)$ of the Zipf distribution(Z) is 200. Consequently, we have three datasets with $89,037$ (G), $45,306$ (P), and $36,892$ (Z) interactions, respectively. All the interactions are simulated as data streams with an arrival rate of 1 per millisecond. We set τ to 100, which means that the score of a node with expect out-degree $deg^+(v)$ is 100; and we set $\alpha = 1$. Figure 10

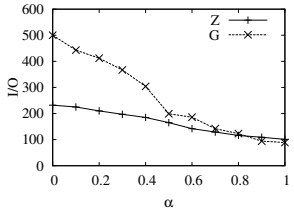

Fig. 10. I/O cost by varying α (a weight parameter)

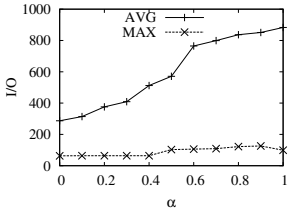

Fig. 11. I/O cost by various strategies on τ selection

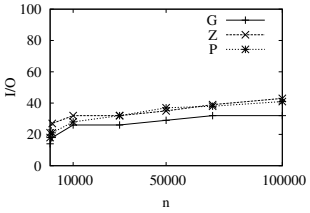

Fig. 12. I/O cost by varying n (the maximal number of nodes)

shows the I/O efficiency with respect to α. We count the I/O cost as the size of summaries, namely the number of interactions in $\mathbf{S}(T)$. The I/O cost for the dataset P is 0 in the experiments, which means that no node in the propagation process gains a score that reaches 100. The findings show that the same fixed threshold does not work well across different datasets. For both G and Z in Figure 10, the I/O cost decreases as α increases. As we know, α controls the weight of **proScope**. Thus, when α is small, the **proRadius** becomes more important in Equation 1. As a result, nodes that are far away from a seed are more likely to be captured, which yields a higher I/O cost.

(**Threshold** τ): We compare our proposal that uses the current maximum score (MAX) against using the average historical score AVG. We use the same datasets as above. Figure 11 shows the findings on dataset G, which indicate that the I/O cost of MAX is much lower than that of AVG. And with a given α, the summarization with MAX converges faster to a relatively steady state than with AVG. MAX requires less updates on the summarized spreading trees than does AVG. Compared with the findings in Figure 10, the I/O cost increases as α increases when using AVG, because a larger value of α yields a larger score. This allows more nodes of a $\mathcal{D}(G)$ to be summarized, which increases the I/O cost. However for MAX, the cost remains almost the same when $\alpha < 0.6$ and it increases only slightly afterwards. We obtain similar results on the other two datasets.

By Equation 6, α never decreases because τ is based on the MAX strategy. This is beneficial for summarization for two reasons: i) With MAX, a larger α allows a bit more nodes to be summarized if diffusion rises; ii) a larger α decreases the influence of **proRadius** such that the summarization converges faster. This keeps OSNet from capturing too many nodes even when many are far away from seeds.

(**Maximum Possible Summary Size** n): We evaluate the effect of n in Equation 6 by varying n from 100 to 100,000. The findings in Figure 12 for all the three datasets show that the I/O cost increases as n increases. A larger n yields a smaller α by Equation 6. Figure 12 thus shows the same I/O cost trend as does Figure 10. However, the variation in Figure 12 is slight. For simplicity, we suggest to set n to be the (average) number of nodes of a T, which is also the maximum number of nodes that can be summarized in an $\mathbf{S}(T)$.

4.4 Evaluation on a Real-life Social Network

We use data from Sina Weibo, a Chinese Twitter-like micro-blogging service platform (http://www.weibo.com) that has two important features that are not yet offered by

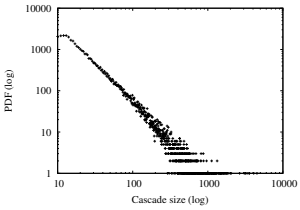

Fig. 13. Cascade size distribution of Weibo Data

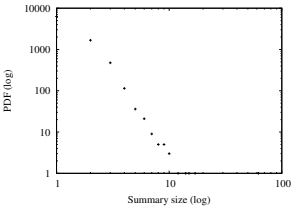

Fig. 14. Summary size distribution of Weibo Data

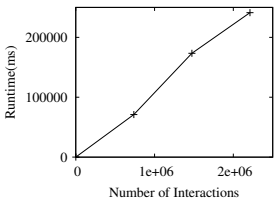

Fig. 15. Runtime (including data arrival time)

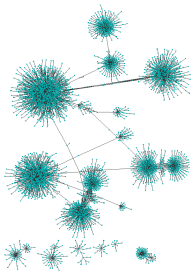

Fig. 16. A sample of diffusion processes

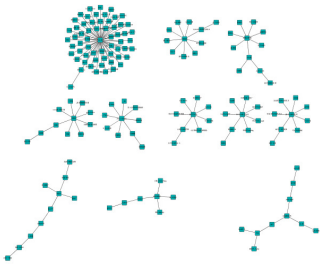

Fig. 17. A sample of summaries

Twitter: 1) A user can comment on any other user's tweets, which yields more user interactions; 2) The retweeting/forwarding chain is visible to the public, which is important for studying diffusion processes. Our dataset covers more than 1.8 million users, and we reconstruct the diffusion processes from their replies. There are 41,561 cascades (diffusion processes) with 2,211,221 interactions. We show that the probability density distribution (PDF) of the cascade size (Log-Log) as a property of the original data in Figure 13. The input is simulated as a stream with an arrival rate of 10 per millisecond, and OSNet outputs 8,647 summaries in which a seed has at least one infectee. Among the results, the summary with the most edges has 62 edges. The PDF of the summary size (Log-Log) is shown in Figure 14, which shows that most of the summaries are small. Figure 15 shows that the runtime of OSNet is proportional to the number of interactions in the dataset. Figure 16 shows a sample of diffusion processes represented by cascades. Figure 17 gives the corresponding OSNet summaries. Although the cascades in Figure 16 may merge on some nodes, the summaries are separated from each other w.r.t. stories. This is because OSNet models diffusion using spreading trees that naturally separate cascades. The summaries in Figure 17 show a vocabulary of patterns, which may be used for event classification or diffusion prediction based on diffusion processes.

5 Related Work

Statistical methods [3] are widely used to characterize properties of large graphs. However, most of these methods do not produce topological summaries, and their results are hard to interpret. Graph pattern mining [7] can be used for summarizing graphs,

but usually yields overwhelmingly large numbers of patterns. Although constraint-based graph mining approaches [23, 24] are introduced to reduce the number of patterns, they only work for specific constraints. Further, summaries of diffusion processes are not inherently frequent.

Next, graph OLAP has been introduced to summarize large graphs [16,19]. However, most studies are designed for static network analyse and are limited to user-specified aggregation operations. Graph compression or simplification mainly focus on generating compact graph representations to simplify storage and manipulation. Much of the work has focused on lossless web graph compression [1,17]. Most of these studies, however, only focus on reducing the number of bits needed to encode a link, and few compute topological summaries since the compressed representation is not a graph. Based on the MDL principle, Navlakha et al. [15] propose an error bounded representation that recreates the original graph within a bounded error. Toivonen et al. [20] merge nodes of a graph that share similar properties. Compared with these studies, our approach is developed to summarize diffusion processes.

As one of the attempts to consider time-evolving networks, Liu et al. [14] compress weighted time-evolving graphs, and they encode a dynamic graph by a three-dimensional array. The goal is to minimize the overall encoding cost of the graph. This is equivalent to compressing a sequence of static graphs according to time slices. Ferlež et al. [5] propose TimeFall in a principled MDL way to monitor network evolution, which clusters text in scientific networks and uses MDL to connect clusters. This class of studies are inherently distinct from ours in four aspects: 1) we use general networks and do not have assumptions on text processing; 2) OSNet takes as argument an interaction stream rather than a timestamped offline network; 3) we do not assume to be given a sequence of time-sliced graphs; 4) we aim to summarize diffusion processes. There are also studies on multiple social networks [12] and their temporal dynamics [8], including trend mining [4]. They focus on tasks different from ours.

6 Conclusion and Future Work

We studied the problem of dynamic network summarization and proposed an online, incremental summarization framework, OSNet, capable of simultaneously capturing the most intuitively interesting summaries that best represented network dynamics.

Several directions for future research are promising, including the development of techniques capable of exploiting the networks underlying diffusion networks, parallel processing of spreading trees, and summarization with structural network changes.

Acknowledgments. This research was supported in part by Geocrowd, funded by the European Commission as an FP7 Peoples Marie Curie Action under Grant Agreement Number 264994; the Singapore National Research Foundation; and the Pinnacle Lab at Singapore Management University. This material is based on work supported by the National Science Foundation under Grant No. CNS-1314632, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National

Science Foundation or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

1. Boldi, P., Vigna, S.: The webgraph framework I: compression techniques. In: WWW, pp. 595–602 (2004)
2. Cha, M., Mislove, A., Gummadi, K.P.: A measurement-driven analysis of information propagation in the Flickr social network. In: WWW, pp. 721–730 (2009)
3. Chakrabarti, D., Faloutsos, C.: Graph Mining: Laws, Tools, and Case Studies. Morgan & Claypool Publishers (2012)
4. Desmier, E., Plantevit, M., Robardet, C., Boulicaut, J.-F.: Trend mining in dynamic attributed graphs. In: ECML/PKDD, pp. 654–669 (2013)
5. Ferlež, J., Faloutsos, C., Leskovec, J., Mladenic, D., Grobelnik, M.: Monitoring network evolution using MDL. In: ICDE, pp. 1328–1330 (2008)
6. Hage, C., Jensen, C.S., Pedersen, T.B., Speicys, L., Timko, I.: Integrated data management for mobile services in the real world. In: VLDB, pp. 1019–1030 (2003)
7. Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)
8. Kumar, R., Novak, J., Tomkins, A.: Structure and evolution of online social networks. In: SIGKDD, pp. 611–617 (2006)
9. Leskovec, J., McGlohon, M., Faloutsos, C., Glance, N.S., Hurst, M.: Patterns of cascading behavior in large blog graphs. In: SDM, pp. 551–556 (2007)
10. Lin, Y.-R., Sundaram, H., Kelliher, A.: Summarization of social activity over time: people, actions and concepts in dynamic networks. In: CIKM, pp. 1379–1380 (2008)
11. Liu, S., Liu, Y., Ni, L.M., Fan, J., Li, M.: Towards mobility-based clustering. In: SIGKDD, pp. 919–928 (2010)
12. Liu, S., Wang, S., Zhu, F., Zhang, J., Krishnan, R.: HYDRA: Large-scale social identity linkage via heterogeneous behavior modeling. In: SIGMOD Conference, pp. 51–62 (2014)
13. Liu, S., Yue, Y., Krishnan, R.: Adaptive collective routing using gaussian process dynamic congestion models. In: SIGKDD, pp. 704–712 (2013)
14. Liu, W., Kan, A., Chan, J., Bailey, J., Leckie, C., Pei, J., Kotagiri, R.: On compressing weighted time-evolving graphs. In: CIKM, pp. 2319–2322 (2012)
15. Navlakha, S., Rastogi, R., Shrivastava, N.: Graph summarization with bounded error. In: SIGMOD Conference, pp. 419–432 (2008)
16. Qu, Q., Zhu, F., Yan, X., Han, J., Yu, P.S., Li, H.: Efficient topological OLAP on information networks. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part I. LNCS, vol. 6587, pp. 389–403. Springer, Heidelberg (2011)
17. Raghavan, S., Garcia-molina, H.: Representing web graphs. In: ICDE, pp. 405–416 (2003)
18. Sun, J., Faloutsos, C., Papadimitriou, S., Yu, P.S.: GraphScope: parameter-free mining of large time-evolving graphs. In: SIGKDD, pp. 687–696 (2007)
19. Tian, Y., Hankins, R.A., Patel, J.M.: Efficient aggregation for graph summarization. In: SIGMOD Conference, pp. 567–580 (2008)
20. Toivonen, H., Zhou, F., Hartikainen, A., Hinkka, A.: Compression of weighted graphs. In: SIGKDD, pp. 965–973 (2011)
21. Xie, R., Zhu, F., Ma, H., Xie, W., Lin, C.: CLEar: A real-time online observatory for bursty and viral events. PVLDB 7(11) (2014)

22. Yang, J., Counts, S.: Predicting the speed, scale, and range of information diffusion in Twitter. In: ICWSM, pp. 355–358 (2010)
23. Zhu, F., Qu, Q., Lo, D., Yan, X., Han, J., Yu, P.S.: Mining top-k large structural patterns in a massive network. PVLDB 4(11), 807–818 (2011)
24. Zhu, F., Zhang, Z., Qu, Q.: A direct mining approach to efficient constrained graph pattern discovery. In: SIGMOD Conference, pp. 821–832 (2013)