Data Min Knowl Disc (2018) 32:1251–1274
https://doi.org/10.1007/s10618-018-0572-z

Efficiently summarizing attributed diffusion networks

Sorour E. Amiri[1] · **Liangzhe Chen**[1] · **B. Aditya Prakash**[1]

Received: 10 December 2017 / Accepted: 11 May 2018 / Published online: 18 May 2018

Abstract Given a large attributed social network, can we find a compact, diffusion-equivalent representation while keeping the attribute properties? Diffusion networks with user attributes such as friendship, email communication, and people contact networks are increasingly common-place in the real-world. However, analyzing them is challenging due to their large size. In this paper, we first formally formulate a novel problem of summarizing an attributed diffusion graph to preserve its attributes and influence-based properties. Next, we propose ANeTS, an effective sub-quadratic parallelizable algorithm to solve this problem: it finds the best set of candidate nodes and merges them to construct a smaller network of 'super-nodes' preserving the desired properties. Extensive experiments on diverse real-world datasets show that ANeTS outperforms all state-of-the-art baselines (some of which do not even finish in *14 days*). Finally, we show how ANeTS helps in multiple applications such as Topic-Aware viral marketing and sense-making of diverse graphs from different domains.

Keywords Attributed graph · Summarization · Topic aware influence maximization

Responsible editor: Jesse Davis, Elisa Fromont, Derek Greene, and Björn Bringmann.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10618-018-0572-z) contains supplementary material, which is available to authorized users.

✉ Sorour E. Amiri
esorour@cs.vt.edu

Liangzhe Chen
liangzhe@cs.vt.edu

B. Aditya Prakash
baditya p@cs.vt.edu

[1] Department of Computer Science, Virginia Tech, Blacksburg, USA

1 Introduction

Suppose we are monitoring network-traffic in a computer system. Legitimate user inputs such as keyboard and mouse events start a series of network requests. Some of these network requests are malicious and they are sent when a user visits a compromised website or the host is infected by malware. This data can be represented by an attributed diffusion graph (Zhang et al. 2014), where nodes are user input and network requests and their attributes are their types and time-stamps. There is an edge between two nodes when one node requests the other one (see Fig. 1a, c). Such networks are diffusion graphs since a content such as a network request is spreading over the network. Can we find malicious network requests as anomalies in this network? Further can we also understand the connection structure of these malicious nodes? These insights can help security analysts to efficiently evaluate the network-traffic dependency and perform further forensics. However, malicious nodes can be triggered anytime and anywhere in the network. This combined with the large size of the graph and sparsity of the malicious requests [0.3% of the total requests (Zhang et al. 2015)] make finding them extremely hard in the Network-traffic graph.

Attributed influence/diffusion networks are commonly seen in other domains as well. Email communication, citation, product co-purchase, and social networks are some other examples. Understanding and analyzing such networks is essential for many applications including viral marketing, immunization, anomaly detection, pattern finding and making sense of propagation. However, the increasingly larger sizes of such networks makes this difficult. While for many tasks there are very efficient algorithms for plain networks [such as the influence maximization problem (Chen et al. 2009)] scaling to millions of nodes, the corresponding state-of-the-art algorithms (e.g. Chen et al. 2015) on attributed networks are slow and can not scale even to graphs with $50k$ nodes (see experiments). Summarizing these attributed networks to get a smaller representation can help us to easily scale-up these data mining tasks. By finding such summaries, we can better visualize and understand the dynamics of the network, and speed-up attribute and influence-related analysis. The idea would be to run the analysis

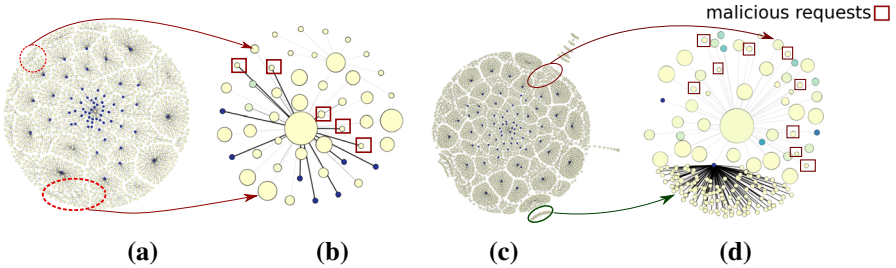

Fig. 1 Finding malicious requests. **a**, **c** Original *Network-traffic* graphs of two different users. Dark blue nodes are type 'user-input' and yellow are type 'network-request'. Note we do not show the time-stamp attributes here. **b**, **d** Summary graphs generated by ANeTS. Color of each supernode is the combination of color of nodes inside it. Size of super-nodes $\propto$ # merged nodes and red squares highlight some of the malicious nodes found (Color figure online)

on the much smaller summary, saving running time, instead of doing it on the entire graph.

In this paper, we study the novel problem of summarizing influence-networks where nodes have features. Despite their importance, there is surprisingly little work on summarizing attributed graphs (see Table 1 and related work). Also, dealing with attributed networks is challenging due to their high dimensional nature. We present an efficient and scalable algorithm ANeTS which takes these graphs and returns a smaller (in nodes and edges) attributed diffusion network of 'super-nodes' and 'super-edges' as the summarization satisfying the following properties.

P1. Diffusion consistency The summary graph must have similar diffusion properties as the original one. It has been shown that the so called epidemic threshold captures the diffusion property as a primary characteristic of the influence-networks (Prakash et al. 2011). Also, the epidemic threshold previously has been used in the literature (Purohit et al. 2014). So, it is natural to maintain it while summarizing influence-networks.

P2. Soft Attribute consistency The summary graph must have similar attribute characteristics as the original one. Hence, we want to group similar nodes into the super-nodes in the summary graph. It means that the nodes in the summary should be as homogeneous as possible. We want soft criteria to measure the homogeneity in attribute properties to be able to summarize the original graph as much as possible with minimum loss in attribute characteristics of it.

P3. Scalability & Parallelizability The summarization must be scalable to the input size and easily parallelizable. It is essential since the large size of today's influence-networks makes it critical to design faster algorithms.

In our *Network-traffic* example, Fig. 1b, d show the corresponding summary graphs (composed of super-nodes and super-edges) generated by our algorithm ANeTS. We also show some of the correspondence between the summary nodes and regions of the original network via circles. Note that these summaries help highlight the main structure of the original graph. Desirably, the super-nodes are homogeneous with respect to the nodes "type" attribute. Further these summaries also help us to find malicious network requests as anomalies. Unmerged singleton nodes in the summary seem interesting; we realized that they are usually of type 'user-input'. However, there are a number of singleton nodes which are structurally very similar to these user inputs but their attributes type is 'network-request'. For example in Fig. 1b there are single nodes that are directly connected to the center of the graph with a high weighted edge, just as user-inputs, but their attribute type is different. These are suspicious and turns out that indeed they are malicious requests based on the ground truth (Zhang et al. 2014). Our summary graphs can correctly detect them and also show their overall connection structure. Note that this also clearly demonstrates the need to take into account *both* structure and attributes to identify these anomalies. Hence a method like ANeTS which preserves both of these properties is required.

The main contributions of our work are:

(I) **Problem formulation** We formulate a novel Attributed Graph Summarization Problem (AGS) to find a smaller diffusion and attribute equivalent summaries.

(II) **Efficient algorithm** We propose an effective sub-quadratic algorithm ANeTS which easily scales to datasets, $\sim 20\times$ larger than the datasets our main com-

Table 1 Feature-based comparison of **ANeTS** with alternative approaches

Properties	**ANeTS**	VEGAS (Shi et al. 2015)	SNAP (Tian et al. 2008)	Plain graph sum. e.g. (Purohit et al. 2014; Qu et al. 2014)	Graph community detection e.g. (Xu et al. 2012)
Generate a summary	✓	✓	✓	✓	
P1. Diffusion consistency	✓			✓	
P2. Soft Attribute consistency	✓	✓			✓
P3. Scalability & Parallelizability	✓			⋰	

ANeTS has all the desirable features. Dotted check mark indicates only some work in that category satisfy the property

petitor used (Shi et al. 2015), while in many cases our baselines do not finish in *14 days* or run out of memory. We also parallelize it to further speed it up.

(III) **Extensive experiments** Our extensive experiments on diverse datasets show the effectiveness and efficiency of our method. Further we also show how to use ANeTS to dramatically speed-up the topic-aware influence maximization task, while maintaining the quality of solutions. Finally, our case studies also show that ANeTS can help make sense of complex attributed networks.

The rest of the paper is organized in the standard way. All proofs are in the supplementary appendix: http://people.cs.vt.edu/esorour/papers/appendix_pkdd2018.pdf.

2 Related work

Our work is related to graph summarization, sparsification, and graph community detection (see a brief comparison in Table 1: ANeTS is the only method that satisfies all the desired properties while others only satisfy some of them).

Community detection or node clustering There is an increasing interest in finding communities for graphs with content/attributes (Akoglu et al. 2012; Perozzi et al. 2014; Xu et al. 2012). Yang et al. (2013) model interactions between network structure and node attributes to find overlapping communities. Some methods combine structural and attribute similarity through a unified distance measure to find communities (Ruan et al. 2013; Zhou et al. 2010). Dhillon et al. (2004) find spectral connection between k-means and normalized cut. Kloumann and Kleinberg (2014) use seed set expansion for community detection. Yang and Leskovec (2015) distinguish functional and structural community. Perozzi et al. (2014) find both focused clusters and outliers from an input exemplar node set. Günnemann et al. (2011) introduce subspace clustering on graphs with feature values. These algorithms only give clusters and not diffusion/attribute equivalent summary graphs like we do.

Plain graph summarization and sparsification The methods in this category aim to find compact representations of graphs which maintain desired properties. The properties can be defined based on specific user queries (Fan et al. 2012), action logs (Mathioudakis et al. 2011), or, more generally, the encoding cost (Navlakha et al. 2008), weights of nodes and edges (Qu et al. 2014), influence-properties (Purohit et al. 2014) and connectivity of nodes (Toivonen et al. 2011). All work on graphs without attributes (not satisfying **P2**).

As pointed out in surveys (Khan et al. 2017; Liu et al. 2016), there are only a few recent approaches summarizing attributed graphs. Tian et al. (2008) and Seah et al. (2012) restrict summary graphs to 'grouped' nodes with the *exact same* subset of attributes. In contrast, we tackle a more general problem of summarizing general attributed graphs with flexible criteria by allowing grouping of nodes with *similar* attributes in principle. Also (Wu et al. 2014) formulate summarization as an information theoretic problem—however their method is not diffusion consistent (**P1**) and scalable & parallel (**P3**). The most closely related work is the VEGAS algorithm (Shi et al. 2015) which summarizes influence flows from a source node. However, their pipeline is specifically designed for citation networks and is not scalable in practice (see experiments).

3 Problem formulation

We formulate our novel problem on DAW-graphs.

Definition 1 (*DAW-graph*) $\mathbf{G}(\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{W})$ is a Directed Attributed Weighted Graph with n nodes ($\mathcal{V}$) and m edges ($\mathcal{E}$). $\mathcal{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_n\}$ are the node attributes ($\in \mathbb{R}^{d\times 1}$) and $\mathcal{W} = \{w_1, \ldots, w_m\}$ are the edge weights (wlog, $\in [0, 1]$).

Goals Our goal is to generate a summary DAW-graph satisfying **P1** and **P2**. Purohit et al. (2014) proposed a plain-graph summarization problem based on merging edges and minimizing a structural distance (between the original graph and the summary) based on influence-based properties. First, we start with their structural distance and merging process. Next, we show how to modify the problem formulation and merging process to preserve both attributes and influence.

Starting point *Structural distance* Motivated by recent work (Prakash et al. 2011), we can maintain the influence based properties by keeping the leading eigenvalue λ of the adjacency matrix of the summary DAW-graph $\mathbf{G^s}$ close to the original $\mathbf{G}$. This is true for almost all cascade models including the IC model, as it maintains the so-called 'epidemic-threshold' (the point where the epidemic 'takes-off'). It has also been used in influence-based summarization of plain graphs (Purohit et al. 2014). Hence we can define the *normalized* structural distance between $\mathbf{G^s}$ and $\mathbf{G}$ as follows,

$$\mathcal{D}_{\text{str}}(\mathbf{G}, \mathbf{G^s}) = \frac{|\lambda_{\mathbf{G}} - \lambda_{\mathbf{G^s}}|}{\lambda_{\mathbf{G}}} \tag{1}$$

where λ_G is the first eigenvalue of the adjacency matrix of G.

Merge operation We want to *merge* edges inside each super-node to get the summary. Suppose we merge two connected nodes 'x' and 'y' with no attributes to form a new super-node 'c'. For determining the new edge-weights, we adopt prior work on influence-based summarization of plain-graphs (Purohit et al. 2014) and define the new edge weights in a way to maintain the *local* influence around the old nodes. Formally,

Definition 2 (*Merge operation*) Assume $N^i(c) = N^i(x) \cup N^i(y)$ indicates the set of in-neighbors (i.e. the set of vertices adjacent to 'c') and $N^o(c) = N^o(x) \cup N^o(y)$ indicates the set of out-neighbors of a super-node 'c' (i.e. the set of vertices adjacent from 'c'). Assume $w_{t,c}$ and $w_{c,t}$ denote the weight of the corresponding edges. If the (super-) node-pair (x, y) is now contracted to a new super-node c, and $w_{(x,y)} = \omega_1$ and $w_{(y,x)} = \omega_2$, then the new edges are weighted as:

$$w_{(t,c)} = \begin{cases} \frac{(1+\omega_1)w_{t,x}}{2} & \forall t \in N^i(x) \backslash N^i(y) \\ \frac{(1+\omega_2)w_{t,y}}{2} & \forall t \in N^i(y) \backslash N^i(x) \\ \frac{(1+\omega_1)w_{t,x}+(1+\omega_2)w_{t,y}}{4} & \forall t \in N^i(x) \cap N^i(y) \end{cases}$$

$$w_{(c,t)} = \begin{cases} \frac{(1+\omega_2)w_{x,t}}{2} & \forall t \in N^o(x) \backslash N^o(y) \\ \frac{(1+\omega_1)w_{y,t}}{2} & \forall t \in N^o(y) \backslash N^o(y) \\ \frac{(1+\omega_2)w_{x,t}+(1+\omega_1)w_{y,t}}{4} & \forall t \in N^o(x) \cap N^o(y) \end{cases} \tag{2}$$

Adding attributes Next, we extend the above formulation.

Attribute distance We also want to maintain the attribute-based properties of the original graph. It implies that the nodes inside each super-node of the summary must be homogeneous to be able to accurately represent them by a super-node. Hence, motivated by K-means, we formulate our problem to minimize the distance of node attributes in each super-node to its centroid. Equation 3 formally defines the normalized attribute-based distance between $\mathbf{G}$ and $\mathbf{G^s}$ (here $|v^s|$ indicates the number of nodes in super-node v^s):

$$\mathcal{D}_{\text{att}}(\mathbf{G}, \mathbf{G^s}) = \frac{\sum_{v^s \in \mathcal{V}^s} \sum_{v \in v^s} ||\mathbf{f}_v - \bar{\mathbf{f}}_{v^s}||_2^2}{\sum_{v \in \mathcal{V}} ||\mathbf{f}_v - \bar{\mathbf{f}}||_2^2},$$
$$\text{Where, } \bar{\mathbf{f}}_{v^s} = \frac{\sum_{v \in v^s} \mathbf{f}_v}{|v^s|} \text{ and, } \quad \bar{\mathbf{f}} = \frac{\sum_{v \in V} \mathbf{f}_v}{n} \tag{3}$$

Note, $\mathcal{D}_{\text{att}}$ measures the homogeneity of super-nodes in the summary graph: lower $\mathcal{D}_{\text{att}}$ leads to higher homogeneity.

Merge operation We also extend the merging process by adding the nodes attributes to it. Motivated by the $\mathcal{D}_{\text{att}}$, the attributes of c should be the centroid of x and y: if $\mathbf{f}_x$ and $\mathbf{f}_y$ are the attribute vectors of x and y, then $\bar{\mathbf{f}}_c = \frac{|x| \cdot \mathbf{f}_x + |y| \cdot \mathbf{f}_y}{|x| + |y|}$ (matching our attribute distance).

Attributed graph summarization (AGS) problem We want to find a summary $\mathbf{G^s}$ which is similar to the original $\mathbf{G}$ in structure ($\mathcal{D}_{\text{str}}$) and attributes ($\mathcal{D}_{\text{att}}$). Formally:

Given a strongly connected DAW-graph $\mathbf{G}(\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{W})$, a reduction fraction $0 < \alpha < 1$, and the merge operation.

Find a summary DAW-graph $\mathbf{G^{s*}}(\mathcal{V}^s, \mathcal{E}^s, \mathcal{F}^s, \mathcal{W}^s)$ by merging edges, with $n^s = (1 - \alpha) \cdot n$, and such that:

$$\mathbf{G^{s*}} = \arg\min_{\mathbf{G^s}} \mathcal{D}(\mathbf{G}, \mathbf{G^s}) \tag{4}$$

$$\text{where, } \mathcal{D}(\mathbf{G}, \mathbf{G^s}) = \frac{1}{2} \cdot \left[\mathcal{D}_{\text{str}}(\mathbf{G}, \mathbf{G^s}) + \mathcal{D}_{\text{att}}(\mathbf{G}, \mathbf{G^s})\right] \tag{5}$$

Remark If there are no attributes in $\mathbf{G}$, we only need to minimize the $\mathcal{D}_{\text{str}}$. Hence, AGS problem reduces to the influence-based plain-graph summarization (Purohit et al. 2014). Also, if $\mathbf{G}$ is a fully connected graph, then all nodes are structurally equivalent. So, nodes will be merged merely based on their attributes and AGS essentially becomes a variant of the classic K-Means problem (Jain 2010) and only minimizes $\mathcal{D}_{\text{att}}$.

4 Our solution

Both of the special cases of our problem are NP-hard (Kanungo et al. 2002; Purohit et al. 2014). Nevertheless, there are heuristic methods to solve each of them based on spectral methods or iterative algorithms. However, existing spectral methods for eigenvalue problems rely on matrix perturbation theory, while those for attribute-based data clustering rely on eigenvectors of similarity graphs. It is not clear how to

combine these two approaches to make them work on DAW-graphs. While the AGS can be formulated probably as a semidefinite program (SDP), it is not clear if standard methods like line search or gradient descent for the AGS problem will be trivial or fast. The main challenge is that it is not obvious how to compute the gradient of largest eigenvalue λ fast. Indeed, people have used SDP solvers for eigenvalue minimization problems (Ghosh and Boyd 2006), but it is very slow. Hence, we decided to estimate the changes in $\mathcal{D}$ (as an approximation of gradient) and follow the popular approach for solving the k-mean style algorithms as they show good performance in practice. In this section, we propose a sub-quadratic algorithm that successively refines a solution while maintaining feasibility. We first describe the serial algorithm and then propose a careful parallel framework for our solution.

4.1 Overview

We present our framework Attributed Network Summarization (ANeTS) to tackle the AGS problem. Our main idea is to combine local matrix perturbations with iterative algorithms for K-means style problems (e.g. Lloyd's algorithm) effectively and efficiently for DAW-graphs. For each $\mathbf{G^s}$ there is a super-nodes assignment vector Φ such that $\Phi[v] \leftarrow v^s$ indicates node v belongs to super-node v^s. Also, $\mathbf{G^s}_\Phi$ is the corresponding summary graph of the assignment Φ. Our approach updates the super-nodes assignments iteratively while reducing the cost function of its corresponding $\mathbf{G^s}$ until convergence.

The framework is as follows:

Step 1 *Initialization*: Start with a super-nodes assignments Φ_0.

While not converged **do** t++

Step 2 *Find the best assignment*: Perturb Φ_t, evaluate the quality of the corresponding summary graph according to Eq. 5 and select the best assignment.

Step 3 *Update assignments*: Update the super-nodes assignments as Φ_{t+1} and the centroids of super-nodes. Also, measure the distance of the summary graph corresponding to the current assignment to the original graph [(i.e. $\mathcal{D}(\mathbf{G}, \mathbf{G^s})$ Eq. 5].

end

Step 4 *Merging*: Use Φ_{final} to extract $\mathcal{V}^s$ and merge nodes in the same super-node and update the edge weights to get $\mathbf{G^s}$.

Even though this framework is fairly straightforward, we face various challenges: (1) Initialization: since AGS is on graphs, we can not randomly initialize Φ_0 unlike the clustering algorithms which can. Moreover, we want to find a good starting point that converges to a high-quality solution in few iterations. (2) Running time: Naïvely following the above framework is very expensive, since in each iteration we must compute the largest eigenvalue of all possible summary graphs and update the centroids.

First, we explain Steps 2, 3, and 4 and then how to speed them up. Then, we design an initialization in Step 1.

4.2 Step 2: Finding the best assignment

In Step 2, we perturb Φ_t to find a better assignment. *First*, we find the best assignment for each node x in the graph [(i.e. $v^s_{best}(x)$], while keeping the assignment of all other nodes fixed to Φ_t—we only move x. The $v^s_{best}(x)$ is the super-node such that moving x into it drops the $\mathcal{D}$ the most (i.e. minimizes the $\mathcal{D}$). Moving x naïvely may lead to some inconsistency in the solution. We will explain the issue later and propose a 'cleanup' method to tackle the problem. Next, we choose the perturbation as the Φ_{t+1} which gives the smallest $\mathcal{D}$ among all nodes x.

4.2.1 Finding the next super-node assignment for each node

As mentioned, $v^s_{best}(x)$ is the best new super-node of x which maximizes the drop of $\mathcal{D}$. We define the *drop function* $\Delta(x, B; \Phi)$ as the change in $\mathcal{D}(\mathbf{G}, \mathbf{G^s})$ when x moves from A to B given the super-node assignment Φ. Formally,

$$\Delta(x, B; \Phi) = \mathcal{D}\left(\mathbf{G}, \mathbf{G^s}_{\Phi[x]\leftarrow A}\right) - \mathcal{D}\left(\mathbf{G}, \mathbf{G^s}_{\Phi[x]\leftarrow B}\right) \tag{6}$$

where $\Phi[x] \leftarrow B$ indicates node x is assigned to super-node B and $\mathbf{G^s}_{\Phi[x]\leftarrow B}$ is its corresponding summary graph. Hence, $v^s_{best}(x) = \arg\max_{v^s \in \mathcal{V}^s} \Delta(x, v^s; \Phi_t)$. Since we want to *localize* the search space to find $v^s_{best}(x)$, we only examine the super-nodes that contain at least one neighbor of x. Furthermore, the Φ_{t+1} in Step 2 is the assignment maximizing the largest drop of each node i.e. the next best assignment maximizes $\Delta(x, v^s_{best}(x); \Phi_t)$ over all x.

4.2.2 Cleanup

We merge edges of $\mathbf{G}$ to get super-nodes. Hence, we have the following requirement,

Requirement 1 (Connectivity of super-nodes) *The corresponding sub-graph of any super-node $v^s \in \mathcal{V}^s$ in the original graph $\mathbf{G}$ is weakly connected.*

If we perturb Φ_t naïvely we may end up with a Φ_{t+1} with disconnected corresponding super-nodes, violating Requirement 1. It can be formally explained using the following definition for the validity concept,

Definition 3 (*Valid $\mathcal{V}^s$*) A super-node set $\mathcal{V}^s$ is valid iff

- (*Condition 1*) $|\mathcal{V}^s| = (1 - \alpha) \cdot |\mathcal{V}|$
- (*Condition 2*) Each $v^s \in \mathcal{V}^s$ is a weakly connected subgraph.

Φ_{valid} is the corresponding assignment of a valid $\mathcal{V}^s$. Figure 2a shows an example of such a situation: there are two super-nodes A and B in the graph shown by dotted circles. If we just move node x to B, super-node A will be divided into two connected components (Fig. 2b), which violates Condition 2 above. Also, if we assign each connected component to a new super-node we will violate Condition 1.

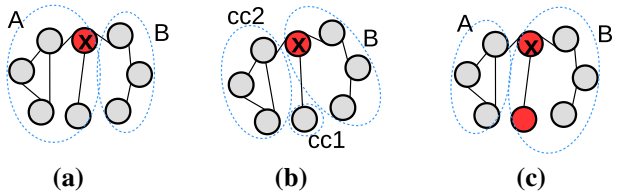

Fig. 2 Clean up example

Algorithm 1: Clean-up algorithm

Data: Φ, $A \in \mathcal{V}^s$, $B \in \mathcal{V}^s$, $x \in \mathcal{V}$
Result: Φ^x_{valid}
1 Initialize $\Phi^x_{\text{valid}} = \Phi$;
2 Move x form super-node A to B;
3 find $CC = \{cc_i\}_{i=1}^{q}$ using BFS on A;
4 $cc^* = \underset{cc_i \in \{cc_1, cc_2, \ldots, cc_q\}}{\arg\min} \mathcal{D}(\mathbf{G}, \mathbf{G^s}_{\Phi[\mathbf{cc_i}] \leftarrow B})$;
5 $\Phi^x_{\text{valid}} = \Phi[A/cc^*] \leftarrow B$;

How can we resolve this problem? An intuitive solution is to move more nodes to B along with x to keep A connected. As shown in Fig. 2c moving cc_1 and x to B keeps the super-nodes connected and valid. So, it implies that we must move a group of nodes instead of just one. We also want to get a $\mathcal{V}^s$ with high quality. So, we need to select the best group of nodes to move and have a valid and high-quality $\mathcal{V}^s$. Hence, we propose an additional step to 'cleanup' the assignment of nodes in the graph.

To maintain the validity of the super-node set we propose the Clean-up algorithm (see Algorithm 1). Assume we want to move node x from super-node A to its best super-node B, which makes A disconnected. We now select additional nodes to move along with x to B. *First*, we extract all connected components $\{cc_i\}_{i=1}^{q}$ generated in A after moving x by using BFS algorithm. *Second*, we move all nodes of A to B except one connected component. It guarantees that both super-nodes A and B remain connected. Now the best connected component cc_i to keep in A is the one which gives the lowest drop if all of its nodes move to B. Formally, $cc^* = \arg\min\ \mathcal{D}(\mathbf{G}, \mathbf{G^s}_{\Phi[cc_i] \leftarrow B})$, for all $\{cc_i\}_{i=1}^{q}$, where $\Phi[cc_i] \leftarrow B$ indicates all nodes in connected component cc_i are assigned to B and, $\mathbf{G^s}_{\Phi[cc_i] \leftarrow B}$ is its corresponding summarized graph. We compute $\mathcal{D}(\mathbf{G}, \mathbf{G^s}_{\Phi[cc_i] \leftarrow B})$ by measuring the drop of $\mathcal{D}$ for moving all nodes of cc_i to B. We call it *group drop function* and calculate it as,

$$\Delta(\mathbf{cc}, B; \Phi) = \mathcal{D}\left(\mathbf{G}, \mathbf{G^s}_{\Phi[\mathbf{cc}] \leftarrow A}\right) - \mathcal{D}\left(\mathbf{G}, \mathbf{G^s}_{\Phi[\mathbf{cc}] \leftarrow B}\right) \tag{7}$$

where **cc** is a connected subgraph in super-node A. Finally, the valid assignment of x is, $\Phi^x_{\text{valid}} = \Phi[A/cc^*] \leftarrow B$. Hence, we move as many nodes as necessary to keep the assignment valid in each iteration.

Lemma 1 *The time complexity of* Clean-up *algorithm (Algorithm 1) is* $O(m_A)$ *where* m_A *is the number of edges of the super-node* A.

In summary, the Clean-up algorithm is the best local approach to keep the validity of the $\mathcal{V}^s$ in *linear time*.

4.2.3 The next best assignment

The next best assignment is the one which minimizes the distance i.e.

$$\Phi_{t+1} = \underset{\Phi^x_{\text{valid}}}{\arg\min}\ \mathcal{D}\left(\mathbf{G}, \mathbf{G^s}_{\Phi^x_{\text{valid}}}\right) \qquad \forall x \in \mathcal{V}. \tag{8}$$

4.3 Step 3: Updating and step 4: merging

After finding Φ_{t+1} in Step 2 we update the super-nodes' information. To that end, we update the attribute vectors of all the corresponding super-nodes using Eq. 3, and $\mathcal{D}(\mathbf{G}, \mathbf{G^s}_{\mathbf{G^s}_{\Phi_{t+1}}})$.

We iterate over Steps 2 and 3 until convergence, and obtain the best assignment Φ_{final} that optimizes Eq. 4. In Step 4, we *create* the actual summary $\mathbf{G^s}$ by repeatedly applying the merge operation (Definition 2) on edges inside the super-nodes corresponding to Φ_{final}.

4.4 Speeding up

In Step 2, we need to evaluate the change in $\mathcal{D}$ (Eqs. 6, 7) in the drop functions, while moving a set of nodes from super-node A to B. Consider Eq. 6:

$$\begin{aligned}
\Delta(\mathbf{x}, B; \Phi) &= \mathcal{D}\left(\mathbf{G}, \mathbf{G^s}_{\Phi[\mathbf{x}]\leftarrow A}\right) - \mathcal{D}\left(\mathbf{G}, \mathbf{G^s}_{\Phi[\mathbf{x}]\leftarrow B}\right) \\
&= 1/2\left[\mathcal{D}_{\text{str}}\left(\mathbf{G}, \mathbf{G^s}_{\Phi[\mathbf{x}]\leftarrow A}\right) - \mathcal{D}\left(\mathbf{G}, \mathbf{G^s}_{\Phi[\mathbf{x}]\leftarrow B}\right)\right] \\
&\quad + \left[\mathcal{D}_{\text{att}}\left(\mathbf{G}, \mathbf{G^s}_{\Phi[\mathbf{x}]\leftarrow A}\right) - \mathcal{D}\left(\mathbf{G}, \mathbf{G^s}_{\Phi[\mathbf{x}]\leftarrow B}\right)\right] \\
&= 1/2\left[\Delta\mathcal{D}_{\text{str}}(\mathbf{x}, B; \Phi) + \Delta\mathcal{D}_{\text{att}}(\mathbf{x}, B; \Phi)\right]
\end{aligned}$$

as $\mathcal{D} = \frac{1}{2}[\mathcal{D}_{\text{str}} + \mathcal{D}_{\text{att}}]$. Our idea is to reduce the time-complexity by approximating $\Delta\mathcal{D}_{\text{str}}$ and $\Delta\mathcal{D}_{\text{att}}$ in Eqs. 6 and 7 instead of computing them naively. We also give a practical improvement which reduces the constant factors.

4.4.1 Fast computation of structural distance

Computing $\Delta\mathcal{D}_{\text{str}}$ involves computing the change in the leading eigenvalue of possible $\mathbf{G^s}$ and $\mathbf{G}$. Doing this naively will take $O(mnn^s)$ to compute the λ of all possible assignments (assuming we use the Lanczsos method which takes $O(m)$ edges time). To speed this up, we approximate $\Delta\mathcal{D}_{\text{str}}$ instead. Using matrix-perturbation theory, Purohit et al. (2014) derive a constant-time first-order approximation $\Delta\widehat{\lambda}_{x,y}$ for $\Delta\lambda_{x,y}$ after merging one edge (x, y) in a non-attributed graph. However $\Delta\mathcal{D}_{\text{str}}$ involves multiple edges, and generalizing first order approximation of $\Delta\widehat{\lambda}_{x,y}$ for merging multiple edges is still costly (see appendix). Hence, we intuitively estimate the change of λ by

just aggregating the $\Delta\widehat{\lambda}_{x,y}$ of all merged edges in every super-node. Consider Eq. 6 again: as we are only moving node x from A to B, we no longer merge the edges between x and other members of A. Instead we merge edges between x and nodes in B. Using this fact $\Delta\mathcal{D}_{\text{str}}$ for Eq. 6 can be estimated as:

$$\Delta\mathcal{D}_{\text{str}}(\mathbf{x}, B; \Phi) \simeq \frac{\lambda_{\mathbf{G}^{\mathbf{s}}\Phi[x]\leftarrow B} - \lambda_{\mathbf{G}^{\mathbf{s}}\Phi[x]\leftarrow A}}{\lambda_{\mathbf{G}}} \simeq \frac{\sum_{y\in B}(\Delta\widehat{\lambda}_{x,y}) - \sum_{z\in A}(\Delta\widehat{\lambda}_{x,z})}{\lambda_{\mathbf{G}}}. \quad (9)$$

We can similarly estimate $\Delta\mathcal{D}_{\text{str}}$ for Eq. 7 as well. See the complete derivation in appendix. This estimation reduces the time complexity of evaluating $\Delta\mathcal{D}_{\text{str}}$ from $O(mnn^s)$ to $O(m)$.

4.4.2 Fast computation of attribute distance

We need to compute $\Delta\mathcal{D}_{\text{att}}$ as well from above. For example, in Eq. 6 it means that we need to reevaluate the $\bar{\mathbf{f}}_B$ for each possible move of x to a super-node B which costs $O(dn^2)$ in each iteration. Therefore, to speed-up ANeTS, we disregard the change of centroids while computing $\mathcal{D}_{\text{att}}$. It is reasonable to assume that the centroids themselves change smoothly between the moves. Hence, we approximate as follows:

$$\Delta\mathcal{D}_{\text{att}}(\mathbf{x}, B; \Phi) \simeq \frac{||x - \bar{\mathbf{f}}_A||_2^2 - ||x - \bar{\mathbf{f}}_B||_2^2}{\sum_{v\in\mathcal{V}} ||\mathbf{f}_v - \bar{\mathbf{f}}||_2^2} \quad (10)$$

In the same way, we can estimate $\Delta\mathcal{D}_{\text{att}}$ for the group drop function in Eq. 7 too. See the complete derivation in appendix. This estimation reduces the complexity of evaluating $\Delta\mathcal{D}_{\text{att}}$ from $O(dn^2)$ to $O(dn)$.

4.4.3 A practical improvement

Note that Φ_t and Φ_{t+1} differ in only two super-nodes (we move a group of nodes from one super-node to another one). So, in Step 3, we only need to update the centroids of the two updated super-nodes. Also, we can divide the nodes of $\mathbf{G}$ into two groups: (group 1) Nodes in the updated super-nodes and (group 2) the rest. In each iteration, we need to recalculate the drop function of nodes in group 1 for all possible super-nodes. However, we need to recalculate the drop function of nodes in group 2 for the two updated super-nodes only. Although this does not change the time-complexity, it does help considerably in avoiding unnecessary calculations in ANeTS.

4.5 Step 1: Initialization

One issue still remains: we need to initialize the assignments at the beginning of ANeTS. We can not randomly assign nodes into super-nodes as it may not give a feasible solution. Our intuition is to gradually merge *unimportant* edges with *similar* (attributed) end-nodes to end up with a high-quality starting point as follows: (1) Compute a 'score' for each edge in $\mathbf{G}$. It estimates the $\mathcal{D}_{\text{str}} + \mathcal{D}_{\text{att}}$ of merging the end

Algorithm 2: The ANeTS framework

Data: DAW-graph $\mathbf{G}(\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{W}), \alpha$
Result: DAW-graph $\mathbf{G^S}(\mathcal{V}^s, \mathcal{E}^s, \mathcal{F}^s, \mathcal{W}^s)$

1 //Step 1: Initializing super-nodes;
2 $t \leftarrow 0, \Phi_0 =\leftarrow$ INITIALIZATION$(\mathbf{G}, \alpha)$ (Sec. 4.5);
3 **while** ***not*** *converged* **do**
4 $\quad t \leftarrow t + 1$;
5 $\quad$ //Step 2: Find the next best super-node assignment Φ_{t+1} (Sec. 4.2);
6 $\quad$ **for** $x \in \mathcal{V}$ **do**
7 $\quad\quad \Phi^x_{\text{valid}} =$ Clean-up$(\mathbf{G}, v^s_l(x), v^s_{best}(x), x)$;
8 $\quad\quad$ Compute $\mathcal{D}(\mathbf{G}, \mathbf{G^S}_{\Phi^x_{\text{valid}}})$;
9 $\quad \Phi_{t+1} = \arg\min_{\Phi^x_{\text{valid}}} \mathcal{D}(\mathbf{G}, \mathbf{G^S}_{\Phi^x_{\text{valid}}})$ (Eq. 8);
10 $\quad$ //Step 3: Update super-nodes (Sec. 4.3);
11 $\quad$ Update $\bar{\mathbf{f}}_{v^s}$ for each $v^s \in \mathcal{V}^s$ corresponded to Φ_{t+1};
12 $\quad$ Update $\mathcal{D}(\mathbf{G}, \mathbf{G^S}_{\Phi_{t+1}})$ (Eq. 5). ;
13 //Step 4: Merging (Sec. 4.3);
14 Merge nodes of each super-node to get $\mathcal{E}^s$, and $\mathcal{W}^s$;

nodes of each edge in $\mathbf{G}$. (2) Gradually assign end nodes of edges with the lowest score to the same super-node until we get the target number of super-nodes. We compute the score of an edge $(x, y) \in \mathcal{E}$ as, $score(x, y) = \frac{\Delta\widehat{\lambda}_{x,y}}{\lambda_{\mathbf{G}}} + \frac{||\bar{\mathbf{f}}_x - \bar{\mathbf{f}}_y||_2^2}{\sum_{v \in \mathcal{V}} ||\mathbf{f}_v - \bar{\mathbf{f}}||_2^2}$. Note that although our initialization is very intuitive, it does not directly minimize Eq. 5. So, we need to follow our steps of ANeTS to improve the results for the AGS problem.

4.6 The complete (serial) algorithm

Algorithm 2 is the pseudo-code of the complete serial version of ANeTS algorithm. Also, we have the following lemmas,

Lemma 2 ***ANeTS converges*** *in finite number of iterations and reduces $\mathcal{D}$ in each iteration.*

Lemma 3 ***Time complexity of serial ANeTS*** *is sub-quadratic $O(m \cdot \log m + n_{itr} \cdot (\mathrm{n}_x \mathrm{m}_x + n^s \cdot n \cdot d))$. Also, the memory complexity of ANeTS is linear $O(n \cdot d + m)$. The n, d, and n_{itr} are the number of nodes, attributes and iterations respectively. E_o is the number of edges between nodes in different processors and n_x and m_x is the number of nodes and edges of the largest super-node.*

4.7 Scaling-up ANeTS: Parallelization

Although the time complexity of serial ANeTS is better than state-of-the-art competitors, it does not scale up to graphs with more than $20k$ nodes. The bottleneck is in finding Φ_{t+1} in each iteration (i.e. Step 2: lines 6–8) and updating the attributes of super-nodes (line 12). This motivates us to propose an efficient parallel framework

Algorithm 3: Parallel Clean-up

Data: Φ, a set of triples $Tr = (A \in \mathcal{V}^s, B \in \mathcal{V}^s, x \in \mathcal{V})$, processors node assignments
Result: Set of Φ^x_{valid}
Initialize $\Phi^x_{\text{valid}} = \Phi$ for each triple in Tr in parallel;
Move each x in Tr form its super-node A to B in parallel;
In each processor j find CC_j of N^j_A associated with each triple in parallel;
//Merge connected components;
for *each triple in* Tr ***and*** $(a, b) \in \mathcal{E}$ *where* a *and* b *are in different processors* i *and* j **do**
    merge CC_i and CC_j;
for *each triple in* Tr **do**
    $cc^* = \underset{cc_i \in \{cc_1, cc_2, \ldots, cc_n\}}{\arg\min} \mathcal{D}(\mathbf{G}, \mathbf{G^s}_{\Phi[cc_i] \leftarrow B})$;
    $\Phi^x_{\text{valid}} = \Phi[A/cc^*] \leftarrow B$;

for ANeTS to make it faster. However, designing it is not straightforward due to the complexity of the Clean-up step. For example, a first-cut approach would simply distribute the nodes to processors. Each processor finds the best super-nodes, and does the clean-up separately. The main drawback is that, as real networks are skewed, there would be unbalanced workloads due to different super-node sizes.

(3) Our approach We run Clean-up in parallel and make sure the processors have balanced workload. Instead of having $\mathbf{G}$ as global information, each processor accesses a part of $\mathbf{G}$ associated with its assigned nodes. We do this process for all nodes in parallel—see Algorithm 3. Assume $N_A = \{v_{A_1}, v_{A_2}, \ldots, v_{A_h}\}$ is the set of nodes in super-node A after moving x. In the serial algorithm (i.e. $p = 1$), we run BFS to detect CCs in the subgraph of A. When $p > 1$, nodes in A can be in different processors. Hence, we propose the following procedure to find CCs in parallel: Assume N^j_A is the subset of nodes in A which are in processor j. (I) Extract CCs of each N^j_A using BFS. So for super-node A we will have p sets of CCs computed in each processor. (II) Next, merge these p sets to get CCs of A. In this step, merge two CCs into one iff there is at least one edge between them. Continue merging CCs until we can not merge more. Finally, we compute the drop of distance for each component by accumulating the drop of the nodes inside them.

Lemma 4 ***Time complexity of parallel ANeTS*** *is* $O\left(m \cdot \log m + n_{itr} \cdot \left(\frac{\mathrm{n}_x \mathrm{m}_x + n^s \cdot n \cdot d}{p} + E_o + p \cdot d\right)\right)$*; p is #processors and E_o is #edges with end-points in different processors.*

5 Sample application: topic-aware diffusion

Our summary $\mathbf{G^s}$ can be used to speed-up other applications while maintaining performance. Next we show how to use ANeTS to scale the topic-aware influence maximization (TIM) problem (Chen et al. 2015). This problem is useful in many applications such as predicting activity or product/opinion adoption, in various kinds of datasets such as tweets, DBLP, etc.

Problem setting Consider a social graph $\mathrm{SG}(\mathrm{V_{SG}}, \mathrm{E_{SG}}, \mathbf{P})$, where $\mathrm{V_{SG}}$ is a set of people and $\mathrm{E_{SG}}$ with directed influence relations (edges) between people. $\mathbf{P}(u, v) = [\mathrm{p}_1, \ldots, \mathrm{p}_T]$ is the weight vector of each edge where p_i indicates the probability that node u can influence v in topic i (T is the number of topics). The Topic-aware Influence Propagation (TIP) model (Barbieri et al. 2012) is a cascade model used to model influence-propagation of a topic-aware content Q along the edges of a social graph SG. The content $Q = [q_1, \ldots, q_T]$ is represented by the topic distribution of the content where q_i is the probability of topic 'i' in content Q. A vertex $v \in \mathrm{V_{SG}}$ is active if it has been influenced by (or adopted) Q and inactive otherwise. There is an initial set of 'seed' S active nodes. Each active node u has a single chance to activate each of its currently inactive neighbor v independently with probability $\mathbf{P}(u, v) \cdot Q'$. This cascade process continues until no more activations are possible. The final number of active nodes is the 'influence spread' of seed set S given the content Q. Formally the TIM problem is:

Given a social graph $\mathrm{SG}(\mathrm{V_{SG}}, \mathrm{E_{SG}}, \mathbf{P})$, its TIP model, a content $Q = [q_1, \ldots, q_T]$, and a budget 'k',

Find $S^*_{TIM} = \underset{S_{TIM} \subseteq \mathrm{V_{SG}}, |S_{\mathsf{TIM}}| = k}{\arg\max} \sigma_{TIP}(S_{\mathsf{TIM}}|Q)$.

Our approach The TIM problem is NP-hard, and current algorithms (Chen et al. 2015) are slow (e.g. they do not finish for graphs larger than $50k$ in 10 days). We use the state-of-the-art algorithm TIM (Chen et al. 2015), and propose ANeTS-based AnT to scale it up.

Firstly, SG is not a DAW-graph. So we first convert it to a DAW-graph $\mathbf{G}$ by mapping the edge attributes of SG to node attributes. Next, we summarize $\mathbf{G}$ and map it back to a smaller social graph SG^s of super-nodes and super-edges. Then, we run the TIM algorithm on (the much smaller) SG^s to get a seed-set of super-nodes. Finally, we just randomly select a node in each super-node of selected seeds as the final solution 'S'. Intuitively, the selected seed set using this approach is an accurate solution because we merge nodes into super-nodes with similar topic influence probabilities to their neighbors in the original graph. Hence randomly picking any node within the super-node as a seed should give the same expected influence spread (i.e. $\sigma_{TIP}(S^*_{tim}|Q) \simeq \sigma_{TIP}(S^*_{\mathsf{AnT}}|Q)$). Hence our framework AnT is:

(1) **Map weights to attributes** Convert the social graph $\mathrm{SG}(\mathrm{V_{SG}}, \mathrm{E_{SG}}, \mathbf{P})$ to a DAW-graph $\mathbf{G}(\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{W})$ with same number of nodes and edges using the following simple scheme: (I) For each node $u \in \mathcal{V}$ we have a T dimensional attribute vector $\mathbf{f}_u \in \mathcal{F}$. This $\mathbf{f}_u$ is the average value of all edges starting from it corresponding node $u_{\mathrm{SG}} \in \mathrm{SG}$. The mapping captures the average influence behavior of a node $u_{\mathrm{SG}} \in \mathrm{SG}$ as the attribute of u in $\mathbf{G}$. (II) Assign an arbitrary uniform probability to all edges of DAW-graph as $\mathcal{W}$.

(2) **Get a summary** Summarize $\mathbf{G}$ to $\mathbf{G^s}$ using ANeTS.

(3) **Map attributes to weights** Since $\mathbf{G^s}$ is a DAW-graph we convert it back to a social graph $\mathrm{SG}^s(\mathrm{V^s_{SG}}, \mathrm{E^s_{SG}}, \mathbf{P}^s)$. In $\mathbf{G^s}$, for each super node we have T attributes. The edge weight from super-node $u^s_{\mathrm{SG}} \in \mathrm{V^s_{SG}}$ to $v^s_{\mathrm{SG}} \in \mathrm{V^s_{SG}}$ is simply the attribute of its corresponding node $u^s \in \mathbf{G^s}$ [i.e. $\mathbf{P}(u^s_{\mathrm{SG}}, v^s_{\mathrm{SG}}) = \bar{\mathbf{f}}_{u^s}$]. This mapping maintains the average influence behavior of super-nodes.

(**4**) **Solve** Run the state-of-the-art TIM algorithm (Chen et al. 2015) on the smaller graph SG^s to get k seed super-nodes.
(**5**) **Pull back** To get the seed nodes of the original graph we randomly select a node inside the seed super-nodes $s_1, \ldots, s_k$.

6 Empirical study

We design various experiments to evaluate ANeTS in this section. We implemented ANeTS in Python. Our experiments were conducted on a 4 Xeon E7-4850 CPU with 512 GB of 1066 MHz main memory and we will release our code for research purposes.
Datasets We collected a number of datasets with different scales and domains such as social networks, system network traffic, movie rating, co-authorship and email communication network for our experiments. See Table 2 for details. Note, the size of the attributed graphs is $O(\#Nodes \times \#Attr + \#Edges)$ in contrast with the size of plain-graphs with no attributes which is only $O(\#Edges)$.

(1) *Disney* (Perozzi et al. 2014) Nodes are movies and links indicated a co-purchase relation. Attributes are price, average rating, etc.
(2) *Memetracker* (Gomez-Rodriguez et al. 2012) Nodes are blog/website and links means who-copies-from whom. Attributes are the type (blog/website), the number of posts and the rest indicate the time node adopts a popular meme.
(3) *Citeseer*/*PubMed* (Ruan et al. 2013; Sen et al. 2008) Nodes are publications, and edges indicate citation relationships. Attributes in *Citeseer* are binary and show stemmed words from the publications. In *PubMed* attributes are a TF/IDF weighted word vector of each publication.
(4) *Facebook*/*Google+*/*YouTube* (Gomez-Rodriguez et al. 2012; Perozzi et al. 2014) Nodes are users and links indicate the friendship. Attributes are extracted from users profiles. The attributes for Facebook and Google+ dataset are binary values and YouTube data has real values.
(5) *Enron* (Gomez-Rodriguez et al. 2012) Nodes are email addresses and edges shows email transmissions in Enron Inc.. Attributes are the average number of recipients, average content length, etc.

Baselines We compare ANeTS to the most related approaches, including a matrix decomposition based algorithm for summarizing attributed graphs [VEGAS (Shi et al. 2015)], influence-based summarization algorithm for plain graphs (CoarseNET (Purohit et al. 2014)], attributed-graph clustering/community algorithms [BAGC (Xu et al. 2012), CODICIL (Ruan et al. 2013)], and a min-cut partitioning approach [METIS (Karypis and Kumar 1998)].

Note that BAGC, CODICIL, and METIS only find node clusters/communities: to adapt them to find a summary graph, we merge nodes in the same cluster/community using Definition 2. Also, note that the datasets we used are much larger than the datasets that our main competitor VEGAS used (i.e. $\sim 20\times$ larger) and similar to the other graph community detection algorithms such as BAGC.We updated Sect. 6 accordingly. For each method, we calculated the time complexity. In overall, their time complexity varies from linear and sub-quadratic (e.g., METIS and CoarsNET) to quadratic (e.g., VEGAS).

Table 2 Details about our 8 datasets

Dataset	#Nodes	#Edges	#Attr	Size = #Nodes × #Attr + #Edges
1. *Disney*	124	335	28	3807
2. *Memetracker*	960	5000	76	77,960
3. *Citeseer*	2111	3669	3703	7,820,702
4. *Facebook*	4039	88,234	1402	5,750,912
5. *Google+*	3820	280,742	6	303,662
6. *Enron*	13,533	176,967	20	447,627
7. *PubMed*	19,717	44,336	500	9,902,836
8. *YouTube*	77,381	367,151	300	23,581,451

6.1 Performance of ANeTS

Effectiveness We calculate $\mathcal{D} = \frac{1}{2}[\mathcal{D}_{\text{str}} + \mathcal{D}_{\text{att}}]$ to show the performance of different algorithms. On average ANeTS took 200 itrs. to converge and reduces the initial $\mathcal{D}$ by 68% and Clean-up triggered around half of the times. It shows that even with our intuitive initialization, multiple iterations and Clean-up are necessary to find a high-quality summary Note in many cases VEGAS, BAGC and CODICIL do not even finish after 14 days! or run out of memory. It shows, AGS is challenging and clearly ANeTS can fill the gap between attributed and plain graph summarization.
Minimizing $\mathcal{D}$ We show the results of some datasets in Table 3. It shows the $\mathcal{D}$, $\mathcal{D}_{\text{str}}$, and $\mathcal{D}_{\text{att}}$ with $\alpha = 0.7$ and Fig. 3 shows the results with different α values (since VEGAS is very slow, we could not run it for multiple α values). ANeTS has constantly the best performance for all datasets getting a 44% on avg. improvement which is significant considering the challenges of the problem. ANeTS reduces the $\mathcal{D}$ as much as the graph allows: e.g. *Google+* (Fig. 3b) is a dense graph with avg. deg. 146.9 and $\mathcal{D}$ of ANeTS is much lower than the baselines even with high reduction factors. On the other hand, *Enron* (Fig. 3c) is a sparser graph with avg. deg. 26.1. So, performance degrades mainly with high reduction factors similar to baselines.
Balanced summary Table 3 clearly shows that ANeTS minimizes $\mathcal{D}$ by minimizing the $\mathcal{D}_{\text{str}}$ along with $\mathcal{D}_{\text{att}}$ as much as possible while other approaches usually capture one of the aspects. For example, CoarseNET only optimizes $\mathcal{D}_{\text{str}}$, and usually produces a much larger $\mathcal{D}_{\text{att}}$ than ANeTS, hence giving less meaningful summaries. Note, the very sparse structure of *PubMed* (Avg. deg. = 4.4) results in a similar score for ANeTS and CoarseNET. However, ANeTS gives a more balanced summary (i.e. $\mathcal{D}_{\text{str}}$ and $\mathcal{D}_{\text{att}}$ are evenly minimized in comparison with other methods). Similarly, CODICIL has good performance on $\mathcal{D}_{\text{att}}$ while does not preserve the diffusive property (a larger $\mathcal{D}_{\text{str}}$ value in comparison with ANeTS).
Homogeneity The low $\mathcal{D}_{\text{att}}$ of ANeTS implies the super-nodes of $\mathbf{G^s}$ are homogeneous. To examine it we computed the entropy of nodes attributes in the same super-node as a measure for homogeneity of super-nodes as follows,

$$H_{\mathbf{G^s}} = \frac{1}{|\mathcal{V}^s|} \cdot \sum_{v^s \in \mathcal{V}^s} entropy(v^s) = -\frac{1}{|\mathcal{V}^s|} \cdot \sum_{v^s \in \mathcal{V}^s} \sum_{v \in v^s} Pr(f_v|v^s) \log Pr(f_v|v^s)$$

Table 3 The results of ANeTS and baselines with $\alpha = 0.7$

Data	ANeTS	VEGAS	CoarseNET	BAGC	CODICIL	METIS
Disney						
$\mathcal{D}$	**0.30**	0.49	0.34	0.82	0.59	0.69
$\mathcal{D}_{\text{str}}$	0.06	0.38	0.02	0.71	0.55	0.76
$\mathcal{D}_{\text{att}}$	0.54	0.60	0.66	0.93	0.63	0.62
Memetracker						
$\mathcal{D}$	**0.34**	0.69	0.41	0.64	0.39	0.67
$\mathcal{D}_{\text{str}}$	0.22	0.86	0.19	0.70	0.23	0.65
$\mathcal{D}_{\text{att}}$	0.46	0.52	0.63	0.58	0.57	0.70
Citeseer						
$\mathcal{D}$	**0.32**	0.78	0.34	0.56	0.43	0.70
$\mathcal{D}_{\text{str}}$	0.00	0.98	0.00	0.25	0.23	0.79
$\mathcal{D}_{\text{att}}$	0.64	0.58	0.70	0.63	0.63	0.62
Facebook						
$\mathcal{D}$	**0.24**	0.48	0.33	0.95	0.71	0.75
$\mathcal{D}_{\text{str}}$	0.00	0.61	0.00	0.97	0.80	0.87
$\mathcal{D}_{\text{att}}$	0.48	0.35	0.66	0.93	0.62	0.63
Google+						
$\mathcal{D}$	**0.18**	0.66	0.38	0.84	0.54	0.81
$\mathcal{D}_{\text{str}}$	0.13	0.99	0.07	0.99	0.58	0.85
$\mathcal{D}_{\text{att}}$	0.23	0.33	0.70	0.69	0.50	0.78
Enron						
$\mathcal{D}$	**0.38**	–	0.51	0.83	×	0.60
$\mathcal{D}_{\text{str}}$	0.36	–	0.26	0.97	×	0.67
$\mathcal{D}_{\text{att}}$	0.40	–	0.76	0.69	×	0.54
PubMed						
$\mathcal{D}$	**0.34**	–	**0.34**	0.80	0.43	0.52
$\mathcal{D}_{\text{str}}$	0.04	–	0.00	0.75	0.23	0.34
$\mathcal{D}_{\text{att}}$	0.64	–	0.69	0.86	0.63	0.70
YouTube						
$\mathcal{D}$	**0.26**	-	0.34	×	×	0.68
$\mathcal{D}_{\text{str}}$	0.03	–	0.02	×	×	0.69
$\mathcal{D}_{\text{att}}$	0.47	–	0.66	×	×	0.68

Bold numbers are winners, and ANeTS performs best in all datasets. '–' means the method does not finish after 14 days. '×' means the method runs out of memory

where $Pr(f_v|v^s)$ is the probability of observing the attribute value of v in the supernode v^s. Note, lower entropy $H_{\mathbf{G^s}}$ implies higher homogeneity. ANeTS gives 7 times lower entropy than the other baselines—detailed results omitted due to lack of space.

Scalability To examine the running time of ANeTS we extract subgraphs with different sizes from *YouTube* ($\sim$ 20M dataset size), and run it. As expected from the

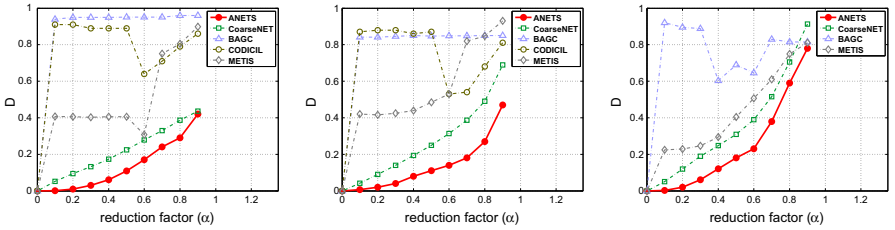

Fig. 3 The $\mathcal{D}(\mathbf{G}, \mathbf{G^s})$ achieved by ANeTS and baselines with different α. ANeTS is best with any α in all datasets. **a** *Facebook*, **b** *Google+*, **c** *Enron*

complexity, we observed that ANeTS scales near-linear w.r.t the size of the DAW-graph $\mathbf{G}$ (see Fig. 4g). Also we got $\sim 9\times$ speedup after parallelizing ANeTS using 10 processors (see Fig. 4h).

6.2 Application 1: topic-aware diffusion

Here we apply ANeTS to speed-up the original TIM task (see the AnT framework in Sect. 5). Our experiments show that we are able to achieve up to $20\times$ speedup on large networks while maintaining the quality of solutions. We can use any off-the-shelf algorithm to solve the TIM problem on original and summarized graphs. We used state-of-the-art online TIM (Chen et al. 2015) algorithm. Finally, as the topic aware propagation probabilities are not available for our datasets, following literature (Chen et al. 2015), we generate a random probability vectors for each edge and we assume there are 10 topics (i.e. $T = 10$) in each dataset.

Effectiveness of AnT We examine the expected influence spread of AnT and TIM across various datasets, number of seed nodes and reduction factors (i.e. α). Figure 4a shows the expected influence spread ratio $\rho = \frac{\sigma_{\mathsf{AnT}}(S|Q)}{\sigma_{\mathsf{TIM}}(S|Q)}$ with $k = 5$ and $\alpha = 0.7$. Figure 4b, c show the influence spread ratio ρ with various number of seed nodes and reduction factors. In *all* the experiments, the influence spread of AnT is within 1% of influence spread of TIM approach. In some cases such as *Enron*, we even perform slightly better. Note that TIM did not finish on *Google+* after **10 days** (AnT finished in around *an hour*). Also, Fig. 4c indicates that even with extremely compact summary (i.e. $\alpha = 0.9$) AnT works as well as TIM.

Speed-ups by AnT Here we show the running time and speed-up of using AnT versus TIM. We define the speed-up as $\tau = \frac{\textit{Running time of } \mathsf{TIM}}{\textit{Running time of } \mathsf{AnT}}$.

W.R.T. graph size Figure 4d shows that τ increases as the size of graph grows. E.g. in *YouTube* we get $7\times$ speedup while the quality of the results is same as the plain TIM algorithm.

W.R.T. number of seeds (k) Figure 4e shows the results for experiments on *YouTube* with $\alpha = 0.7$ and number of seeds varies from 1 to 100. The running time of AnT increases only slightly as k increases, while TIM scales very poorly: with $k = 100$ TIM does not even finish within *10 days*.

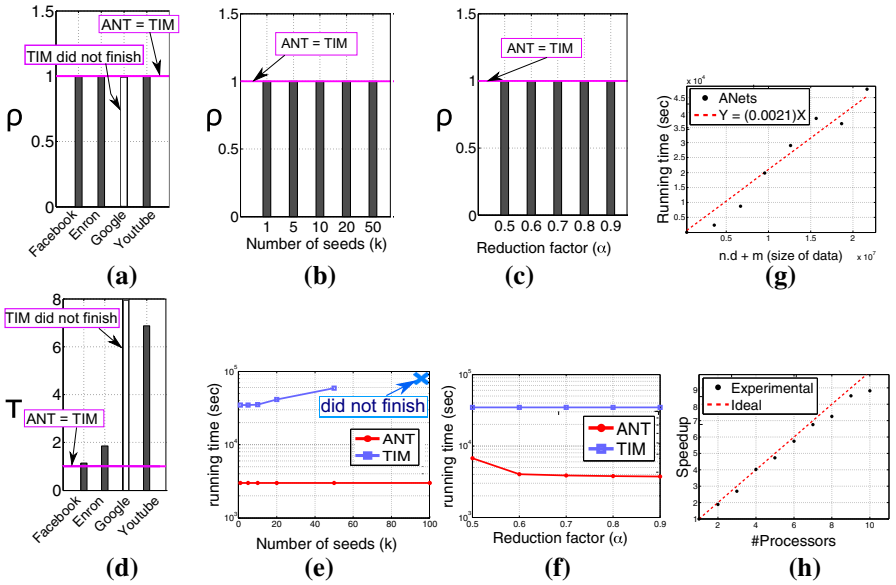

Fig. 4 **a**–**f** Performance of AnT. **a** expected influence spread ratio ρ and **d** speedup τ. **b**, **e** ρ and run time of AnT and TIM versus k. **c**, **f** ρ and run time of AnT and TIM versus α. AnT is significantly faster than TIM, giving similar results. **g**, **h** Efficiency of ANeTS. **g** running time of ANeTS and, **h** parallelization speed-up. ANeTS is *near-linear* in running time and in parallelization speed-up as well

W.R.T. (α) Figure 4f shows that the running time of AnT decreases with more smaller summaries (i.e. higher α).

6.3 Application 2: making sense of graphs and anomaly detection

Here we show how to use ANeTS to understand and explore complex networks and detect anomalies (*Network-traffic*, *Memetracker* and *Portland*).

Network-traffic Figure 1 shows two examples of network-traffic graphs. As explained earlier, it is hard to recognize malicious requests by simply inspecting the original graphs. However, our summaries highlight these anomalous nodes, matching the ground-truth (Zhang et al. 2014) and also showing that they are structurally similar to user inputs—Fig. 1b, d (which makes them hard to find in the first place). Hence our summary helps provide structural evidence for system and network assurance and is useful for human experts cognition, and decision making in cyber security.

Memetracker We summarize the network with $\alpha = 0.97$ to have a small summary graph easy to visualize. *Memetracker* is a blog/News network, and the edges indicate who copied from whom. Attributes are the type of nodes (i.e. blog or news website), the number of posts about memes, and the first time that the node posted anything about memes. The Fig. 5 shows the $\mathbf{G^s}$ of the *Memetracker* dataset. Node colors in each graph represent different attributes of the super-nodes.

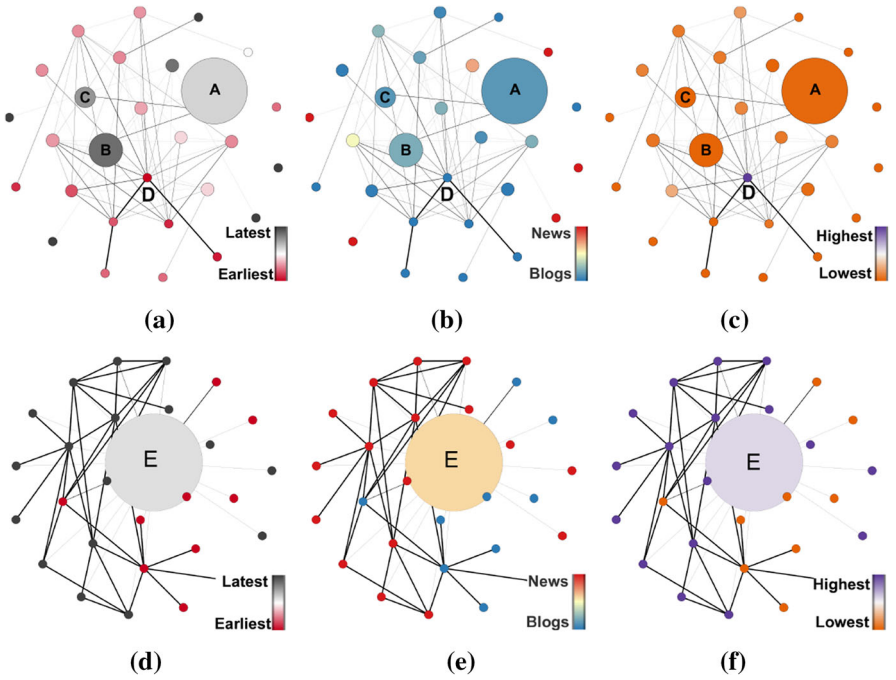

Fig. 5 *Memetracker*—Node colors in each graph represent an attribute of the super-nodes. ANeTS (top) and CoarseNET (bottom) summary: **a**, **d** the average time of infection, **b**, **e** the type of the websites, and **c**, **f** # posts related to the memes. Size of super-nodes $\propto$ the # merged nodes (Color figure online)

ANeTS should ensure that nodes in a super-node are similar w.r.t the influence structure and attributes. Super-node 'A' contains mainstream news media sites such as CNN, BBC, Guardian, etc. It shows that these websites are structurally similar. Also, their behavior in reporting news (i.e. time of report and # posts) is similar. Figure 5a shows that mainstream news media sites usually report memes earlier than their connected blogs/news sites. Also the high degree centrality ($= 16$) of 'A' in the summary graph demonstrates that other nodes copy content from them.

'B' and 'C' are immediate neighbors of super-node 'A'. Super-node 'C' mostly contains news blogs and 'B' has websites and blogs related to sports and entertainment such as ESPN. Figure 5a shows the 'B' and 'C' usually get infected through mainstream news media. According to Fig. 5a, b blogs publish memes very quickly but not many websites follow them and report the news later. Their low degree (~ 2) and low PageRank (~ 0.02) show that many websites specially mainstream news websites do not follow them.

Our summary can also help in anomaly detection. See the *single* purple node 'D'. It is a news website which has the highest number of posts and earliest time of reporting. It is suspicious as it was not merged with the other group of mainstream websites by ANeTS. Turns out that this website belongs to "gopusa.com" which is a spam news website and not affiliated to the Republican National Committee. Our summary

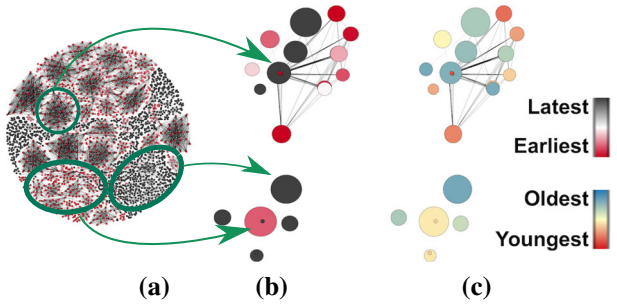

Fig. 6 *Portland*. Node colors represent atts. **a** Original graph, **b** avg. time of infection, **c** avg. age. Size of super-nodes $\propto$ # merged nodes (Color figure online)

clearly shows it as an anomaly. In contrast, CoarseNET does not highlight any of the above patterns as it preserves only the structural properties.

Portland Figure 6 shows the *Portland* summary. It is a people contact network in Portland area: nodes are people and links are their interactions. Attributes are age, location and time of infection of people. We ran a *Targeted* flu-infection scenarios on the dataset (Anderson et al. 1992): the infection starts at a location, and older individuals have higher probability of getting infected through their young neighbors. Colors represent attributes and the location of nodes represents their actual lat-long.

Our summary gives a better insight about how the disease propagates. For example, 'the disease starts from a location infecting young people, and gradually moves to other areas of the network infecting the elderly' (see Fig. 6). These kinds of insights help epidemiologists in understanding and eventually controlling contagious diseases better. Such a pattern cannot be easily observed on the large original network due to the complex interplay between the attributes and network structure. However, our summary can easily highlight them.

7 Discussion and conclusions

We proposed ANeTS, a new unsupervised, scalable and parallelizable algorithm which gets high-quality summaries of attributed influence graphs, in *near-linear* time in practice (in contrast to non-trivial competitors). We carefully design a high-quality deterministic starting point for ANeTS. Next, ANeTS iteratively changes the summary with deterministic decisions until convergence. The summary can be used for many applications: e.g. it speeds-up the TIM task and detects malicious nodes in a network; and it intuitively highlights structurally and attributed homogeneous regions in the network—helping in sense-making of complex network datasets.

Effect of attributes Most real-world graphs are skewed; so typically many nodes will not be structurally important. Hence plain-graph methods such as CoarseNET will likely give unbalanced summary graphs (like in Fig. 5). However, as we showed in the experiments, considering attributes makes a significant difference and gives more semantically meaningful and balanced summaries.

Alternative approaches For this challenging task, ANeTS improves the state-of-the-art in multiple ways: (a) it easily outperforms competitors for the AGS problem giving nodes with higher homogeneity; (b) enables summarization of $\sim 20\times$ larger DAW-graphs (current methods like VEGAS do not even finish in 14 days); and (c) it helps scale existing algorithms for other tasks like TIM $\sim 20\times$ times.

Flexibility ANeTS naturally generalizes both plain-graph summarization and attributed data-clustering in a flexible framework. For example, we can replace the Euclidean distance with any distance metric suitable for K-means style algorithms. Extending our method to a streaming setting (as it is iterative already), and to incorporate overlapping K-Means to give more complex summaries will be interesting.

Future work The ANets framework assumes real values for attributes of nodes. However, we can apply our method to binary attributes as well. It is an interesting future work to extend ANets to more general categorical values. Also, since we are using an iterative approach to get the best summary, there is a possibility to get stuck in local minimums. Even though ANets works pretty well in practice, it is an interesting future work to propose an approach which can avoid the local minimum issue. Finally, it would be interesting to study the effect of other metrics to measure the structural distance. For example, we can use the Laplacian matrix or generalize the structural distance to use other eigenvalues for robustness.

References

Akoglu L, Tong H, Meeder B, Faloutsos C (2012) Pics: parameter-free identification of cohesive subgroups in large attributed graphs. In: Proceedings of the 2012 SIAM international conference on data mining. SIAM, pp 439–450

Anderson RM, May RM, Anderson B (1992) Infectious diseases of humans: dynamics and control, vol 28. Wiley Online Library, Oxford, UK

Barbieri N, Bonchi F, Manco G (2012) Topic-aware social influence propagation models. In: Data mining (ICDM), 2012 IEEE 12th international conference on. IEEE, pp 81–90

Chen W, Wang Y, Yang S (2009) Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 199–208

Chen S, Fan J, Li G, Feng J, Tan K-L, Tang J (2015) Online topic-aware influence maximization. Proc VLDB Endow 8(6):666–677

Dhillon IS, Guan Y, Kulis B (2004) Kernel k-means: spectral clustering and normalized cuts. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 551–556

Fan W, Li J, Wang X, Wu Y (2012) Query preserving graph compression. In: Proceedings of the 2012 ACM SIGMOD international conference on management of data. ACM, pp 157–168

Ghosh A, Boyd S (2006) Growing well-connected graphs. In: Decision and control, 2006 45th IEEE conference on. IEEE, pp 6605–6611

Gomez-Rodriguez M, Leskovec J, Krause A (2012) Inferring networks of diffusion and influence, vol 5. ACM, New York, p 21

Günnemann S, Boden B, Seidl T (2011) DB-CSC: a density-based approach for subspace clustering in graphs with feature vectors. Springer, Berlin, pp 565–580

Jain AK (2010) Data clustering: 50 years beyond k-means. Pattern Recognit Lett 31(8):651–666

Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY (2002) An efficient k-means clustering algorithm: analysis and implementation. IEEE Trans Pattern Anal Mach Intell 24(7):881–892

Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 20(1):359–392

Khan A, Bhowmick SS, Bonchi F (2017) Summarizing static and dynamic big graphs. Proc VLDB Endow 10(12):1981–1984

Kloumann IM, Kleinberg JM (2014) Community membership identification from small seed sets. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 1366–1375

Liu Y, Dighe A, Safavi T, Koutra D (2016) A graph summarization: a survey. arXiv preprint arXiv:1612.04883

Mathioudakis M, Bonchi F, Castillo C, Gionis A, Ukkonen A (2011) Sparsification of influence networks. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 529–537

Navlakha S, Rastogi R, Shrivastava N (2008) Graph summarization with bounded error. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data. ACM, pp 419–432

Perozzi B, Akoglu L, Iglesias Sánchez P, Müller E (2014) Focused clustering and outlier detection in large attributed graphs. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 1346–1355

Prakash BA, Chakrabarti D, Valler NC, Faloutsos M, Faloutsos C (2011) Threshold conditions for arbitrary cascade models on arbitrary networks. ICDM, Vancouver, Canada

Purohit M, Prakash BA, Kang C, Zhang Y, Subrahmanian V (2014) Fast influence-based coarsening for large networks. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 1296–1305

Qu Q, Liu S, Jensen CS, Zhu F, Faloutsos C (2014) Interestingness-driven diffusion process summarization in dynamic networks. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, Berlin, pp 597–613

Ruan Y, Fuhry D, Parthasarathy S (2013) Efficient community detection in large networks using content and links. In: Proceedings of the 22nd international conference on World Wide Web. ACM, pp 1089–1098

Seah B-S, Bhowmick SS, Dewey CF, Yu H (2012) Fuse: a profit maximization approach for functional summarization of biological networks. BMC Bioinform 13(3):S10

Sen P, Namata GM, Bilgic M, Getoor L, Gallagher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93–106

Shi L, Tong H, Tang J, Lin C (2015) Vegas: visual influence graph summarization on citation networks. IEEE Trans Knowl Data Eng 27(12):3417–3431

Tian Y, Hankins RA, Patel JM (2008) Efficient aggregation for graph summarization. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data. ACM, pp 567–580

Toivonen H, Zhou F, Hartikainen A, Hinkka A (2011) Compression of weighted graphs. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 965–973

Wu Y, Zhong Z, Xiong W, Jing N (2014) Graph summarization for attributed graphs. In: Information Science, Electronics and Electrical Engineering (ISEEE), 2014 international conference on, vol 1. IEEE, pp 503–507

Xu Z, Ke Y, Wang Y, Cheng H, Cheng J (2012) A model-based approach to attributed graph clustering. In: Proceedings of the 2012 ACM SIGMOD international conference on management of data. ACM, pp 505–516

Yang J, Leskovec J (2015) Defining and evaluating network communities based on ground-truth. Knowl Inf Syst 42(1):181–213

Yang J, McAuley J, Leskovec J (2013) Community detection in networks with node attributes. In: Data mining (ICDM), 2013 IEEE 13th international conference on. IEEE, pp 1151–1156

Zhang H, Yao DD, Ramakrishnan N (2014) Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery. In: Proceedings of the 9th ACM symposium on information, computer and communications security. ACM, pp 39–50

Zhang H, Sun M, Yao DD, North C (2015) Visualizing traffic causality for analyzing network anomalies. In: Proceedings of the 2015 ACM international workshop on international workshop on security and privacy analytics. ACM, pp 37–42

Zhou Y, Cheng H, Yu JX (2010) Clustering large attributed graphs: an efficient incremental approach. In: Data mining (ICDM), 2010 IEEE 10th international conference on. IEEE, pp 689–698