Graph Summarization Methods and Applications: A Survey

YIKE LIU, TARA SAFAVI, ABHILASH DIGHE, and DANAI KOUTRA, University of Michigan, Ann Arbor

While advances in computing resources have made processing enormous amounts of data possible, human ability to identify patterns in such data has not scaled accordingly. Efficient computational methods for condensing and simplifying data are thus becoming vital for extracting actionable insights. In particular, while data summarization techniques have been studied extensively, only recently has summarizing interconnected data, or *graphs*, become popular. This survey is a structured, comprehensive overview of the state-of-the-art methods for summarizing graph data. We first broach the motivation behind and the challenges of graph summarization. We then categorize summarization approaches by the type of graphs taken as input and further organize each category by core methodology. Finally, we discuss applications of summarization on real-world graphs and conclude by describing some open problems in the field.

CCS Concepts: • **Mathematics of computing** → **Graph algorithms**; • **Information systems** → **Data mining**; **Summarization**; • **Human-centered computing** → *Social network analysis*; • **Theory of computation** → *Unsupervised learning and clustering*; • **Computing methodologies** → *Network science*;

Additional Key Words and Phrases: Graph mining, graph summarization

ACM Reference format:
Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph Summarization Methods and Applications: A Survey. *ACM Comput. Surv.* 51, 3, Article 62 (June 2018), 34 pages.
https://doi.org/10.1145/3186727

1 INTRODUCTION

As technology advances, the amount of data that we generate and our ability to collect and archive such data both increase continuously. Daily activities like social media interaction, web browsing, product and service purchases, itineraries, and wellness sensors generate large amounts of data, the analysis of which can immediately impact our lives. This abundance of generated data and its velocity call for data summarization, one of the main data mining tasks.

Since summarization facilitates the identification of structure and meaning in data, the data mining community has taken a strong interest in the task. Methods for a variety of data types

Y. Liu and T. Safavi contributed equally to this article.

This material was based on work supported in part by the National Science Foundation under grant IIS 1743088, Trove, and the University of Michigan. 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.

Authors' addresses: Y. Liu, T. Safavi, A. Dighe, and D. Koutra, Bob and Betty Beyster Building, 2260 Hayward St, Ann Arbor, MI 48109; emails: {yikeliu, tsafavi, adighe, dkoutra}@umich.edu.

https://doi.org/10.1145/3186727

have been proposed: sequence data and events (Casas-Garriga 2005), itemsets and association rules (Liu et al. 1999; Yan et al. 2005; Ordonez et al. 2006; Mampaey et al. 2011), spatial data (Lin et al. 2003), transactions and multi-modal databases (Wang and Karypis 2004; Chandola and Kumar 2005; Shneiderman 2008; Xiang et al. 2010), data streams and time series (Cormode and Muthukrishnan 2005b; Palpanas et al. 2008), video and surveillance data (Pan et al. 2004; Damnjanovic et al. 2008), and activity on social networks (Lin et al. 2008; Mehmood et al. 2013).

This survey focuses on the summarization of interconnected data (otherwise known as graphs or networks), a problem in graph mining with connections to relational data management and visualization. Graphs are ubiquitous, representing a variety of natural processes as diverse as friendships between people (Wasserman and Galaskiewicz 1994; Backstrom et al. 2006; Devineni et al. 2015), communication patterns (de Melo et al. 2010; Koutra et al. 2013; Yang et al. 2017), and interactions between neurons in the brain (Sporns 2010; Brugere et al. 2016; Safavi et al. 2017).

Graph definitions and examples. Formally, a *plain* graph or network is an abstract data type consisting of a finite set of vertices (nodes) $\mathcal{V}$ and a set of links (edges) $\mathcal{E}$. The latter represent interactions between pairs of vertices. A graph is often represented by its adjacency matrix $\mathbf{A}$, which can be binary, corresponding to whether there exists an interaction between two vertices, or numerical, corresponding to the strength of the connection. We will refer to a graph with numerical or categorical labels (attributes or annotations) for its nodes or edges as a *labeled* graph. A network that changes over time (i.e., nodes/edges get added/deleted) is called *dynamic* or *time-evolving* and is often described by a series of adjacency matrices, one per timestamp. Examples of graphs are social networks, traffic networks, computer networks, phone call or messaging networks, location check-in networks, protein–protein interaction networks, user–product review or purchase networks, and functional or structural brain connectomes, among others.

Graph summarization benefits and applications. Graph summarization has various benefits, which include the following:

— *Reduction of data volume and storage*: Graphs of real-world datasets are often massive. For example, as of August 2017, the Facebook social network had 2 billion users, and more than 100 billion emails were exchanged daily. Summarization techniques produce small summaries that require significantly less storage space than their original counterparts. Graph summarization techniques can decrease the number of I/O operations, reduce communication volume between clusters in a distributed setting, allow loading the summary graph into memory, and facilitate the use of graph visualization tools while avoiding the "hairball" visualization problem.
— *Speedup of graph algorithms and queries*: While a plethora of graph analysis methods exist, many cannot efficiently handle large graphs. Summarization techniques produce smaller graphs that maintain the most salient information from the original graph. The resultant summary graph can be queried, analyzed, and understood more efficiently with existing tools and algorithms.
— *Interactive analysis support*: As the systems side makes advancements in interactive graph analysis, summarization is introduced to handle information extraction and speed up user analysis. The resultant graph summaries make it possible to visualize datasets that are originally too large to load into memory.
— *Noise elimination*: Real graph data are frequently large scale and considerably noisy with many hidden, unobserved, or erroneous links and labels. Such noise hinders analysis by increasing the workload of data processing and hiding the more "important" information. Summarization serves to filter out noise and reveal patterns in the data.

Given its advantages, graph summarization has extensive applications, including clustering (Cilibrasi and Vitányi 2005), classification (Leeuwen van Leeuwen et al. 2006), community detection (Chakrabarti et al. 2004), outlier detection (Smets and Vreeken 2011; Akoglu et al. 2012), pattern set mining (Vreeken et al. 2011), finding sources of infection in large graphs (Prakash et al. 2012), and visualization (Dunne and Shneiderman 2013; Jin and Koutra 2017b), among others.

The problem of graph summarization has been studied algorithmically in the fields of graph mining and data management, while interactive exploration of the data and appropriate display layouts have been studied in visualization. In this survey, we review graph summarization mostly from a methodological perspective, answering how we can algorithmically obtain summaries of graph data. We also give pointers to visual analytics platforms that can consume algorithmic outputs and explore display options.

1.1 Challenges

Overall, the notion of a graph summary is not well defined. A summary is application dependent and can be defined with respect to various goals: It can preserve specific structural patterns, focus on some network entities, preserve the answers to graph queries, or maintain the distributions of graph properties. Overall, graph summarization has five main challenges:

(1) *Data volume*: The main target of graph summarization is to reduce the size of the input graph data so that other analyses can be performed efficiently. At the same time, though, summarization techniques are themselves faced with the challenge of processing large amounts of data. The requirement of efficiency often steers their design toward techniques that scale well with the size of the input graph. Table 1 points to methods that are linear on the size of the input.

(2) *Complexity of data*: Graph operations often cannot be easily partitioned and parallelized because of the many interactions between entities, as well as the complexity of entities themselves. Furthermore, the heterogeneity of nodes and edges continues to increase in real networks (Sun and Han 2012; Koutra et al. 2017). Accordingly, incorporating side information from heterogeneous sources (text, images, etc.) may require highly detailed design (e.g., multi-layer networks (Kivel et al. 2014)) and quantification in algorithms. For example, in social networks, users can chat or share with each other, follow or friend each other, and a single user profile alone contains additional information. Finally, real datasets often contain noise or missing information, which may interfere with the pattern mining process. Sections 3 and 4 review methods for attributed and dynamic networks, which tend to be more complex than methods for plain networks.

(3) *Definition of interestingness*: Summarization involves extracting of important or interesting information. However, the definition of "interesting" is itself subjective, usually requiring both domain knowledge and user preferences. Moreover, the cutoff between "interesting" and "uninteresting" can be difficult to determine in a principled way; usually, it is decided by considering the tradeoffs among time, space, and information preserved in the summary, as well as the complexity of mapping solutions obtained from the summary back onto the original nodes and edges. Each presented graph summarization technique uses different optimization formulations to define the interestingness of a summary.

(4) *Evaluation*: Evaluation of summarization outputs depends on the application domain. From the database perspective, a summary is good if it efficiently supports both global and local queries with high accuracy. In the context of summarizing community information, either community preservation is maximized or reconstruction error is minimized. Compression-based techniques seek to minimize the number of bits needed to describe

Table 1. Qualitative Comparison of All Graph Summarization Techniques Based on the Properties of the Input Graph

		Input Graph				Algorithmic Properties				
	Method	Weighted	Undirect.	Directed	Heterog.	Prm-free	Linear	Technique	Output	Objective
Static Plain Graphs	GraSS (LeFevre and Terzi 2010)	✘	✔	✘	✘	✘	✘	grouping	supergraph	query efficiency
	Weighted Compr. Toivonen et al. (2011)	✔	✔	✔*	✘	✘	✘	grouping	supergraph	compression
	coarseNet (Purohit et al. 2014)	✔	✘	✔	✘	✘	✔*	grouping	supergraph	influence
	l_p-reconstr. Error (Riondato et al. 2014)	✔	✔	✘	✘	✘	✔	grouping	supergraph	query efficiency
	Motifs (Dunne and Shneiderman 2013)	✘	✔	✘	✔	✘	✘	grouping	supergraph	visualization
	CoSum (Zhu et al. 2016)	✔*	✔	✘	✔	✔	✔	grouping	supergraph	entity resolution
	Dedensification (Maccioni and Abadi 2016)	✘	✔*	✔	✔	✔	✔*	(edge) grouping	sparsified graph	query efficiency
	VNM (Buehrer and Chellapilla 2008)	✘	✘	✔	✘	✘	✘	(edge) grouping	sparsified graph	patterns
	MDL Repres. (Navlakha et al. 2008)	✘	✔	✔*	✘	✔	✘	compression	supergraph	compression
	VoG (Koutra et al. 2014a)	✘	✔	✘	✘	✔	✔*	compression	structure list	patterns, visualiz.
	OntoVis (Shen et al. 2006)	✘	✔	✘	✔	✔	✔	simplification	sparsified graph	visualization
	Egocentric Abstr. (Li and Lin 2009)	✘	✘	✔	✔	✘	✘	simplification	sparsified graph	influence
	CSI (Mehmood et al. 2013)	✘	✘	✔	✘	✔	✘	influence	supergraph	influence
	SPINE (Mathioudakis et al. 2011)	✔	✘	✔	✘	✘	✘	influence	sparsified graph	influence
Static Labeled Graphs	S-Node (Raghavan and Garcia-Molina 2003)	✘	✘	✔	✘	✔	✘	grouping	supergraph	query efficiency
	SNAP/k-SNAP (Tian et al. 2008)	✘	✔	✔*	✘	✔	✔	grouping	supergraph	query efficiency
	CANAL (Zhang et al. 2010)	✘	✔	✔*	✘	✔	✘	grouping	supergraph	patterns
	Probabilistic (Hassanlou et al. 2013)	✔	✘	✔	✘	✔	✔	grouping	supergraph	compression
	Query-Pres. (Fan et al. 2012)	✘	✘	✔	✘	✘	✔	grouping	supergraph	query efficiency
	ZKP (Shoaran et al. 2013)	✘	✘	✔	✘	✔	✔	grouping	supergraph	privacy
	Randomized (Chen et al. 2009)	✘	✔	✘	✔	✔	✘	grouping	supergraph	patterns
	d-summaries (Song et al. 2016)	✘	✘	✔	✔	✘	✘	grouping	supergraph	query efficiency
	SUBDUE (Cook and Holder 1994)	✘	✔	✔	✔	✔	✘	compression	supergraph	patterns
	AGSUMMARY (Wu et al. 2014)	✘	✘	✔	✘	✔	✔	compression	supergraph	compression
	LSH-based (Khan et al. 2014)	✘	✘	✔	✘	✘	✔	compression	supergraph	compression
	VEGAS (Shi et al. 2015)	✔*	✘	✔	✘	✘	✔*	influence	supergraph	influence
Dynamic Graphs	NetCondense(Adhikari et al. 2017)	✔	✔	✔	✘	✘	✘	grouping	temporal supergraph	influence
	TCM (Tang et al. 2016)	✔	✔	✔	✘	✘	✔	grouping	supergraph	query efficiency
	TimeCrunch (Shah et al. 2015)	✘	✔	✘	✔	✘	✔	compression	ranked list of temporal structures	temporal patterns, visualization
	OSNet (Qu et al. 2014)	✘	✔	✘	✘	✘	✔	influence	subgraphs of diffusion over time	influence
	Social Activity (Lin et al. 2008)	✘	✔	✘	✔	✘	✘	influence	temporal themes	influence, visualization

(e.g., Weighted, (Un)directed, homogeneous/heterogeneous), their algorithmic properties (i.e., user-defined parameters, complexity linear on the number of edges, core technique, output), and their main objectives. Notation: * for the input means that the algorithm can be extended to that type of input, but details are not in the article; for complexity * indicates sub-linearity.

the input graph or else the number of nodes/edges or the normalized number of bits per edge. Furthermore, evaluations become more complex when more elements, such as visualization and multi-resolution summaries, are involved. In these cases, user studies and qualitative criteria may be employed.

(5) *Change over time*: Graph summaries should evolve over time, since real data are usually dynamic (Leskovec et al. 2005). For instance, social network activity, brain functions, and email communications—all naturally represented as graphs—change with time. How to incorporate the dynamic nature of such data in computation and perform analysis efficiently becomes an essential question. Section 4 reviews methods that treat dynamic graphs as a sequence of static snapshots or streams.

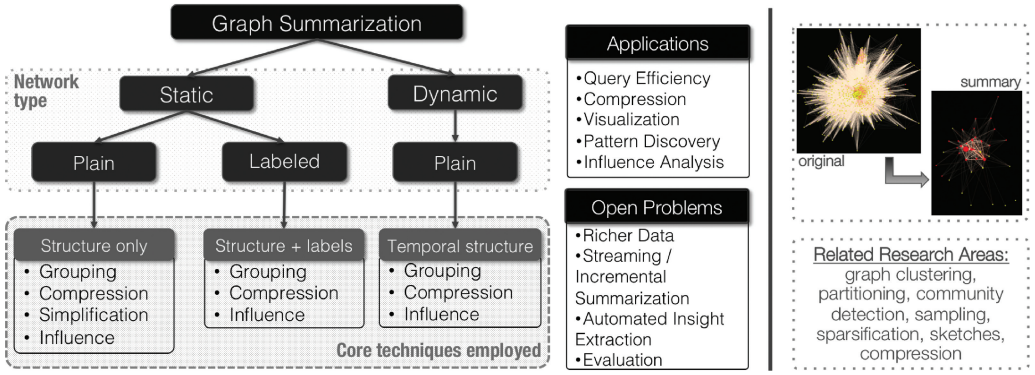

Fig. 1. Overview of our survey. Taxonomy of graph summarization algorithms based on the input type and the core employed technique; alternative approaches; applications; and open problems.

As demonstrated by these challenges, graph summarization is a difficult and multifaceted problem.

1.2 Types of Graph Summaries

In this survey, we categorize graph summarization methods based on the type of data handled and the core techniques employed. Below we give the main types of graph summaries, and Table 1 provides detailed information for each approach:

Input: Static or dynamic. Most summarization methods operate on static networks, leveraging graph structure (links), and, if available, the node/edge attributes. Despite the prevalence of large dynamic networks, only recent research efforts address their efficient summarization. In some cases, static methods are adapted to handle dynamic networks seen as series of static snapshots. In other cases, new methods for graph streams are devised. In this survey, we first categorize summarization methods based on their input type (Figure 1).

Input: Homogeneous or heterogeneous. The most well-studied instance in graph summarization, and graph mining more generally, is the homogeneous graph with one entity and one link type. However, some approaches apply to heterogeneous graphs by treating various types of nodes (e.g., students, instructors) and relations between them (e.g., teacher, friends, classmates) differently. These methods tend to be more complex but also more expressive.

Core technique. Across the literature, graph summarization methods employ a set of core techniques:

— *Grouping or aggregation based*: This is the most popular technique. Some *node-grouping* methods recursively aggregate nodes into "supernodes" based on an application-dependent optimization function, which can be based on structure and/or attributes. Others employ existing clustering techniques and map each densely connected cluster to a supernode. *Edge-grouping* methods aggregate edges into compressor or virtual nodes.
— *Bit compression based*: This approach, a common technique in data summarization, minimizes the number of bits needed to describe the input graph via its summary. Some methods are lossless and can perfectly reconstruct the original graph from the summary. Others are lossy, compromising recovery accuracy for space savings.

— *Simplification or sparsification based*: These methods streamline an input graph by removing less "important" nodes or edges, resulting in a sparsified graph.
— *Influence based*: These approaches aim to discover a high-level description of the influence propagation in large-scale graphs. Techniques in this category formulate the summarization problem as an optimization process in which some quantity related to information influence is maintained.

Output: Summary type. The output of a summarization approach can be (i) a supergraph, which consists of supernodes or collections of original nodes, and superedges between them; (ii) a sparsified graph, which has fewer nodes and/or edges than the original network; or (iii) a list of (static or temporal) structures or influence propagations, which are seen independently instead of in the form of a single summary graph. Moreover, the summary can be (a) flat, with nodes simply grouped into supernodes, or (b) hierarchical, with multiple levels of abstraction.

Output: Non-overlapping or overlapping nodes. In its simplest form, a summary is non-overlapping: Each original node belongs only to one summary element (e.g., supernode, subgraph). Overlapping summaries, where a node may belong to multiple elements, can capture complex inherent data relationships but may also complicate intepretation and visualization.

Main objective. The key objectives of graph summarization include query efficiency and approximate computations, compression and data size reduction, static or temporal pattern discovery, visualization and interactive large-scale visual analytics, influence analysis and understanding, entity resolution, and privacy preservation.

1.3 Differences from Prior Surveys

Previous work on surveying the graph summarization literature is scarce. You et al. (2013) present some summarization algorithms for *static* graphs, focusing mostly on grouping- and compression-based methods. The tutorial by Lin et al. (2013) provides more specific categorization and descriptions of ongoing work but again only addresses static graph summarization. By contrast, we review a wide set of proposed methodologies for both static *and* dynamic graph summarization. Specifically, in this survey:

(1) We create a taxonomy (Figure 1) on the three main instances of the graph summarization problem: for plain static graphs (Section 2), for static graphs with additional side information or labels (Section 3), and for (plain) graphs that evolve over time (Section 4). Within each instance of the problem, we present key algorithmic ideas and methodologies used to solve it.
(2) We highlight methodological properties that are useful to researchers and practitioners, such as input/output data types and end goal (for example, compression vs. visualization), and present them concisely in Table 1.
(3) We give connections between methods of graph summarization and related fields that, while not directly supporting graph summarization, have potential in summarization tasks. These fields include compression, sparsification, and clustering and community detection.
(4) We review real-world applications of graph summarization and identify open problems and opportunities for future research (Sections 5 and 6).

2 STATIC GRAPH SUMMARIZATION: PLAIN NETWORKS

Most work in static graph summarization focuses solely on graph structure without side information or labels. At a high level, the problem of summarization or aggregation or coarsening of static,

plain graphs is described as

Problem 1. Summarization of Static, Plain Graphs.
Given a static graph G without any side information, or its adjacency matrix $\mathbf{A}$,
Find a summary graph or a set of structures or a compressed data structure to concisely describe the given graph.

The first block of Table 1 qualitatively compares and explicitly characterizes static graph summarization methods for plain networks. Here, we review these methods by organizing them into categories based on the core methodology that they employ for the summarization task. When applicable, we first give the high-level idea per method type and then describe the corresponding technical details.

2.1 Grouping-Based Methods

Grouping-based methods are among the most popular techniques for summarization. We distinguish grouping-based graph summarization methods into two main categories: (i) node-grouping and (ii) edge-grouping. In Section 2.2, we discuss methods that use bit-level compression as their primary summarization technique and grouping as a complementary technique.

2.1.1 Node-Grouping Methods. Some approaches employ existing clustering techniques to find clusters that then map to supernodes. Others recursively aggregate nodes into supernodes, connected via superedges, based on an application-dependent optimization function.

Node clustering-based methods. Although node grouping and clustering are related in that they result in collections of nodes, they have different goals. In the context of summarization, node grouping is performed so that the resultant graph summary has specific properties, e.g., query-specific properties or maintenance of edge weights. On the other hand, clustering or partitioning usually targets the minimization of cross-cluster edges or a variant thereof, without the end goal of producing a graph summary. Moreover, unlike role mining (Henderson et al. 2011, 2012; Gilpin et al. 2013) or structural equivalence (Peleg and Schäffer 1989), which seek to identify "functions" of nodes (e.g., bridge or spoke nodes) and find role memberships, summarization methods seek to group nodes that have not only structural similarities but are also connected or close to each other in the network and thus can be replaced with a supernode.

Although the goal of clustering is not graph summarization, the outputs of clustering algorithms can be easily converted to non-application-specific summaries. In a nutshell, a small representation of the input graph can be obtained by (i) mapping all the nodes that belong to the same cluster/community to a supernode and (ii) linking them with superedges with weight equal to the sum of the cross-cluster edges or else the sum of the weights of the original edges (Newman and Girvan 2004; Yang and Leskovec 2013; Low et al. 2012). Although the clustering output can be viewed as a summary graph, a fundamental difference from tailored summarization techniques is that the latter groups nodes that are linked to the rest of the graph in a similar way, while clustering methods simply group densely connected nodes. There exist comprehensive introductions to clustering techniques (Leskovec et al. 2014; Aggarwal 2015) and work on clustering or community detection methods (Aggarwal and Wang 2010), so we do not cover them in this survey. Among the most popular partitioning methods are Graclus (Dhillon et al. 2005), spectral partitioning (Alpert et al. 1999), and METIS (Karypis and Kumar 1999). Although METIS is a well-known partitioning approach that finds "hard" node memberships, it constructs a series of graph "summaries" by iteratively finding the maximal graph matching and merging nodes that are incident to an edge of the matching. The bisection result on the most coarsened graph is then projected backwards to

the original graph. Via this process, it is possible to obtain a compact, hierarchical representation of the original graph, which resembles other node-grouping summarization methods.

Node aggregation-based methods. One representative algorithm of hierarchical clustering-based node grouping is GraSS (LeFevre and Terzi 2010), which targets accurate query handling. This summarization method supports queries on the adjacency between two nodes, as well as the degree and the eigenvector centrality of a node. The graph summaries are generated by greedily grouping nodes such that the normalized reconstructed error, $\frac{1}{|\mathcal{V}|^2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}}|\bar{A}(i,j)-A(i,j)|$, is minimized—$\mathbf{A}$ is the original adjacency matrix of the graph and $\bar{\mathbf{A}}$ is the real-valued approximate adjacency matrix, each entry of which intuitively represents the probability of the corresponding edge existing in the original graph given the summary. The resulting summaries are represented as a group of vertex sets with information about the number of edges within and between clusters. These sets are used to generate a probabilistic approximate adjacency matrix on which incoming queries are computed. For example, if many edges cross vertex sets A and B, then it is likely that a node in A is connected to a node in B. In another variant, GraSS leverages Minimum Description Length (MDL) to automatically find the optimal number of supernodes in the summary.

While GraSS does not guarantee output quality, Riondato et al. (2014) propose a method of generating supernodes and superedges with guarantees. Here, the objective is to find a supergraph that minimizes the l_p-reconstruction error, or the p-norm of $\mathbf{A}-\bar{\mathbf{A}}$, as opposed to the normalized reconstruction error in GraSS, given a number of supernodes k. The proposed approach, which uses sketching, sampling, and approximate partitioning, is the first polynomial-time approximation algorithm of its kind with runtime $O(|\mathcal{E}|+|\mathcal{V}|\cdot k)$. This method targets efficiency for the same types of queries as GraSS, as well as triangle and subgraph counting queries.

Toivonen et al. (2011) focus on compressing graphs with edge weights, proposing to merge nodes with similar relationships to other entities (structurally equivalent nodes) such that approximation error is minimized and compression is maximized. In merging nodes to obtain a compressed graph, the algorithm maintains either edge weights or strengths of connections of up to a certain number of hops. Specifically, in the simplest version of the solution, each superedge is assigned the mean weight of all edges it represents. In the generalized version, the best path between any two nodes is "approximately equally good" in the compressed graph and original graphs, but the paths do not have to be the same. The definition of path "goodness" is data and application dependent. For example, the path quality can be defined as the maximum flow through the path for a flow graph or the probability that the path exists for a probabilistic or uncertain graph.

The methods described above all minimize some version of the approximation or reconstruction error. Other node-grouping approaches seek summaries that maintain specific properties of the original graph, a goal that resembles the target of graph sparsification methods (Spielman and Srivastava 2011; Hübler et al. 2008). One example is diffusive properties related to the spectrum of the graph, and specifically its first eigenvalue λ_1 (Purohit et al. 2014), which are crucial in diffusion and propagation processes like epidemiology and viral marketing. In this case, the summarization problem is formulated as a minimization of the change in the first eigenvalue between the adjacency matrices of the summary and the original graph. For efficiency, the method repeatedly merges pairs of *adjacent* nodes, and uses a closed form to evaluate the change in λ_1, derived using matrix perturbation theory. Node pairs are merged in increasing order of change in λ_1—the light edges with small "edge scores" in step 1 of Figure 2 are good candidates for merging—and the merging process stops when the user-specified number of nodes is achieved. At every step, edges are reweighted so that λ_1 is maintained (step 2 in Figure 2). The temporal extension of this approach is discussed in Section 4.

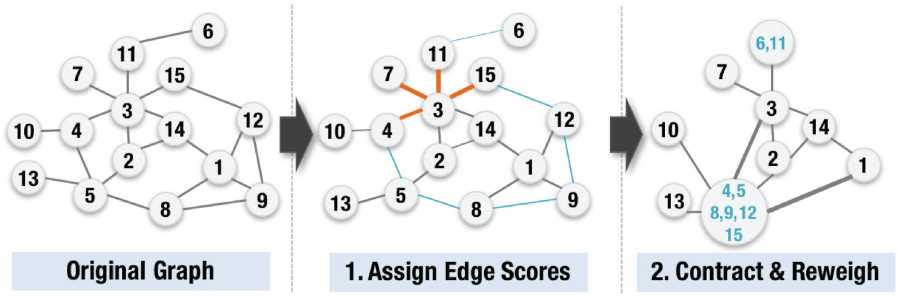

Fig. 2. Overview of coarseNet (Purohit et al. 2014). All the edges in the original graph are weighted equally. In step 1, edges with small width result in small changes in λ_1, while heavy edges result in big changes and are not good candidates for contraction. In step 2, the edge width depicts the new edge weight after obtaining the coarsened network.

In the visualization domain, Dunne and Shneiderman (2013) introduce motif simplification to enhance network visualization. Motif simplification replaces common links and common subgraphs, like stars and cliques, with compact glyphs to help visualize and simplify the complex relationships between entities and attributes. This approach uses exact pattern discovery algorithms to identify patterns and subgraphs, replacing these with glyphs to result in a less cluttered network display. We give an example in Section 5.2.

Beyond the end goal of summarization itself, node grouping can be applied to many graph-based tasks. CoSum (Zhu et al. 2016) involves summarization on k-partite heterogeneous graphs to improve record linkage between datasets, otherwise known as entity resolution. CoSum transforms an input k-type graph into another k-type summary graph composed of supernodes and superedges, using links between different types to improve the accuracy of entity resolution. The algorithm jointly condenses vertices into a supernode such that each supernode consists of nodes of the same type with high similarity and creates superedges that connect supernodes according to the original links between their constituent nodes. The resultant summary achieves better performance in entity resolution than generic approaches, especially in datasets with missing values and one-to-many or many-to-many relations.

2.1.2 Edge-Grouping Methods. Unlike node-grouping methods that group nodes into supernodes, edge-grouping methods aggregate edges into *compressor* or *virtual nodes* to reduce the number of edges in a graph in either a lossless or lossy way. Note that in this section, "compression" does not refer to bit-level optimization, as in the following section but rather to the process of replacing a set of edges with a node.

Graph Dedensification (Maccioni and Abadi 2016) is an edge-grouping method that compresses neighborhoods around high-degree nodes, accelerating query processing and enabling direct operations on the compressed graph. Following the assumption that high-degree nodes are surrounded by redundant information that can be synthesized and eliminated, Maccioni and Abadi (2016) introduce "compressor nodes," which represent common connections high-degree nodes. To provide global guarantees and reduce the scope of compressor handling during query processing, dedensification only occurs when every node has at most one outgoing edge to a compressor node, and every high-degree node has incoming edges coming only from a compressor node. These guarantees are then used to create query processing algorithms that enable direct pattern matching queries on the compressed graph.

Similar approaches include the "connector" motif in visualization-based summarization (Dunne and Shneiderman 2013) discussed in Section 2.1.1 and Virtual Node Mining (VNM) (Buehrer and

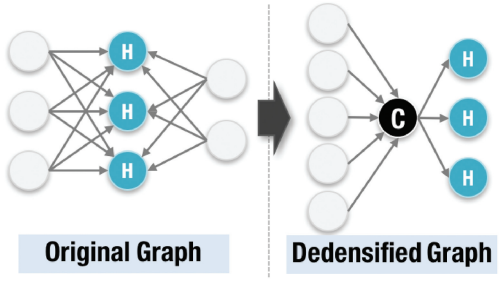

Fig. 3. Example of graph dedensification (Maccioni and Abadi 2016). Many edges are removed after the addition of the compressor node C, which connects to the high-degree nodes H.

Chellapilla 2008), which is used as a lossy compression scheme for the Web graph to accommodate community-related queries and other random access algorithms on link servers. Like SUBDUE (Cook and Holder 1994) (Section 3.2), VNM uses a frequent mining approach to extract meaningful connectivity formations by casting the outlinks/inlinks of each vertex as a transaction/itemset. Then, for each recurring pattern, it removes the links from its vertices and generates a new vertex (virtual node) in the graph, which is added as an outlink. The process may be viewed exactly like graph dedensification (Figure 3), although dedensification provides exact answers due to its losslessness and does not suffer from the space/time tradeoff of graph indexing.

2.2 Bit Compression-Based Methods

Bit compression is a common technique in data mining. In graph summarization, the goal of these approaches is to minimize the number of bits needed to describe the input graph, where the summary consists of a model for the input graph and its unmodeled parts. The graph summary or model is significantly smaller than the original graph and often reveals various structural patterns, like bipartite subgraphs, that enhance understanding of the original graph structure. As mentioned in the previous section, some of these approaches primarily use compression and secondary grouping techniques. However, some others aim solely to compress a given graph without necessarily creating a graph summary or finding comprehensible graph structures.

Here we focus mostly on the former approaches, which often formulate summarization as a model selection task. These works employ the two-part Minimum Description Length (MDL) code, the goal of which is to minimize the description of the given graph G and the model class M in terms of bits:

$$minL(G, M) = L(M) + L(G|M), \tag{1}$$

which is given as the description length of the model, $L(M)$, and the description length of the graph given the model (i.e., the errors or unmodeled parts with respect to the model). For completeness, we also present some graph compression methods that can be adapted to summarization, although not originally designed for that purpose.

Relying on this two-part MDL representation, Navlakha et al. (2008) introduce an approach to summarize graphs with bounded error. This representation, obtained by aggregating nodes in the summary generation, consists of a graph summary S and a set of corrections C (Figure 4). The summary is an aggregate graph in which each node corresponds to a set of nodes in G, and each edge represents the edges between all pairs of nodes in the two sets. The correction term specifies the list of edge-corrections that must be applied to the summary to exactly recreate G. The cost of a representation, R, is the sum of the storage costs of both S and C: $cost(R) = |E_S| + |C|$, where E_S is

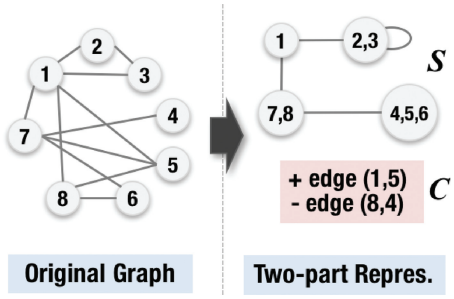

Fig. 4. Two-part MDL representation (Navlakha et al. 2008). Graph summary S and corrections C. Since S does not capture the edge (1,5) properly, it is added in C. Similarly, the summary "captures" edge (8,4), which is missing in the original graph, so it is removed in C.

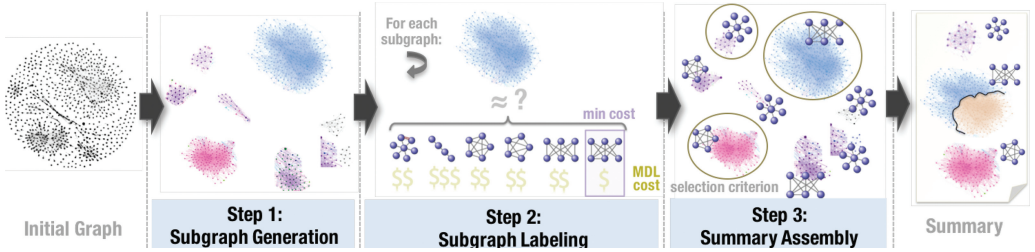

Fig. 5. VoG (Koutra et al. 2014a). Overview of vocabulary-based graph summarization.

the set of superedges in S. The MDL-based graph summary is found by aggregating groups of nodes (thereby falling also into the grouping-based summarization category) as long as they decrease the MDL cost of the graph. To this end, a simple but costly greedy heuristic iteratively combines node pairs that give the maximum cost reduction into supernodes. To reduce the complexity to cubic on the average degree of the graph, a randomized algorithm randomly picks a node and merges it with the best node in its 2-hop neighborhood. This formulation also supports lossy compression with bounded reconstruction error to achieve even higher space savings. This summarization approach gives up to two times more compact summaries than graph compression (Boldi and Vigna 2004) and clustering (Dhillon et al. 2005) methods.

Similarly to Navlakha et al. (2008), Ahnert (2013) introduces a biological application for the discovery of dominant relationship patterns in transcription networks, such as the networks of *Saccharomyces cerevisiae* and *Escherichia coli*. In biology, the terms "power graph" and "power nodes/edges" are used to refer to what we call supergraphs and supernodes/edges. In this application, most supernodes are shown to have functional meaning, and the superedges signify large-scale functional relationships between different subsystems of transcription networks.

Addressing an information-theoretic optimization problem also based on MDL, VoG (Koutra et al. 2014a; Koutra and Faloutsos 2017), or *vocabulary-based summarization of graphs*, succinctly describes large-scale graphs with a few possibly overlapping, easily understood structures encoded in the model M. The graph summary is given in terms of a predefined "vocabulary" of structures that goes beyond the simple rectangles that most summarization and clustering methods find, identifying cliques and near-cliques, stars, chains, and (near-) bipartite cores. VoG is modular (Figure 5): (i) it first performs graph clustering by adapting the node reordering method SlashBurn (Lim et al. 2014) to extract ego-networks and other disconnected components; (ii) it labels

the extracted subgraphs with the appropriate structures in the assumed vocabulary (i.e., cliques and near-cliques, stars, chains, and full or near-bipartite cores) using MDL as a model selection criterion; and, finally, (iii) it creates a summary by employing heuristics that choose only the subgraphs that minimize the total encoding cost of the graph, $L(G, M)$, as it is defined in Equation (1). Some of the exact structures in the vocabulary are part of the motif simplification scheme (Dunne and Shneiderman 2013) (Section 2.1.1), but VoG is distinct in that it allows for near-structures that appear often in real-world graphs and uses MDL for summarization. Likewise, VoG and Navlakha et al. (2008)'s MDL representation are similar in that they use MDL for summarization, but the latter is confined to summarizing a graph in terms of non-overlapping cliques and bipartite cores, while VoG supports a more diverse set of structures or vocabulary. Moreover, it is possible to expand the vocabulary to address the needs of specific applications or domains. Extensions of VoG (Liu et al. 2015) have been applied to empirically evaluate the summarization power of various graph clustering methods, such as METIS (Karypis and Kumar 1999). Similarly to VoG, Miettinen and Vreeken (2011) and Miettinen and Vreeken (2014) discuss MDL for Boolean matrix factorization, which can be viewed as a summary in terms of possibly overlapping full cliques in directed graphs.

Connections to graph compression. Graph summarization and compression are related. Graph summarization methods leverage compression to find a smaller representation of the input graph, *while discovering structural patterns*. In these cases, although compression is the means, finding the absolutely smallest representation of the graph is *not* the end goal. The patterns that are being unearthed during the process may lead to suboptimal compression. However,, in graph compression works, the goal is to compress the input graph as much as possible to minimize storage space, irrespective of patterns.

Since compression and summarization are distinct fields, we only give a few fundamental methods in the former, including: the "Eulerian data structure" to handle neighbor queries in social networks (Maserrat and Pei 2010) and extensions of this work to community-preserving compression (Maserrat and Pei 2012); node reordering techniques, such as zip block encoding in Gbase (Kang et al. 2011), bipartite minimum logarithmic arrangement (Dhulipala et al. 2016) for inverted indices, and techniques for real graphs with power-law degree distributions (Lim et al. 2014); edge reordering techniques (Goonetilleke et al. 2017); compression of web graphs using lexicographic localities (Boldi and Vigna 2004); extensions to social networks (Grabowski and Bieniecki 2014; Chierichetti et al. 2009); breadth-first search-based approaches (Apostolico and Drovandi 2009); lossy edge encoding per triangle (Feng et al. 2013); weighted graph compression to maintain edge weights up to a certain number of hops (Toivonen et al. 2011); provably optimal compression of Erdös-Rényi random graphs using structural entropy (SZIP) (Choi and Szpankowski 2012); and minimal probabilistic tile cover mining (Liu and Chen 2016) that has applications to binary matrices and bipartite graphs.

2.3 Simplification-Based Methods

Simplification-based summarization methods streamline the original graph by removing less "important" nodes or edges, resulting in a sparsified graph. As opposed to supergraphs, here the summary graph consists of a subset of the original nodes and/or edges. In addition to simplification-based summarization methods, some existing graph algorithms have the potential for simplification-based summarization, such as sparsification, sampling, and sketching.

A representative work on *node* simplification-based summarization techniques is OntoVis (Shen et al. 2006), a visual analytical tool that relies on node filtering for the purpose of understanding large, heterogeneous social networks in which nodes and links respectively represent different

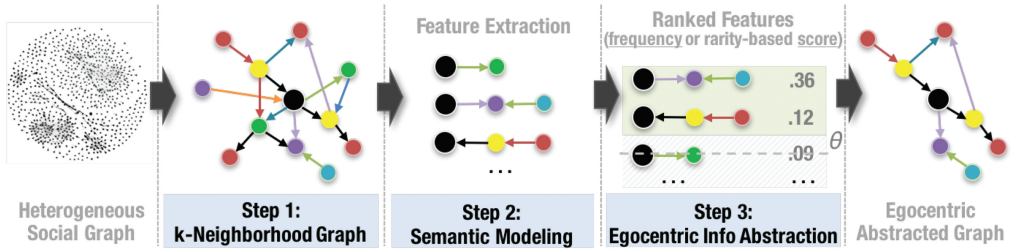

Fig. 6. Overview of egocentric abstraction (Li and Lin 2009). The features are ranked by frequency (score) in step 2. Depending on the policy, only the frequent or rare features are used. In the example, the abstraction graph is based on frequent features above a threshold θ.

concepts and relations. OntoVis uses information in the ontology that relates nodes and edges, such as the degree of nodes of specific type, to semantically prune the network. OntoVis supports semantic abstraction, structural abstraction, and importance filtering. In semantic abstraction, the user constructs a derived graph from the original graph by including only nodes whose types are selected from the original ontology graph. For example, in a terrorism network, selection of the node type "terrorist organization" results in a semantic abstraction of different terrorist organizations. Structural abstraction simplifies the graph while preserving the essential structure of the entire network, for example, by removing one-degree nodes and duplicate paths. Importance filtering makes use of statistical measures like node degree for evaluating connectivity and relevance between node types.

Targeting the same type of graph as OntoVis, Li and Lin (2009) propose a four-step unsupervised algorithm for egocentric information abstraction of heterogeneous social networks using edge, instead of node, filtering (Figure 6). First, during the semantic modeling step, features (or else linear combinations of relations or path-based patterns) are automatically selected and extracted according to the surrounding network substructure (k-hop neighborhoods). Second, the statistical dependency is measured between the features per ego node. Third, during the egocentric information abstraction step, irrelevant information is removed by applying distilling criteria, such as keeping the most frequent or rare features. Finally, in the fourth step, an egocentric abstracted graph is constructed incrementally on the remaining features, allowing the user to visualize the smaller resulting graph.

Connections to graph sampling, sparsification, and sketches. A complementary approach toward "compressing" a graph involves sampling nodes or edges from it (Hübler et al. 2008; Batson et al. 2013). Note, though, that sampling focuses more on obtaining sparse subgraphs that can be used to approximate properties of the original graph (degree distribution, size distribution of connected components, diameter, or community structure (Maiya and Berger-Wolf 2010)) and less on identifying patterns that collectively summarize the input graph to enhance user understanding.

Various sampling techniques have been studied (Mathioudakis et al. 2011; Ahmed et al. 2013), and a comprehensive tutorial on graph sampling was presented at KDD (Hasan et al. 2013). Sampling techniques include sampling nodes according to their in- or out-degree, PageRank, or substructures, such as spanning trees; as well as sampling edges uniformly or according to their weights or their effective resistance (Spielman and Srivastava 2011) to maintain the graph spectrum up to some multiplicative error or to maintain node reachability (transitive reduction (Aho et al. 1972)). Although sampling has the potential to allow better visualization (Rafiei and Curial 2005) and approximate specific queries with theoretical guarantees, it cannot detect graph structures, often operates on individual nodes/edges instead of collective patterns, and may need

additional processing to make sense of the sample. Related to the goal of maintaining specific graph properties is the k-spanner (Peleg and Schäffer 1989), which is the sparsest subgraph in which the distance between pairs of nodes is at most k times the distance in the initial graph. A common category of the problem is the tree k-spanner, which approximates the original graph with a tree that satisfies the distance property. Finding a k-spanner is NP-hard except for the case of $k = 2$, which can be solved in $O(|\mathcal{E}| + |\mathcal{V}|)$ time.

Graph sketches (Ahn et al. 2012; Liberty 2013; Ghashami et al. 2016), or data synopses obtained by applying linear projections, are also relevant. Graph sketching can be viewed as linear dimensionality reduction, where the linearity of sketches makes them applicable to the analysis of streaming graphs with node and edge additions and deletions and distributed settings, such as MapReduce (Dean and Ghemawat 2004).

2.4 Influence-Based Methods

Influence-based methods seek to find a compact, high-level description of the influence dynamics in large-scale graphs to understand the patterns of influence propagation at a global level. Usually such methods formulate graph summarization as an optimization process in which some quantity related to information influence is maintained. These summarization methods are scarce and have been mostly applied on social graphs, where important influence-related questions arise.

Community-level Social Influence (CSI) (Mehmood et al. 2013) is a representative work that focuses on summarizing social networks via information propagation and social influence analysis. Like some other graph summarization techniques, CSI relies on existing clustering approaches: It detects a set of communities using METIS (Karypis and Kumar 1999) and then finds their reciprocal influence by extending the popular Independent Cascade model (Kempe et al. 2003) to communities instead of individual nodes. To balance between data fit and model complexity, CSI uses MDL and Bayesian Information Criterion (BIC) approaches to select the number of communities for the graph model. Unlike influence propagation approaches that find representative cascades for information diffusion, CSI leads to a compact representation of the input network where the nodes correspond to communities and the directed edges represent influence relationships. Note that the output of CSI is different from grouping-based summarization techniques in which the superedges simply represent aggregate connections between the adjacent supernodes. SPINE, an alternative to CSI (Mathioudakis et al. 2011), sparsifies social networks to only keep the edges that "explain" the information propagation—those that maximize the likelihood of the observed data. This problem is shown to be NP-hard to approximate within any multiplicative factor. Inspired by the idea of decomposing sparsification into a number of subproblems equal to the nodes in the network, SPINE is a greedy algorithm that achieves efficiency with practically little compromise in quality. Unlike CSI, it simply eliminates original edges and does not group nodes into communities or supernodes.

2.5 Other Types of Graph Summaries

Although not our main focus, we briefly present methods that represent a network (i) visually with a small set of anomalous patterns, distribution plots of graph properties, or carefully selected nodes or (ii) with latent representations.

Visualization-based systems. Various graph visualization platforms for pattern identification exist. For example, Apolo (Chau et al. 2011) routes attention by visualizing the neighborhoods of a few user-selected seed nodes, which can be interactively explored. A follow-up anomaly detection system, OPAvion (Akoglu* et al. 2012), mines graph features using the Hadoop-based graph mining framework Pegasus (Kang et al. 2009), spots anomalies by employing OddBall (Akoglu et al. 2010) for mining distributions of egonet-related features (e.g., number of nodes vs. edges), and

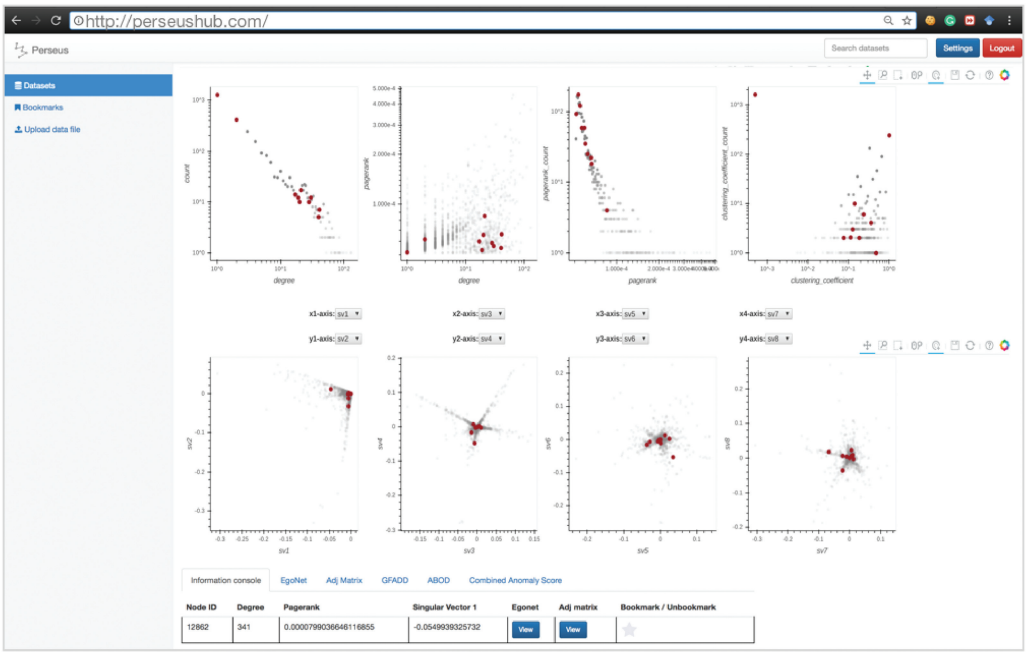

Fig. 7. The front-end of Perseus-Hub, with linked plots for graph properties. The annotated red points correspond to anomalies found during offline pre-processing.

interactively visualizes the anomalous nodes via Apolo. Finally, the large-scale system Perseus (Koutra et al. 2015; Jin et al. 2017) enables comprehensive graph analysis by supporting the coupled summarization of graph properties (computed on Hadoop or Spark) and structures, guiding attention to outliers, and allowing the user to interactively explore normal and anomalous node behaviors in distribution plots and ego-network representations (Figure 7). Other visualization-based methods include scaled density plots to visualize scatter plots (Shneiderman 2008), random and density sampling (Bertini and Santucci 2004) for datasets with thousands of points, and rescaled visualization of spy, distribution, and correlation plots of massive graphs (Kang et al. 2014).

Visualization-based graph summarization is also related to visual graph analytics in that summaries of graphs can support interactive visualization. However, the traditional focuses of visual graph analytics, such as the layout of the data displayed and new visualization or user interaction techniques, differ from the typical goals of graph summarization. Widely used visualization tools, such as Gephi (Bastian et al. 2009), Cytoscape (Shannon et al. 2003), and the Javascript D3 library (Bostock et al. 2011), support interactive exploration of networks and operations such as spatializing, filtering, and clustering. Although these platforms work well on small and medium-sized graphs, they cannot render large-scale networks with many thousands or millions of nodes, or else they are compromised by high latency. These tools can benefit from graph summarization methods that result in smaller network representations or patterns thereof, which can be displayed more easily.

Domain-specific summaries. Beyond visualization, Jin and Koutra (2017a) propose an *optimization* problem for summarizing a graph in terms of representative, domain-specific graph properties. The summaries are required to be concise, diverse, domain-specific, interpretable, and fast to compute. This is the first work to target domain-specific summarization by automatically leveraging

the knowledge encoded in multiple networks from a specific domain, like social science or neuroscience. Although it is related to visualization-based systems that support coupled summarization of graph properties (e.g., PERSEUS (Koutra et al. 2015; Jin et al. 2017), described above), this method automates the selection of the graph properties to be included in the graph "summary" based on the domain from which the data comes.

Latent representations. A variety of methods obtain low-dimensional representations of a network in a latent space. For instance, matrix factorization methods like SVD, CUR (Drineas et al. 2006), and CMD (Sun et al. 2007) all lead to low-rank approximations of an adjacency matrix, which can be viewed as sparsified approximate "summaries" of the original graph. Recent interest in deep learning has lead to novel *node* representation learning techniques (e.g., Perozzi et al. (2014), Grover and Leskovec (2016), Wang et al. (2016), Tang et al. (2015), and Ribeiro et al. (2017)), but these methods present nodes as low-dimensional vectors instead of finding a compact graphical representation of the whole network, which is the goal of summarization.

3 STATIC GRAPH SUMMARIZATION: LABELED NETWORKS

So far we have reviewed summarization methods that use the structural properties of static graphs without additional information like node and edge attributes. However, many real graphs are annotated, labeled, or attributed. For example, in a social network, a typical node representing a user is associated with information about age, gender, and location; transportation graphs may have information about the capacity of streets (edges) and the maximum speed per street; forums like Quora, which can be interpreted as networks of questions and answers, have comments, upvotes, and downvotes. A general definition of graph summarization for static, labeled graphs is given as follows:

PROBLEM 2. Summarization of Static, Labeled Graphs.
Given a static graph G and side information, such as node attributes,
Find a labeled summary graph or a set of labeled structures to **concisely describe** the given graph.

Overall, the main challenge in summarizing labeled graphs is the efficient combination of two different types of data: structural connections and attributes. Currently, most existing works focus on node attributes alone, although other types of side information are certainly of interest in summarization. For instance, joint summarization of multimodal data—including graphs, text, images, and streaming data—has various applications. However, due to the challenges of multimodal analysis, these methods are underexplored in the literature.

The second block of Table 1 provides qualitative comparisons and explicit characterizations of static graph summarization methods for labeled graphs, which we review next by classifying them based on their core technical methodology. The overview of this section is included in Figure 1.

3.1 Grouping-Based Methods

Grouping-based methods aggregate nodes into supernodes connected by superedges based on both structural properties and node attributes. Grouped nodes are usually structurally close in the graph and share similar attribute values.

As discussed with plain graphs, here attributed clustering or community detection (Zhou et al. 2009; Yang et al. 2013; Xu et al. 2012) methods do not perform summarization but could be leveraged by summarization approaches to obtain compact representations of graphs with attributes. One fundamental difference between summarization and clustering is that the former finds coherent

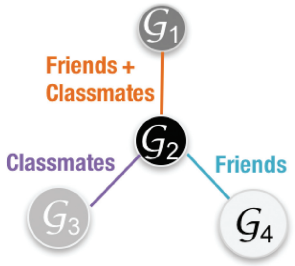

Fig. 8. SNAP summary (Tian et al. 2008) of a student graph. Each student in $\mathcal{G}_1$ has at least one friend and one classmate in $\mathcal{G}_2$. The node size reflects the number of people per group $\mathcal{G}_i$.

sets of nodes with similar connectivity patterns to the rest of the graph, while clustering results in coherent, densely connected groups of nodes.

Optimizing specifically for Web graphs, the S-Node representation (Raghavan and Garcia-Molina 2003) is a novel two-level lossless graph compression scheme. Here a Web graph is a set of small directed graphs consisting of supernodes and superedges, which are pointers to lower-level graphs that encode the interconnections within a small subset of Web pages. S-Node exploits empirically observed properties of Web graphs like domain locality and page similarity, some of which can be viewed as node labels and others as additional textual information, to guide the grouping of pages into supernodes. Using a compression technique called reference encoding for the lower-level directed graphs, S-Node achieves high space efficiency and naturally isolates portions of Web graphs relevant to particular queries. This representation is the first Web graph representation scheme to combine compression with support for both complex queries and local graph navigation.

Mostly studied in the database community, grouping-based attributed graph summarization methods tend to rely on operations related to GROUP BY. SNAP and k-SNAP are two popular database-style approaches (Tian et al. 2008). SNAP relies on (A,R)-compatibility (attribute- and relationship-compatibility), which guarantees that nodes in all groups are homogeneous in terms of attributes, and are also adjacent to nodes in the same groups for *all* types of relationships. For example, in Figure 8, each student in $\mathcal{G}_1$ has at least one friend and classmate in $\mathcal{G}_2$. SNAP begins by creating groups of nodes that share the same attributes and then iteratively splits these groups until the grouping is "compatible" with the relationships, eventually producing the maximum (A,R)-compatible grouping. The nodes of the summary graph given by SNAP correspond to the groups, and the edges are the group relationships. k-SNAP further allows users to control the summary resolution, providing "drill-down" and "roll-up" capabilities to navigate through summaries of different resolutions.

To facilitate interactive summarization, CANAL (Zhang et al. 2010) automates k-SNAP by categorizing numerical attribute values, exploiting domain knowledge about the node labels and graph structure. To point users to the potentially most useful summaries, CANAL incorporates three "interestingness" criteria: (i) *Diversity*, the number of strong relationships connecting groups with different attribute values; (ii) *Coverage*, the fraction of nodes in the original graph that are present in strong group relationships; (iii) *Conciseness*, the sum of the number of groups and strong group relationships, where a lower sum is preferred. Overall, interestingness is given as $\frac{Diversity(S) \times Coverage(S)}{Conciseness(S)}$, where S is the summary graph.

Hassanlou et al. (2013) introduce another database-centered graph summarization approach similar to SNAP, where each node group consists of nodes that have the *same* attribute values using the GROUP BY operation. Unlike SNAP, though, this approach applies to probabilistic graphs or graphs with edges that have probabilities of existence associated with them. Shoaran et al. (2013)

extend this by aiming to protect the privacy of data in the labeled summaries generated by the aforementioned probabilistic technique. Finally, Gehrke et al. (2003) propose a privacy framework that extends Zero-Knowledge Privacy, improving on differential privacy by only considering a random sampling of data with added noise for the summarization.

In the database community, Fan et al. (2012) propose a "blueprint" for lossless queries on compressed attributed graphs. To achieve this, query-specific functions are introduced for compressing the graph, rewriting the query accordingly, and interpreting the result of the rewritten query on the compressed graph. For example, this blueprint can be implemented for queries of reachability (i.e., can node A be reached from node B?) and pattern matching (i.e., is there a subgraph that best satisfies a function provided by the user on path length between nodes in the subgraph?). The key idea is to group nodes that belong to the same equivalence class; intuitively, nodes that are similar in structure and labels are equivalent. This differs from other database-style operations that first group nodes by labels and later analyze the structure. To handle dynamic changes in web, social, and other networks, the authors also introduce unbounded algorithms that evaluate incremental graph structure changes and propagate the changes to the compressed graph representation. Ren and Wang (2015) propose a method similar to Fan et al. (2012) specifically for subgraph isomorphism queries, where groupings are based not only on equivalent nodes but also on edge-specific relationships that optimize the vertex matching order.

In the case of schema-less databases—in particular, for knowledge graphs connecting entities and concepts—Song et al. (2016) propose a lossy graph summarization framework as a collection of *d-summaries*, which intuitively are supergraphs that group similar entities (i.e., with the same attribute or label) within d hops of each other. Specifically, the entities within a d-summary observe what is called *d-similarity*, which preserves directed paths up to length d. Unlike frequent subgraph mining—a building block for various graph algorithms, including summarization—which is NP-hard, computing d-summaries is tractable. To evaluate d-summaries, the authors introduce approximations of an NP-hard "bi-criteria" function that quantifies *informativeness* and *diversity*. The former measure favors large summaries with high coverage of the original graph; the latter penalizes redundancy for entities appearing in many d-summaries. Both summarizing and querying knowledge graphs with d-summaries are efficient and can maintain up to 99% accuracy for subgraph queries in real and synthetic graphs.

Beyond attribute- and relationship-coherent summaries, there exists work on creating summaries from frequently occurring subgraphs in heterogeneous labeled graphs. A representative work is dependence graph summarization, where the vertices are labeled with program operations and the edges represent dependency relationships between them (Chen et al. 2009). The algorithm first generates partitions created by sampling nodes of the same label, resulting in multiple groups with consistent labels. The partitioning/summarization is followed by frequent subgraph mining and verification (removal of false positives). These steps are performed in multiple iterations to find a lower bound on the false-negative rate of frequent subgraph detection.

3.2 Bit Compression-Based Methods

Most compression-based summarization methods leverage MDL to guide the grouping of nodes or the discovery of frequent structures to be replaced with virtual nodes in the summary. Here, the employed compression and/or aggregation techniques consider both the graph structure and node/edge attributes.

The first and most famous frequent-subgraph-based summarization scheme, SUBDUE (Cook and Holder 1994), employs a two-part MDL representation (described in Section 2.2). Beyond the network structure, the MDL encoding accounts for node and edge labels. Greedy beam search is used to iteratively replace the most frequent subgraph in a labeled graph, which minimizes the

MDL cost, with a meta-node. Multiple passes of SUBDUE eventually produce a hierarchical description of the structural regularities in the graph. The resulting representation can be used to either identify anomalous structures (instances that do not compress well) or the most common substructures (substructures that have very low compression cost). Since the introduction of SUBDUE, many methods have been proposed to alleviate the complexity issues of frequent pattern mining on graphs or to extend its application in different settings: Maruhashi et al. (2011) propose *MultiAspectForensics*, a tool to detect and visualize graph patterns; Thomas et al. (2010) introduce MARGIN, an algorithm that reduces the search space of frequent subgraphs by only mining the maximal frequent subgraphs of a graph database; and Wackersreuther et al. (2010) propose a frequent subgraph mining algorithm to operate on dynamic graphs. Similarly to SUBDUE, a grammar-based compression scheme (Maneth and Peternek 2016) recursively replaces frequent "substructures" in directed edge-labeled hypergraphs, like RDF graphs. Rather than frequent subgraphs, these substructures are digrams, or pairs of connected hyperedges: For example, the digram "ab" consists of the edge labels "a" and "b." The process of recursive replacement of digrams stops when no digram occurs more than once. Unlike most compression-based works that use MDL, this approach leverages variable-length δ-codes (Elias 2006) for the connectivity and edge labels.

A simpler information-theoretic approach that does not use frequent subgraph mining directly minimizes the two-part MDL representation of an input network (Wu et al. 2014). The model cost consists of the number of bits to describe three parts: the number of node and attribute groups, the nodes in each group, and the links among groups. The data cost includes the description cost of the links inside each group and the attributes. The greedy summary-generating algorithm employs the MDL cost function to determine whether a certain node grouping is beneficial to the summary as a whole (i.e., it reduces the total encoding cost of the graph). A faster version of the greedy algorithm initializes the summaries using label propagation instead of random initialization.

Beyond its stand-alone utility, MDL can be easily combined with other techniques, such as locality-sensitive hashing (LSH) (Andoni and Indyk 2008), to help with in-memory processing and summary generation. LSH is a popular technique for efficient similarity search (here, nodes in the graph setting). In the context of summarization, it can operate on the structure and labels of each node to efficiently find similar nodes that can be aggregated into a "coherent" group. Khan et al. (2014) propose to LSH-based graph summarization by iteratively computing minhash functions on node neighborhoods, combining these minhash functions into groups, computing hash codes on the groups, and then aggregating the nodes that have the same hash codes. To handle the labels in the graph, adjacency and attribute lists are concatenated together before hashing. Supernodes are used to combine nodes, and, unlike other works, virtual nodes are used to combine edges between groups of nodes. Here, MDL is used to measure the relative increases in compression efficiency achieved by grouping nodes to supernodes and edges to superedges.

We further note that MDL is used frequently for data that, while not explicitly modeled as a graph, can be implicitly viewed as such: R-KRIMP and RDB-KRIMP (Koopman and Siebes 2008, 2009) summarize multi-relational data, which can be viewed as attributed graphs. The former, R-KRIMP, finds characteristic patterns in single data tables, then finds a small set of multi-relational characteristic item sets within this reduced search space. The latter extends R-KRIMP by finding more expressive patterns.

3.3 Influence-Based Methods

Influence-based summarization methods for labeled graphs are currently scarce. The representative method in this category leverages both structural and node attribute similarities to summarize the influence or diffusion process in a large-scale network.

The sole work in this category, VEGAS (Shi et al. 2015), summarizes influence propagation in citation networks via a matrix decomposition-based algorithm. The summarization problem aims to find the community membership matrix H of the nodes (articles in the citation network) such that $\min_{H \geq 0} ||M^G - HH^T||_F^2$, where $M^G = \frac{AA^T + A^TA}{2}$ is the node similarity matrix and A is the adjacency matrix. In the case of labeled networks, M^G is replaced with the generalized similarity matrix $M^D = \frac{(A \odot A^D)(A \odot A^D)^T + (A \odot A^D)^T(A \odot A^D)}{2}$ to incorporate side information. Here, $\odot$ indicates the Hadamard or element-wise product of matrices, and A^D, which may be specified by the user, encodes pairwise attribute similarity between nodes. In more detail, first the maximal influence graph G is computed from the input influence graph I by a rooted graph search that follows the standard BFS/DFS implementation from source node f. Then the matrices M^G, A^D, and M^D are generated. Finally, non-negative matrix factorization is used to solve the above optimization, yielding the community membership matrix H. Nodes are assigned to clusters according to the maximum value in each row of H. Summaries are generated after link pruning, which is performed to select the l best flows (links) for the final summary, dropping all other links.

4 DYNAMIC GRAPH SUMMARIZATION: PLAIN NETWORKS

Analyzing large and complex data is challenging by itself, so adding the dimension of time makes the analysis even more challenging and time-consuming. Despite this, most networks realistically do change over time: for example, communication patterns with others via phone or social networks; the connection between servers in a network; the flow of information, news and rumors; the distance between connected vehicles; the information transmitted between devices in a smart home environment.

For this reason, the temporal graph mining literature is rich, mostly focusing on: laws and patterns of graph evolution in Leskovec and Faloutsos (2007), Ferlez et al. (2008), Leskovec et al. (2008), Leskovec et al. (2005), Sun et al. (2008) and a comprehensive survey by Aggarwal and Subbian (2014); anomaly and change detection in streaming graphs (Aggarwal and Philip 2005) or time-evolving networks (Ferlez et al. 2008; Koutra et al. 2013, 2015) ; discovery of dense temporal cliques and bipartite cores using PARAFAC tensor decomposition and MDL (Sun et al. 2007; Araujo et al. 2014; Koutra et al. 2012); mining of cross-graph quasi-cliques (Pei et al. 2005); clustering using incremental static clustering (Xu et al. 2011) or a probabilistic approach based on mixed-membership blockmodels (Fu et al. 2009); sampling of streaming graphs (Ahmed et al. 2013) and role discovery (Henderson et al. 2012; Rossi et al. 2012).

In this section, we focus on methods that summarize time-evolving networks (third block in Table 1). Summarization techniques for time-evolving networks have not been studied to the same extent as those for static networks, possibly because of the new challenges introduced by the dimension of time. The methods are sensitive to the choice of time granularity, which is often chosen arbitrarily: depending on the application, granularity can be set to minutes, hours, days, weeks, months, years, or some other unit that makes sense in a given setting. The continuous and sometimes irregular change of real-world graphs also complicates evolution tracking, defining online "interestingness" measures, and visualization. The dynamic graph summarization problem may be defined as:

PROBLEM 3. Summarization of Dynamic, Plain Graphs.
Given a dynamic graph, which is observed as a set of streaming edges, or a sequence of adjacency matrices $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_T$ corresponding to the static graphs $G_1, G_2, \ldots, G_T$
Find a temporal summary graph or a set of possibly overlapping temporal structures to **concisely describe** the given dynamic graph.

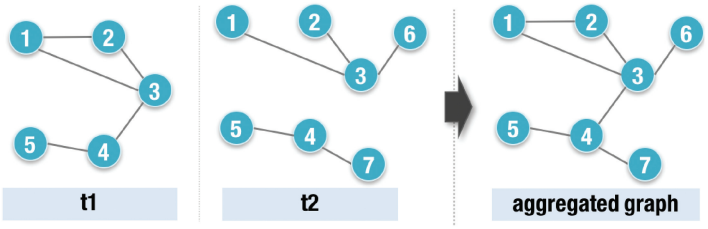

Fig. 9. Aggregated graph example (time t1+t2).

The summary is a time-evolving supergraph with supernodes and superedges or else a sequence of sparsified graphs with fewer nodes/edges than the input dynamic graph.

The simplest approach treats a time-evolving graph as a series of static graph snapshots, which allows the application of static graph summarization techniques on each snapshot. However, the effectiveness of this approach depends heavily on user-specified aggregation operations and the time granularity (Soundarajan et al. 2016), and there is no globally established method for picking the "right" time unit. With small time granularity, the amount of data increases significantly. With large time granularity, interesting dynamics may be missed. Moreover, real-world processes can be unpredictable or bursty. Adjusting the time unit of analysis may be the key to understanding and capturing the important dynamics.

An alternative is to create an aggregate graph that summarizes the input dynamic network based on the recency and frequency of interactions (Figure 9). This has been called an "approximation graph" (Cortes et al. 2001; Hill et al. 2006; Sharan and Neville 2008). Specifically, the interactions between nodes in an approximation graph are aggregated over time and weighted by applying kernel smoothing (e.g., exponential, inverse linear, linear, uniform), where more recent edges are weighted higher than old edges. Edges with weight below a specified threshold can also be pruned to simplify the graph approximation. The approximation graph has been shown to be useful for telecommunications fraud detection (Cortes et al. 2001), anomaly detection and prediction of user behavior in web logs and email networks (Hill et al. 2006), and attribute classification via relational classifier models (Sharan and Neville 2008).

The approximation graph can be used as input to any of the static graph summarization algorithms presented in Section 2. However, this approach has the same shortcoming as the straightforward approach—namely, it depends on the time granularity of the input graph sequence. Probabilistic relational models (PRM) and relational Markov decision processes (RMDP, which are a sequence of PRMs forming a chain that follows a first-order Markov assumption) have also been used to model dynamic graphs (Guestrin et al. 2003), but they cannot model time-varying edges and treat them as fixed over time.

4.1 Grouping-Based Methods

Grouping-based summarization approaches recursively aggregate nodes and timesteps to reduce the size of large-scale dynamic networks.

NetCondense (Adhikari et al. 2017) is a node-grouping approach that maintains specific properties of the original time-varying graph, like diffusive properties important in marketing and influence dynamics, governed by its maximum eigenvalue. In this context, given a dynamic network of T snapshots and an epidemiology model, the goal is to find a reduced network series with few groups of nodes (supernodes) and groups of timesteps so that the change in its maximum eigenvalue is minimized. In its general form, this problem is intractable, but it can be transformed

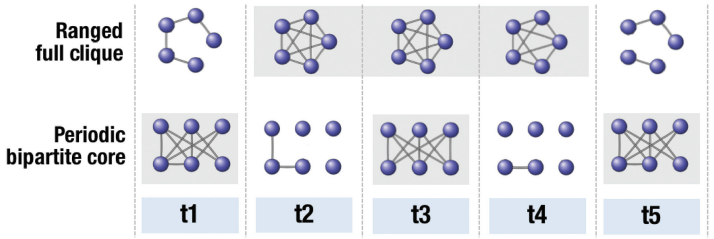

Fig. 10. Examples of temporal patterns identified by TimeCrunch (Shah et al. 2015). Ranged full clique at times t2 through t4 and periodic bipartite core every other timestep.

into an equivalent static-graph problem with a well-conditioned, flattened network whose eigenvalue is easy to compute and has similar diffusive properties as the original dynamic network. This observation allows solving the dynamic problem with an algorithm similar to coarseNet (Purohit et al. 2014) (Section 2). In this case, after flattening the dynamic network, NetCondense repeatedly merges adjacent node pairs and adjacent time pairs, evaluating the change in the flattened network's maximum eigenvalue. The changes are sorted in increasing order and the best node-/time-pairs are merged until the user-specified network size is achieved. NetCondense uses transformations and approximations to achieve sub-quadratic running time and linear space complexity.

In many applications such as network monitoring and urban planning, network edges are observed sequentially. Traditional sketching techniques (Zhao et al. 2011; Cormode and Muthukrishnan 2005a) usually maintain only frequency counts, "dropping" the information of the graphical structure, although the goal in summarization is to both construct a summarized *graph* in linear time and to support edge updates in constant time. To this end, TCM (Tang et al. 2016) approximates a variety of graph queries by creating and querying d graph sketches and returning the minimum answer. Each graph sketch i is created by mapping the original nodes to "node buckets" or supernodes via a hash function h_i. The edges between supernodes in the graph sketch are superedges corresponding to the sum of the connections between their constituent nodes. The more pairwise independent hash functions (sketches) are used, the lower the probability of hash collisions and thus the more precise are the answers to the queries. By maintaining the graphical structure, TCM supports complex analytics over graph streams, such as conditional node queries, aggregated edge weights, aggregated node flows, reachability path queries, aggregate subgraph queries, and triangles.

4.2 Bit Compression-Based Methods

The techniques in this category use compression as a means of extracting meaningful patterns from temporal data. This category's only representative is TimeCrunch (Shah et al. 2015), which succinctly describes a large dynamic graph with a set of important temporal structures. Extending VoG (Koutra et al. 2014b) (Section 2.2), the authors formalize temporal graph summarization as an information-theoretic optimization problem where the goal is to identify the temporal behaviors of local static structures that collectively minimize the global description length of the dynamic graph. A lexicon that describes various types of temporal behavior (flickering, periodic, one-shot) is introduced to augment the vocabulary of static graphs (stars, cliques, bipartite cores, chains). Figure 10 illustrates examples of patterns identified by TimeCrunch.

TimeCrunch (i) first identifies static structures in each timestamp, (ii) labels them using the static lexicon, (iii) stitches them together to find temporal structures, (iv) then labels those using the temporal lexicon, and (v) selects for the summary the temporal structures that help minimize

the MDL cost of describing the time-evolving graph. Stitching static structures corresponds to evolution tracking, which is handled via iterative rank-1 singular value decomposition (SVD) to find potentially temporally coherent structures. Then, cosine similarity ensures the temporal coherence of the discovered structures. Following up on this work, the EcoViz system (Jin and Koutra 2017b; Shah et al. 2017) leverages TimeCrunch to interactively visualize and compare time-evolving summaries of functional human connectomes (i.e., fMRI-based brain networks).

4.3 Influence-Based Methods

Influence and diffusion processes are inherently time evolving. The methods in this category summarize the influence process mainly in dynamic social networks. In Section 2.4 we present two techniques, CSI (Mehmood et al. 2013) and SPINE (Mathioudakis et al. 2011), that summarize social graphs by leveraging information propagation and social influence processes. These approaches have a temporal aspect, since they are summarizing inherently temporal activities in networks, but they operate on static graphs, where the directed edges capture influence.

Here we focus on a method that summarizes interestingness-driven diffusion processes in dynamic graphs. The input of OSNet (Qu et al. 2014) is a stream of time-ordered interactions, represented as undirected edges between labeled nodes. Its goal is to capture cascades (for example, the spread of news) in a directed graph that reveals the flow of dynamics. The output summary consists of subgraphs with "interesting" nodes from the original input graph, where interestingness is defined as a linear combination of the out-degree of a node (the number of nodes that it infects during the diffusion process), and the maximum "propagation radius" (the length of the path from the root of the diffusion process to the node). The core technical ideas of OSNet are (i) to construct spreading trees and (ii) to compute the interestingness of a summary via its entropy and a threshold that can lead to fast convergence. OSNet outperforms static-based summarization techniques (Toivonen et al. 2011; Navlakha et al. 2008) that give a summary per timestamp, since they are not suited for capturing temporal dynamics; the former depends on user-defined parameters, and the latter gives summaries with many disconnected cliques.

Relatedly, Lin et al. (2008) focus on understanding a social group's collective activity over time. To this end, the authors extract activity themes over time using non-negative matrix factorization on a multi-graph (user-photo, user-comment, photo-tag, and comment-tag graphs) to obtain latent spaces for users and concepts. The top k users and terms in the latent space define the "important" actions, which correspond to activity themes. Evolution of themes over time is tracked by applying cosine similarity between their corresponding latent spaces, similarly to the evolution tracking component of TimeCrunch (Shah et al. 2015), which also uses cosine similarity to ensure temporal coherence. Lin et al. (2008) visualize the themes as bubbles connected by edges, each of which has a length inversely proportional to the similarity of the themes.

Connections to graph clustering, sparsification, and compression. As with static graphs, techniques such as clustering, sparsification, and compression are related to summarization methods for dynamic graphs. Some clustering methods extend heuristics that have been used for static graphs, such as modularity (Görke et al. 2010) or minimum-cut trees (Saha and Mitra 2007), and others introduce definitions specific to the temporal domain (Tantipathananandh and Berger-Wolf 2011). As discussed in Section 2, graph sketches (Ahn et al. 2012; Liberty 2013) summarize large amounts of data by applying linear projections. The property of linearity is fundamental, as it makes sketches applicable to the analysis of streaming graphs in centralized or even distributed settings, where they are partitioned in multiple servers with MapReduce (Dean and Ghemawat 2004). One-pass and other efficient streaming algorithms with their theoretical analysis are given in Ahn et al. (2012).

Table 2. Qualitative Comparison of Summarization Approaches for Query Handling

Method	Query Type							Graph Type
	Star	Neighborhood	Degree	Triangles	Patt. Matching	Reachability	PageRank	
Graph dedensification (Maccioni and Abadi 2016) (Section 2)	✓				✓			social, web
l_p-reconstr. Error (Riondato et al. 2014) (Section 2)		✓	✓	✓				social
Query-pres. Fan et al. (2012) (Section 3)					✓	✓		citation, social, web
Neighbor friendly compr. (Maserrat and Pei 2010) (Section 2)		✓						social
GraSS (LeFevre and Terzi 2010) (Section 2)			✓				✓	co-authorship, wiki
S-Node (Raghavan and Garcia-Molina 2003) (Section 3)		✓					✓	collaboration, web
d-summaries (Song et al. 2016) (Section 3)	✓				✓			knowledge graph

Work on compressing dynamic graphs for storage includes lossy compression of time-evolving graphs (Henecka and Roughan 2015), and encoding of dynamic, weighted graphs as three-dimensional arrays (tensor) by reducing heterogeneity and guaranteeing compression error within bounds (Liu et al. 2012). The latter is based on hierarchical clusters of edge weights and graph compression using run-length encoding, traversing first the tensor's time dimension and second the tensor's vertex dimensions. This method maintains the connectivity of the graph as defined by the average shortest paths over all pairs of connected nodes. Thus, it handles related queries with good approximations.

5 GRAPH SUMMARIZATION IN REAL-WORLD APPLICATIONS

As we mention in Section 1, summarization helps mitigate information overload. In this section, we discuss real-world applications of graph summarization, which are myriad and relevant in many domains.

5.1 Summarization for Query Handling and Efficiency

Graph summarization can greatly improve query execution and efficiency across different graph-specific queries. Such queries may seek node-related information like degree, PageRank, or participating triangles, or look to identify or match subgraphs within a larger graph. Table 2 outlines several types of queries used to evaluate graph summarization methods.

Pattern-matching queries are extremely common on graph databases. For example, across star queries involving nodes of different degrees, graph dedensification (Maccioni and Abadi 2016) improves query efficiency as the size of the queried graph increases, yielding the best improvements (up to 10× speedup) for queries involving only high-degree nodes (Section 2). Fan et al. (2012) propose an attributed graph compression method and query transformation scheme for lossless pattern matching queries, achieving a compression rate up to 92% and runtime reduction up to 70% (Section 3). From a systems point of view, Čebirić et al. (2015) propose query-oriented graph summarization on Resource Description Framework (RDF) graphs, which are the standard model for W3C web resources. Many methods for pattern matching queries also exist outside graph summarization in databases and graph analytics (Tong et al. 2007; Tian and Patel 2008; Fan et al. 2013; Pienta et al. 2014), but these are beyond our survey's scope.

Summarization has been shown to improve query efficiency in diverse domains. For example, a typical query on a social network graph might ask whether an edge exists between two nodes, or more generally whether there exists a path between two nodes. Such a query can be answered on

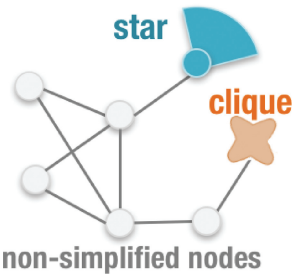

Fig. 11. Example of simplified network with one clique motif glyph and one star motif glyph (Dunne and Shneiderman 2013).

a space-efficient summary of an expected adjacency matrix (Riondato et al. 2014). This approach constructs the graph summary up to 12,500× faster than its baseline GraSS (LeFevre and Terzi 2010) and also achieves lower average query error. Another application is on Web graphs (Raghavan and Garcia-Molina 2003): here, the S-Node representation (Section 3) outperforms other representation schemes by an order of magnitude on complex web navigation queries by loading only a relatively small number of intranode and superedge graphs and avoiding disk I/Os when possible, leading to 75–90% reduction in navigation time compared to baselines. A final application is knowledge graphs (Song et al. 2016), which lead to up to 40× speedup over an optimized frequent subgraph mining algorithm on generated knowledge graphs for a variety of subgraph queries.

5.2 Summarization for Visualization and Pattern Discovery

Summarization can enable visualization of data too large to load, display, and interactively explore in original raw format. For example, Shen et al. (2006) apply OntoVis (Section 2.3) on a large heterogeneous movie network consisting of eight node types (person, movie, role, etc.) with 35,000 nodes and 108,000 links: Though relatively small, this graph is still too dense to fit on a desktop screen. To investigate the relationships between persons and roles, the authors visually observe the summarized network and identify a role-actor relationship where a good actor should be able to play different roles (for example, actors like Woody Allen and Sandra Bullock play three different types of roles). Other works that perform visualization on top of summarization include VoG (Koutra et al. 2014a; Jin and Koutra 2017b; Shah et al. 2017), which visualizes structures of specific types (e.g., cliques, bipartite cores), and Motif Simplification (Dunne and Shneiderman 2013), which visualizes simplified networks of up to 8,000 nodes with glyphs, a toy example of which is given in Figure 11.

Summarization also supports pattern discovery by maintaining "interesting" or "salient" patterns. Consider the `Wikipedia-Controversy` dataset, in which nodes are Wikipedia contributors and edges connect users who edit the same part of the article. Koutra et al. (2014a) apply VoG on this graph to extract the 10 most informative structures, obtaining 8 stars and 2 bipartite subgraphs. The centers of the stars correspond to admins or heavily active contributors. The bipartite cores correspond to edit wars between groups of users, like vandals and responsible editors, on a controversial topic.

SUBDUE (Cook and Holder 1994), one of the most famous frequent-pattern mining methods, is applied in areas as diverse as chemical compound analysis, scene analysis, and CAD circuit design analysis. For example, SUBDUE discovers substructures in chemical compound graphs where atoms are vertices and edges are bonds, in particular discovering the building-block components that are heavily used, such as isoprene units for rubber compounds.

5.3 Summarization for Influence Extraction

Influence analysis is a long-standing research focus and objective of graph mining. In graph summarization, Li and Lin (2009) use egocentric abstraction to extract influence from a simulated heterogeneous crime dataset with nodes as gangs and edges as gang relations. In a user study, they demonstrate that the graph abstraction leads to more accurate, efficient, and confident identification of high-level crime-committing gangs. Furthermore, it is demonstrated that each abstraction view captures different parts of key criminal evidence to some extent: for example, "the gang has hired a middleman intending to commit a crime."

Another example is CoarseNet (Purohit et al. 2014): Applied on cascade network Flixster of 56,000 nodes and 560,000 edges, it is demonstrated that a large fraction of movies propagate in a small number of groups with a multi-modal distribution, suggesting movies have multiple scales of spread. Finally, Mehmood et al. (2013) use community-level social influence analysis on Yahoo! and Twitter graphs to observe almost no correlation between influence and link probabilities. In other words, it is demonstrated that influence relationships do not in general exhibit any clear structure. Even dense communities do not necessarily exhibit strong internal influence.

6 CONCLUSION

In this survey, we present the state-of-the-art in graph summarization. Distinguishing between types of input graph and core summarization techniques, we propose a taxonomy to categorize existing graph summarization algorithms. We introduce the key details of each algorithm and explore relations between relevant works and methods, also providing examples of real-life applications for each algorithm category. Here, we point readers to important open problems in the field.

6.1 Open Research Problems

While graph summarization research is advancing, the field is still relatively new and underexplored. First, further work is to be done in handling diverse input data types. There does not yet exist work on summarizing temporal graphs with side information, even though many real-world networks, like social networks, can easily (and perhaps most accurately) be modeled as temporal attributed graphs. Even beyond the temporal aspect, other static graph types have yet to be addressed. An example is the *multi-layer graph*, which is an important model for Web graphs (Laura et al. 2002); another is the *multiview graph*, which can be "viewed" from its different edge types. For instance, a Twitter graph could comprise separate adjacency matrices for follows, retweets, and messages. As data become increasingly richer, methods will need to handle graphs that comprise multiple views or incorporate other types of data, like time series associated with network nodes.

Another area of improvement is standardizing, generalizing, or extending algorithmic and evaluation techniques. For example, numerous methods tailored toward query efficiency on graph summaries exist, but they either perform approximate queries or are limited to very specific exact queries. Further work should address lossless compression with general-purpose queries. Another example is labeled graphs: Existing methods group nodes with cohesive attribute values, but in some applications heterogeneous clusters are crucial. For example, such clusters could facilitate anomaly detection or else refer to groups with desired diversity, as in an academic or professional setting. In terms of evaluation, current measures are also usually highly application specific. Compression-based methods are evaluated on compression quality; query-oriented methods are evaluated on query latency; and so on. Some common evaluation metrics can make comparison of new and established approaches easier: For example, metrics that evaluate supergraphs on sparsity, least information loss, and ease of visualization.

Finally, a promising new direction is graph summarization using deep node representations learned automatically from the context encoded in the graph. Node representation learning has attracted significant interest in recent years (Perozzi et al. 2014; Grover and Leskovec 2016; Wang et al. 2016; Tang et al. 2015; Ribeiro et al. 2017; Heimann et al. 2018). Given the existence of summarization methods using latent node representations (e.g., via factorization) or manually selected node/egonet/k-hop neighborhood features, as well as the recent successes of deep learning, deep node representations for summarization naturally seem promising.

Overall, summarization methods are becoming increasingly important and useful as the volume of available interconnected data rapidly grows. While we overview several formulations of graph summarization already studied, we conclude by noting that many promising directions in the field remain unexplored and thus full of potential for impact.

REFERENCES

Bijaya Adhikari, Yao Zhang, Aditya Bharadwaj, and B. Aditya Prakash. 2017. Condensing temporal networks using propagation. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. 417–425.

Charu Aggarwal and Karthik Subbian. 2014. Evolutionary network analysis: A survey. *ACM Comput. Surv.* 47, 1 (2014), 10:1–10:36.

Charu C. Aggarwal. 2015. *Data Mining: The Textbook*. Springer.

Charu C. Aggarwal and S. Yu Philip. 2005. Online analysis of community evolution in data streams. In *Proceedings of the SIAM international Conference on Data Mining (SDM'05)*.

Charu C. Aggarwal and Haixun Wang. 2010. A survey of clustering algorithms for graph data. In *Managing and Mining Graph Data*. Springer, 275–301.

Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd International Conference on World Wide Web (WWW'13)*. 37–48.

Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. 2012. Graph sketches: Sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. 5–14.

Sebastian E. Ahnert. 2013. Power graph compression reveals dominant relationships in genetic transcription networks. *Molec. BioSyst.* 9, 11 (2013), 2681–2685.

Alfred V. Aho, M. R. Garey, and Jeffrey D. Ullman. 1972. The transitive reduction of a directed graph. *Siam J. Comput.* 1, 2 (1972), 131–137.

Leman Akoglu, Duen Horng Chau, U. Kang, Danai Koutra, and Christos Faloutsos. 2012. OPAvion: Mining and visualization in large graphs. In *Proceedings of the 2012 SIGMOD Conference*. ACM, 717–720.

Leman Akoglu, Mary McGlohon, and Christos Faloutsos. 2010. OddBall: Spotting anomalies in weighted graphs. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'10)*.

Leman Akoglu, Hanghang Tong, Jilles Vreeken, and Christos Faloutsos. 2012. Fast and reliable anomaly detection in categorical data. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM'12)*. ACM.

Charles J. Alpert, Andrew B. Kahng, and So-Zen Yao. 1999. Spectral partitioning with multiple eigenvectors. *Discr. Appl. Math.* 90, 1 (1999), 3–26.

Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 51, (2008), 117–122.

Alberto Apostolico and Guido Drovandi. 2009. Graph compression by BFS. *Algorithms* 2, 3 (2009), 1031–1044.

Miguel Araujo, Spiros Papadimitriou, Stephan Günnemann, Christos Faloutsos, Prithwish Basu, Ananthram Swami, Evangelos E. Papalexakis, and Danai Koutra. 2014. Com2: Fast automatic discovery of temporal ("comet") communities. In *Advances in Knowledge Discovery and Data Mining*. Lecture Notes in Computer Science, Vol. 8444. Springer, 271–283.

Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. 2006. Group formation in large social networks: Membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*. ACM, 44–54.

Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 2009. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*.

Joshua D. Batson, Daniel A. Spielman, Nikhil Srivastava, and Shang-Hua Teng. 2013. Spectral sparsification of graphs: Theory and algorithms. *Commun. ACM* 56, 8 (2013), 87–94. DOI: http://dx.doi.org/10.1145/2492007.2492029

Enrico Bertini and Giuseppe Santucci. 2004. By chance is not enough: Preserving relative density through non uniform sampling. In *Proceedings of the Information Visualisation Conference*.

Paolo Boldi and Sebastiano Vigna. 2004. The webgraph framework I: Compression techniques. In *Proceedings of the International World Wide Web Conference*. 595–602.

Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3 data-driven documents. *IEEE Trans. Vis. Comput. Graph.* 17, 12 (2011), 2301–2309. DOI: http://dx.doi.org/10.1109/TVCG.2011.185

Ivan Brugere, Brian Gallagher, and Tanya Y. Berger-Wolf. 2016. Network structure inference, a survey: Motivations, methods, and applications. *ACM Comput. Surv.* 51, 2, Article 24. http://arxiv.org/abs/1610.00782.

Gregory Buehrer and Kumar Chellapilla. 2008. A scalable pattern mining approach to web graph compression with communities. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 95–106.

Gemma Casas-Garriga. 2005. Summarizing sequential data with closed partial orders. In *Proceedings of the SIAM International Conference on Data Mining (SDM'05)*. 380–391.

Šelja Čebirić, François Goasdoué, and Ioana Manolescu. 2015. Query-oriented summarization of RDF graphs. *Proc. VLDB Endow.* 8, 12 (2015), 2012–2015.

Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S. Modha, and Christos Faloutsos. 2004. Fully automatic cross-associations. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'04)*. 79–88.

Varun Chandola and Vipin Kumar. 2005. Summarization – Compressing data into an informative representation. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'05)*. 98–105.

Duen Horng Chau, Aniket Kittur, Jason I. Hong, and Christos Faloutsos. 2011. Apolo: Making sense of large network data by combining rich user interaction and machine learning. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'11)*.

Chen Chen, Cindy X. Lin, Matt Fredrikson, Mihai Christodorescu, Xifeng Yan, and Jiawei Han. 2009. Mining graph patterns efficiently via randomized summaries. *Proc. VLDB Endow.* 2, 1 (2009), 742–753.

Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan. 2009. On compressing social networks. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'09)*. 219–228.

Yongwook Choi and Wojciech Szpankowski. 2012. Compression of graphical structures: Fundamental limits, algorithms, and experiments. *IEEE Trans. Inf. Theory* 58, 2 (2012), 620–638.

Rudi Cilibrasi and Paul Vitányi. 2005. Clustering by compression. *IEEE Trans. Inf. Theory* 51, 4 (2005), 1523–1545.

Diane J. Cook and Lawrence B. Holder. 1994. Substructure discovery using minimum description length and background knowledge. *J. Artif. Intell. Res.* 1 (1994), 231–255.

Graham Cormode and S. Muthukrishnan. 2005a. An improved data stream summary: The count-min sketch and its applications. *J. Algor.* 55, 1 (2005), 58–75.

Graham Cormode and S. Muthukrishnan. 2005b. Summarizing and mining skewed data streams. In *Proceedings of the SIAM international Conference on Data Mining (SDM'05)*.

Corinna Cortes, Daryl Pregibon, and Chris Volinsky. 2001. Communities of interest. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*. 105–114.

Uros Damnjanovic, Virginia Fernandez Arguedas, Ebroul Izquierdo, and José M. Martínez. 2008. Event detection and clustering for surveillance video summarization. In *Proceedingsof the 9th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'08)*. IEEE, 63–66.

Pedro O. S. Vaz de Melo, Leman Akoglu, Christos Faloutsos, and Antonio Alfredo Ferreira Loureiro. 2010. Surprising patterns for the call duration distribution of mobile phone users. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD'10)*. 354–369.

Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI'04)*. 10.

Pravallika Devineni, Danai Koutra, Michalis Faloutsos, and Christos Faloutsos. 2015. If walls could talk: Patterns and anomalies in Facebook wallposts. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'15)*. 367–374.

Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. 2005. A fast kernel-based multilevel algorithm for graph clustering. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'05)*. ACM, 629–634.

Laxman Dhulipala, Igor Kabiljo, Brian Karrer, Giuseppe Ottaviano, Sergey Pupyrev, and Alon Shalita. 2016. Compressing graphs and indexes with recursive graph bisection. arXiv:1602.08820.

P. Drineas, R. Kannan, and M. W. Mahoney. 2006. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. *SIAM J. Comput.* 36, 1 (2006), 184–206.

Cody Dunne and Ben Shneiderman. 2013. Motif simplification: Improving network visualization readability with fan, connector, and clique glyphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, 3247–3256.

P. Elias. 2006. Universal codeword sets and representations of the integers. *IEEE Trans. Inf. Theor.* 21, 2 (2006), 194–203.

Wenfei Fan, Jianzhong Li, Xin Wang, and Yinghui Wu. 2012. Query preserving graph compression. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 157–168.

Wenfei Fan, Xin Wang, and Yinghui Wu. 2013. Diversified top-k graph pattern matching. *Proc. VLDB Endow.* 6, 13 (2013), 1510–1521.

Jing Feng, Xiao He, Nina Hubig, Christian Böhm, and Claudia Plant. 2013. Compression-based graph mining exploiting structure primitives. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'13)*. IEEE, 181–190.

Jure Ferlez, Christos Faloutsos, Jure Leskovec, Dunja Mladenic, and Marko Grobelnik. 2008. Monitoring network evolution using MDL. In *Proceedings of the 24th International Conference on Data Engineering (ICDE'08)*. 1328–1330.

Wenjie Fu, Le Song, and Eric P. Xing. 2009. Dynamic mixed membership blockmodel for evolving networks. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML'09)*. ACM, 329–336.

Johannes Gehrke, Edward Lui, and Rafael Pass. 2003. Towards privacy for social networks: A zero-knowledge based definition of privacy. In *Proceedings of the 8th Conference on Theory of Cryptography*. 432–449.

Mina Ghashami, Edo Liberty, and Jeff M. Phillips. 2016. Efficient frequent directions algorithm for sparse matrices. arXiv:1602.00412.

Sean Gilpin, Tina Eliassi-Rad, and Ian Davidson. 2013. Guided learning for role discovery (gLRD): Framework, algorithms, and applications. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13)*. ACM, 113–121.

Oshini Goonetilleke, Danai Koutra, Timos Sellis, and Kewen Liao. 2017. Edge labeling schemes for graph data. In *Proceedings of the Scientific and Statistical Database Management Conference (SSDBM'17)*. ACM, Article 12, 12 pages.

Robert Görke, Pascal Maillard, Christian Staudt, and Dorothea Wagner. 2010. Modularity-driven clustering of dynamic graphs. In *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA'10)*. 436–448.

Szymon Grabowski and Wojciech Bieniecki. 2014. Tight and simple web graph compression for forward and reverse neighbor queries. *Discr. Appl. Math.* 163 (2014), 298–306.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'16)*. ACM.

Carlos Guestrin, Daphne Koller, Chris Gearhart, and Neal Kanodia. 2003. Generalizing plans to new environments in relational MDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*.

Mohammad Al Hasan, Nesreen K. Ahmed, and Jennifer Neville. 2013. Network Sampling: Methods and Applications. Retrieved from https://www.cs.purdue.edu/homes/neville/courses/NetworkSampling-KDD13-final.pdf.

Nasrin Hassanlou, Maryam Shoaran, and Alex Thomo. 2013. Probabilistic graph summarization. In *Web-Age Information Management*. Springer, 545–556.

Mark Heimann, Haoming Shen, and Danai Koutra. 2018. Node representation learning for multiple networks: The case of graph alignment. arXiv:1802.06257.

Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. RolX: Structural role extraction & mining in large graphs. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'12)*. ACM, 1231–1239.

Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. 2011. It's who you know: Graph mining using recursive structural features. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'11)*. ACM, 663–671.

Wilko Henecka and Matthew Roughan. 2015. Lossy compression of dynamic, weighted graphs. In *Proceedings of the 2015 3rd International Conference on Future Internet of Things and Cloud (FiCloud'15)*. 427–434.

Shawndra Hill, Deepak Agarwal, Robert Bell, and Chris Volinsky. 2006. Building an effective representation for dynamic networks. *J. Comput. Graph. Stat.* 15 (2006), 1–25.

Christian Hübler, Hans-Peter Kriegel, Karsten Borgwardt, and Zoubin Ghahramani. 2008. Metropolis algorithms for representative subgraph sampling. In *Proceedings of the 2008 8th IEEE International Conference on Data Mining (ICDM'08)*. IEEE, 283–292.

Di Jin and Danai Koutra. 2017a. Exploratory analysis of graph data by leveraging domain knowledge. In *Proceedings of the 2017 IEEE International Conference on Data Mining*. 187–196.

Di Jin, Aristotelis Leventidis, Haoming Shen, Ruowang Zhang, Junyue Wu, and Danai Koutra. 2017. PERSEUS-HUB: Interactive and collective exploration of large-scale graphs. *Informatics* 4, 3 (2017), 22.

Lisa Jin and Danai Koutra. 2017b. ECOviz: Comparative visualization of time-evolving network summaries. In *Proceedings of the ACM SIGKDD 2017 Workshop on Interactive Data Exploration and Analytics*.

U. Kang, Jay-Yoon Lee, Danai Koutra, and Christos Faloutsos. 2014. Net-ray: Visualizing and mining web-scale graphs. In *Proceedings of the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'14)*.

U. Kang, Hanghang Tong, Jimeng Sun, Ching-Yung Lin, and Christos Faloutsos. 2011. Gbase: A scalable and general graph management system. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'11)*. ACM, 1091–1099.

U. Kang, Charalampos E. Tsourakakis, and Christos Faloutsos. 2009. PEGASUS: A peta-scale graph mining system—implementation and observations. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'09)*.

George Karypis and Vipin Kumar. 1999. Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th Annual ACM/IEEE Design Automation Conference*. 343–348.

David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the Conference of the ACM Special Interest Group on Knowledge Discovery and Data Mining*. ACM.

Kifayat Ullah Khan, Waqas Nawaz, and Young-Koo Lee. 2014. Set-based unified approach for attributed graph summarization. In *Proceedings of the IEEE 4th International Conference on Big Data and Cloud Computing (BdCloud'14)*. IEEE, 378–385.

Mikko Kivel, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. 2014. Multilayer networks. *J. Complex Netw.* 2, 3 (2014), 203–271.

Arne Koopman and Arno Siebes. 2008. Discovering relational items sets efficiently. In *Proceedings of the SIAM International Conference on Data Mining (SDM'08)*. 108–119.

Arne Koopman and Arno Siebes. 2009. Characteristic relational patterns. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'09)*. 437–446.

Danai Koutra, Abhilash Dighe, Smriti Bhagat, Udi Weinsberg, Stratis Ioannidis, Christos Faloutsos, and Jean Bolot. 2017. PNP: Fast path ensemble method for movie design. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'17)*.

Danai Koutra and Christos Faloutsos. 2017. *Individual and Collective Graph Mining: Principles, Algorithms, and Applications*. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool.

Danai Koutra, Di Jin, Yuanchi Ning, and Christos Faloutsos. 2015. Perseus: An interactive large-scale graph mining and visualization tool. *Proc. VLDB Endow.* 8, 12, 1924–1927.

Danai Koutra, U. Kang, Jilles Vreeken, and Christos Faloutsos. 2014b. VoG: Summarizing and understanding large graphs. In *Proceedings of the SIAM international Conference on Data Mining (SDM'14)*. 91–99.

Danai Koutra, U. Kang, Jilles Vreeken, and Christos Faloutsos. 2014a. VoG: Summarizing and understanding large graphs. In *Proceedings of the SIAM international Conference on Data Mining (SDM'14)*. SIAM.

Danai Koutra, Vasileios Koutras, B. Aditya Prakash, and Christos Faloutsos. 2013. Patterns amongst competing task frequencies: Super-linearities, and the almond-DG model. In *Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'13)*. 201–212.

Danai Koutra, Evangelos E. Papalexakis, and Christos Faloutsos. 2012. TensorSplat: Spotting latent anomalies in time. In *Proceedings of the 2012 16th Panhellenic Conference on Informatics (PCI'12)*. IEEE, 144–149.

Danai Koutra, Neil Shah, Joshua Vogelstein, Brian Gallagher, and Christos Faloutsos. 2015. DeltaCon: A principled massive-graph similarity function with attribution. *ACM Trans. Knowl. Discov. Data* 10, 3, Article 28.

Danai Koutra, Joshua Vogelstein, and Christos Faloutsos. 2013. DeltaCon: A principled massive-graph similarity function. In *Proceedings of the SIAM International Conference on Data Mining (SDM'13)*. 162–170.

Luigi Laura, Stefano Leonardi, Guido Caldarelli, and Paolo De Los Rios. 2002. A multi-layer model for the web graph. In *On-Line Proceedings of the 2nd International Workshop on Web Dynamics*.

Matthijs Leeuwen van Leeuwen, Jilles Vreeken, and Arno Siebes. 2006. Compression picks the item sets that matter. In *Proceedings of the Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'06)*. 585–592.

Kristen LeFevre and Evimaria Terzi. 2010. GraSS: Graph structure summarization. In *Proceedings of the SIAM International Conference on Data Mining (SDM'10)*. 454–465.

Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. 2008. Microscopic evolution of social networks. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'08)*. 462–470.

Jure Leskovec and Christos Faloutsos. 2007. Scalable modeling of real graphs using Kronecker multiplication. In *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*. 497–504.

Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. 2005. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 177–187.

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of Massive Datasets*. Cambridge University Press.

Cheng-Te Li and Shou-De Lin. 2009. Egocentric information abstraction for heterogeneous social networks. In *Proceedings of the International Conference on Advances in Social Network Analysis and Mining (ASONAM'09)*. IEEE, 255–260.

Edo Liberty. 2013. Simple and deterministic matrix sketching. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'13)*. ACM, 581–588.

Yongsub Lim, U. Kang, and Christos Faloutsos. 2014. SlashBurn: Graph compression and mining beyond caveman communities. *IEEE Trans. Knowl. Data Eng.* 26, 12 (2014), 3077–3089.

Shou-De Lin, Mi-Yen Yeh, and Cheng-Te Li. 2013. Sampling and summarization for social networks. In *Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'13)*.

Xuemin Lin, Qing Liu, Yidong Yuan, and Xiaofang Zhou. 2003. Multiscale histograms: Summarizing topological relations in large spatial datasets. In *Proceedings of the International Conference on Very Large Databases (VLDB'03)*. 814–825.

Yu-Ru Lin, Hari Sundaram, and Aisling Kelliher. 2008. Summarization of social activity over time: People, actions and concepts in dynamic networks. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'08)*. 1379–1380.

Bing Liu, Wynne Hsu, and Yiming Ma. 1999. Pruning and summarizing the discovered associations. In *Proceedings of the 5th ACM SIGKDD International Conference Knowledge Discovery and Data Mining (KDD'99)*. 145–154.

Chunyang Liu and Ling Chen. 2016. Summarizing uncertain transaction databases by probabilistic tiles. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'16)*. IEEE, 4375–4382.

Wei Liu, Andrey Kan, Jeffrey Chan, James Bailey, Christopher Leckie, Jian Pei, and Ramamohanarao Kotagiri. 2012. On compressing weighted time-evolving graphs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'12)*. ACM, 2319–2322.

Yike Liu, Neil Shah, and Danai Koutra. 2015. An empirical comparison of the summarization power of graph clustering methods. arXiv:1511.06820.

Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M. Hellerstein. 2012. Distributed graphlab: A framework for machine learning and data mining in the cloud. *Proc. VLDB Endow.* 5, 8 (2012), 716–727.

Antonio Maccioni and Daniel J. Abadi. 2016. Scalable pattern matching over compressed graphs via dedensification. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'16)*. ACM, 1755–1764.

Arun S. Maiya and Tanya Y. Berger-Wolf. 2010. Sampling community structure. In *Proceedings of the 25th International Conference Conference on the World Wide Web (WWW'10)*. ACM, 701–710.

M. Mampaey, J. Vreeken, and N. Tatti. 2011. *Summarizing Data with Itemsets Using Maximum Entropy Models*. Technical Report 2011/02. University of Antwerp.

Sebastian Maneth and Fabian Peternek. 2016. Compressing graphs by grammars. In *Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering (ICDE'16)*. IEEE, 109–120.

Koji Maruhashi, Fan Guo, and Christos Faloutsos. 2011. Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In *Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining*. 203–210.

Hossein Maserrat and Jian Pei. 2010. Neighbor query friendly compression of social networks. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'10)*.

Hossein Maserrat and Jian Pei. 2012. Community preserving lossy compression of social networks. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'12)*. IEEE, 509–518.

Michael Mathioudakis, Francesco Bonchi, Carlos Castillo, Aristides Gionis, and Antti Ukkonen. 2011. Sparsification of influence networks. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'11)*. 529–537.

Yasir Mehmood, Nicola Barbieri, Francesco Bonchi, and Antti Ukkonen. 2013. Csi: Community-level social influence analysis. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 48–63.

Pauli Miettinen and Jilles Vreeken. 2011. Model order selection for boolean matrix factorization. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'11)*. 51–59.

Pauli Miettinen and Jilles Vreeken. 2014. MDL4BMF: Minimum description length for Boolean matrix factorization. *ACM Trans. Knowl. Discov. Data* 8, 4 (2014), 1–30.

Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. 2008. Graph summarization with bounded error. In *Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD'08)*. 419–432.

Mark E. J. Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Phys. Rev. E* 69, 2 (2004), 026113+.

Carlos Ordonez, Norberto Ezquerra, and Cesar A. Santana. 2006. Constraining and summarizing association rules in medical data. *Knowl. Inf. Syst.* 9, 3 (2006), 259–283.

Themis Palpanas, Michail Vlachos, Eamonn J. Keogh, and Dimitrios Gunopulos. 2008. Streaming time series summarization using user-defined amnesic functions. *IEEE Trans. Knowl. Data Eng.* 20, 7 (2008), 992–1006.

Jia-Yu Pan, Hyung-Jeong Yang, and Christos Faloutsos. 2004. MMSS: Multi-modal story-oriented video summarization. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'04)*. 491–494.

Jian Pei, Daxin Jiang, and Aidong Zhang. 2005. On mining cross-graph quasi-cliques. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'05)*. 228–238.

David Peleg and Alejandro A. Schäffer. 1989. Graph spanners. *J. Graph Theory* 13, 1 (1989), 99–116.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'14)*. ACM, 701–710.

Robert Pienta, Acar Tamersoy, Hanghang Tong, and Duen Horng Chau. 2014. MAGE: Matching approximate patterns in richly-attributed graphs. In *Proceedings of the 2014 IEEE International Conference on Big Data*. 585–590.

B. Aditya Prakash, Jilles Vreeken, and Christos Faloutsos. 2012. Spotting culprits in epidemics: How many and which ones? In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'12)*. IEEE.

M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. S. Subrahmanian. 2014. Fast influence-based coarsening for large networks. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'14)*. ACM, 1296–1305.

Qiang Qu, Siyuan Liu, Christian S. Jensen, Feida Zhu, and Christos Faloutsos. 2014. Interestingness-driven diffusion process summarization in dynamic networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD'14)*. 597–613.

Davood Rafiei and Stephen Curial. 2005. Effectively visualizing large networks through sampling. In *Proceedings of the 16th IEEE Visualization Conference (VIS'05)*. 48.

Sriram Raghavan and Hector Garcia-Molina. 2003. Representing web graphs. In *Proceedings of the 2003 IEEE International Conference on Data Engineering (ICDE'03)*. IEEE, 405–416.

Xuguang Ren and Junhu Wang. 2015. Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs. *Proc. VLDB Endow.* 8, 5 (2015), 617–628.

Leonardo F. R. Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'17)*. ACM, 385–394.

Matteo Riondato, David García-Soriano, and Francesco Bonchi. 2014. Graph summarization with quality guarantees. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'14)*. IEEE, 947–952.

Ryan Rossi, Brian Gallagher, Jennifer Neville, and Keith Henderson. 2012. Role-dynamics: Fast mining of large dynamic networks. In*Proceedings of the 25th International Conference Companion on the World Wide Web (WWW'12 Companion)*. ACM, 997–1006.

T. Safavi, C. Sripada, and D. Koutra. 2017. Scalable hashing-based network discovery. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'17)*. 405–414.

Barna Saha and Pabitra Mitra. 2007. Dynamic algorithm for graph clustering using minimum cut tree. In *Proceedings of the SIAM International Conference on Data Mining (SDM'07)*. 581–586.

Neil Shah, Danai Koutra, Lisa Jin, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2017. On summarizing large-scale dynamic graphs. *IEEE Data Eng. Bull.* 40, 3 (2017), 75–88.

Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. TimeCrunch: Interpretable dynamic graph summarization. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'15)*.

P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. 2003. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Res.* 13, 11 (2003), 2498.

Umang Sharan and Jennifer Neville. 2008. Temporal-relational classifiers for prediction in evolving domains. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'08)*. 540–549.

Z. Shen, K.-L. Ma, and T. Eliassi-Rad. 2006. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *IEEE Trans. Vis. Comput. Graph.* 12, 6 (2006), 1427–1439.

Lei Shi, Hanghang Tong, Jie Tang, and Chuang Lin. 2015. VEGAS: Visual influence graph summarization on citation networks. *IEEE Trans. Knowl. Data Eng.* 27, 12 (2015), 3417–3431.

Ben Shneiderman. 2008. Extreme visualization: Squeezing a billion records into a million pixels. In *Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD'08)*.

Mahsa Shoaran, Alex Thomo, and Jens H. Weber-Jahnke. 2013. Zero-knowledge private graph summarization. In *Proceedings of the IEEE International Conference on Big Data*. IEEE, 597–605.

Koen Smets and Jilles Vreeken. 2011. The odd one out: Identifying and characterising anomalies. In *Proceedings of the SIAM International Conference on Data Mining (SDM'11)*. 804–815.

Qi Song, Yinhui Wu, and Xin Luna Dong. 2016. Mining summaries for knowledge graph search. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'16)*. 1215–1220.

Sucheta Soundarajan, Acar Tamersoy, Elias B. Khalil, Tina Eliassi-Rad, Duen Horng Chau, Brian Gallagher, and Kevin Roundy. 2016. Generating graph snapshots from streaming edge data. In *Proceedings of the 25th International Conference Companion on the World Wide Web*. 109–110.

Daniel A. Spielman and Nikhil Srivastava. 2011. Graph sparsification by effective resistances. *SIAM J. Comput.* 40, 6 (2011), 1913–1926. https://doi.org/10.1137/080734029

Olaf Sporns. 2010. *Networks of the Brain*. MIT Press, Cambridge, MA.

Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. 2007. GraphScope: Parameter-free mining of large time-evolving graphs. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'07)*. ACM, 687–696.

Jimeng Sun, Charalampos E. Tsourakakis, Evan Hoke, Christos Faloutsos, and Tina Eliassi-Rad. 2008. Two heads better than one: Pattern discovery in time-evolving multi-aspect data. *Data Min. Knowl. Discov.* 17, 1 (2008), 111–128.

Jimeng Sun, Yinglian Xie, Hui Zhang, and Christos Faloutsos. 2007. Less is more: Compact matrix decomposition for large sparse graphs. In *Proceedings of the SIAM International Conference on Data Mining (SDM'07)*.

Yizhou Sun and Jiawei Han. 2012. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW'15)*. 1067–1077.

Nan Tang, Qing Chen, and Prasenjit Mitra. 2016. Graph stream summarization: From big bang to big crunch. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 1481–1496.

Chayant Tantipathananandh and Tanya Berger-Wolf. 2011. Finding communities in dynamic social networks. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'11)*. IEEE, 1236–1241.

Lini T. Thomas, Satyanarayana R. Valluri, and Kamalakar Karlapalem. 2010. MARGIN: Maximal frequent subgraph mining. *ACM Trans. Knowl. Discov. Data* 4, 3 (2010), 10:1–10:42.

Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. 2008. Efficient aggregation for graph summarization. In *Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD'08)*. ACM, 567–580.

Yuanyuan Tian and Jignesh M. Patel. 2008. TALE: A tool for approximate large graph matching. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*. 963–972.

Hannu Toivonen, Fang Zhou, Aleksi Hartikainen, and Atte Hinkka. 2011. Compression of weighted graphs. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'11)*. 965–973.

Hanghang Tong, Christos Faloutsos, Brian Gallagher, and Tina Eliassi-Rad. 2007. Fast best-effort pattern matching in large attributed graphs. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 737–746.

Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes. 2011. Krimp: Mining itemsets that compress. *Data Min. Knowl. Disc.* 23, 1 (2011), 169–214.

Bianca Wackersreuther, Peter Wackersreuther, Annahita Oswald, Christian Böhm, and Karsten M. Borgwardt. 2010. Frequent subgraph discovery in dynamic networks. In *Proceedings of the 8th Workshop on Mining and Learning with Graphs*. ACM, 155–162.

Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'16)*.

Jianyong Wang and George Karypis. 2004. SUMMARY: Efficiently summarizing transactions for clustering. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'04)*. 241–248.

S. Wasserman and J. Galaskiewicz. 1994. *Advances in Social Network Analysis: Research in the Social and Behavioral Sciences*. SAGE Publications.

Ye Wu, Zhinong Zhong, Wei Xiong, and Ning Jing. 2014. Graph summarization for attributed graphs. In *Proceedings of the International Conference on Information Science, Electronics, and Electrical Engineering (ISEEE'14)*. IEEE, 503–507.

Yang Xiang, Ruoming Jin, David Fuhry, and Feodor Dragan. 2010. Summarizing transactional databases with overlapped hyperrectangles. *Data Min. Knowl. Disc.* 23, 2, 215–251.

Kevin S. Xu, Mark Kliger, and Alfred O. Hero III. 2011. Tracking communities in dynamic social networks. In *Proceedings of the 4th International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction (SBP'11)*. 219–226.

Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2012. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD'12)*. ACM, 505–516.

Xifeng Yan, Hong Cheng, Jiawei Han, and Dong Xin. 2005. Summarizing itemset patterns: A profile-based approach. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD'05)*. 314–323.

Jaewon Yang and Jure Leskovec. 2013. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM'13)*. ACM, 587–596.

Jaewon Yang, Julian McAuley, and Jure Leskovec. 2013. Community detection in networks with node attributes. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'13)*. IEEE, 1151–1156.

Liu Yang, Susan T. Dumais, Paul N. Bennett, and Ahmed Hassan Awadallah. 2017. Characterizing and predicting enterprise email reply behavior. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'17)*. ACM, 235–244.

Jinguo You, Qiuping Pan, Wei Shi, Zhipeng Zhang, and Jianhua Hu. 2013. Towards graph summary and aggregation: A survey. In *Social Media Retrieval and Mining*. Springer, 3–12.

Ning Zhang, Yuanyuan Tian, and Jignesh M. Patel. 2010. Discovery-driven graph summarization. In *Proceedings of the 2003 IEEE International Conference on Data Engineering (ICDE'10)*. 880–891.

Peixiang Zhao, Charu C. Aggarwal, and Min Wang. 2011. gSketch: On query estimation in graph streams. *Proc. VLDB Endow.* 5, 3 (2011), 193–204.

Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. *Proc. VLDB Endow.* 2, 1 (2009), 718–729.

Linhong Zhu, Majid Ghasemi-Gol, Pedro Szekely, Aram Galstyan, and Craig A. Knoblock. 2016. Unsupervised entity resolution on multi-type graphs. In *Proceedings of the International Semantic Web Conference*. 649–667.

Received December 2016; revised January 2018; accepted February 2018