Interactive Visual Graph Mining and Learning

RYAN A. ROSSI, Adobe Research, USA
NESREEN K. AHMED, Intel Labs, USA
RONG ZHOU, Google, USA
HODA ELDARDIRY, PARC, USA

This paper presents a platform for interactive graph mining and relational machine learning called GraphVis. The platform combines interactive visual representations with state-of-the-art graph mining and relational machine learning techniques to aid in revealing important insights quickly as well as learning an appropriate and highly predictive model for a particular task (*e.g.*, classification, link prediction, discovering the roles of nodes, finding influential nodes). Visual representations and interaction techniques and tools are developed for simple, fast, and intuitive real-time interactive exploration, mining, and modeling of graph data. In particular, we propose techniques for interactive relational learning (*e.g.*, node/link classification), interactive link prediction and weighting, role discovery and community detection, higher-order network analysis (via graphlets, network motifs), among others. GraphVis also allows for the refinement and tuning of graph mining and relational learning methods for specific application domains and constraints via an end-to-end interactive visual analytic pipeline that learns, infers, and provides rapid interactive visualization with immediate feedback at each change/prediction in real-time. Other key aspects include interactive filtering, querying, ranking, manipulating, exporting, as well as tools for dynamic network analysis and visualization, interactive graph generators (including new block model approaches), and a variety of multi-level network analysis techniques.

CCS Concepts: • **Mathematics of computing → Graph algorithms**; **Approximation algorithms**; *Combinatorics*; *Graph theory*; • **Information systems → Data mining**; • **Human-centered computing → Visualization**; **Visual analytics**; **Web-based interaction**; **Interactive systems and tools**; *Collaborative and social computing*; • **Theory of computation → Graph algorithms analysis**; **Streaming, sublinear and near linear time algorithms**; **Parallel algorithms**; • **Computing methodologies → Artificial intelligence**; **Machine learning**; **Logical and relational learning**; • **Networks** → Network types;

Additional Key Words and Phrases: Statistical relational learning, interactive relational machine learning, interactive visual graph mining, network analysis, visual graph analytics, interactive network visualization, interactive graph learning, higher-order network analysis, interactive role discovery, link prediction, node embeddings, interactive graph generation, rapid visual feedback, direct manipulation, real-time performance

1 INTRODUCTION

Graph mining and relational learning play a prominent role in revealing important insights and modeling phenomena across a variety of domains including social, behavioral, biological, transportation, entertainment, and financial domains [12, 30].[1] This paper presents a fast, flexible, and

[1]Supplementary Materials: Figs. S1 to S11; Tables S1 to S4; and Alg. S1. Video Demo: https://youtu.be/3cCa2jQVb2o

Authors' addresses: Ryan A. Rossi, Adobe Research, 345 Park Ave, San Jose, CA, 95110, USA, rrossi@adobe.com; Nesreen K. Ahmed, Intel Labs, 3065 Bowers Ave, Santa Clara, CA, 95052, USA, nesreen.k.ahmed@intel.com; Rong Zhou, Google, 1600 Amphitheatre Pkwy, Mountain View, CA, 94043, USA, rongzhou@google.com; Hoda Eldardiry, PARC, 3333 Coyote Hill Rd, Palo Alto, CA, 94304, USA, heldardiry@parc.com.

2157-6904/2018/0-ART1 $15.00
https://doi.org/0000001.0000001

completely interactive visual graph mining and machine learning platform called GraphVis for real-time exploration and understanding of complex graph data.[2] The goal of GraphVis is to enable users to quickly obtain the important insights/content from the graph data (*e.g.*, for decision-making, planning) with minimum effort by taking advantage of state-of-the-art *graph mining and learning techniques* paired with a variety of useful *visual representations of the graph data* along with easy-to-use and intuitive *interaction techniques*. It is designed to allow users to quickly explore and understand graph data via intuitive visual representations and easy-to-use interaction techniques to explore it in a free-flowing manner in real-time (Fig. 1(b)). GraphVis allows users to literally move from raw data to insights within seconds by a simple drag-and-drop of a graph file into the browser [6]. Unlike other network visualization software (such as Gephi [13] and Tulip [11]) that requires installation and updates, GraphVis is web-based working directly from the browser. Furthermore, another key difference is that GraphVis is designed to be consistent with the way humans learn via *immediate-feedback* upon every user interaction (*e.g.*, change of a slider for filtering) [2, 6, 62]. Thus, users have rapid, incremental, and reversible control over all graph queries with *immediate* and *continuous visual feedback*. GraphVis strongly follows the rules of direct manipulation [59] and dynamic (visual) querying [2, 58]. Moreover, existing work has mainly focused on the network visualization (*e.g.*, visualizing the structure) and interaction techniques over this visual representation of the nodes and edges. In contrast, this work focuses on developing state-of-the-art interactive graph mining and relational machine learning (RML) algorithms (that often output new representations of the graph data) along with a variety of useful and intuitive visual representations and easy-to-use interaction techniques for identifying important insights from the original graph data as well as the other data representations given as output by the state-of-the-art graph algorithms.

While most work in visual analytics has focused on traditional non-relational data [41, 62], GraphVis is the first *interactive visual graph analytics* platform with real-time interactive visual graph mining and relational machine learning techniques and tools for rapid interactive visual exploration and insight discovery in graphs. Moreover, it is also web-based making it simple and easy-to-use within seconds. We summarize a few additional important and differentiating aspects of GraphVis below.[3] Other key differences are summarized in Table 1 and discussed in Section 2.

- State-of-the-art interactive visual graph mining techniques are developed including *interactive visual role discovery and community detection* as well as other advanced interactive network analysis methods such as *interactive higher-order network analysis*, *k-truss (triangle-core) decomposition*, among others.
- Interactive visual relational learning techniques for predictive modeling of nodes and edges as well as interactive visual link prediction and weighting methods.
- Interactive *visual* graph filtering and querying capabilities for real-time visual exploration of graphs. GraphVis allows the user to query the graph visually by directly selecting the subgraph(s) of interest in the visualization windows. Alternatively, the user may also query the graph using filters based on feature conditions to select interesting nodes, links, or subgraphs.[4] After each graph manipulation is performed such as inserting, deleting, or querying nodes/links (via filtering, or visually by direct selection), all macro and micro graph properties (*e.g.*, triangles, graphlets, k-core) are automatically updated in real-time.

[2]GraphVis is not to be confused with the graph drawing and visualization software called GraphViz [24].
[3]Note that a fundamental difference between GraphVis and existing systems is the simplicity and degree of interactivity provided in GraphVis, as the techniques and tools in GraphVis are designed for rapid interactive visual feedback after each human interaction/event.
[4]Note GraphVis does not allow queries that involve finding an arbitrary subgraph S in the graph G

- Interactive visual graph generation including newly proposed block model approaches that capture community structure. Nodes, edges, subgraph patterns (*e.g.*, k-cliques, k-stars, k-cycles) or even probabilistic patterns generated via a graph model are easily added to the existing graph by directly interacting with the visualization.
- Novel multi-level graph properties and transformation techniques are provided in GraphVis along with multiple visual representations and interaction techniques (*e.g.*, direct manipulation) for real-time exploration and modeling of the various types of graph data in an intuitive and free-flowing fashion with immediate visual feedback.

To aid the analytic process, both nodes and links may be *colored* and *sized* according to a variety of network properties (*e.g.*, k-core number, eccentricity) or customized by the user. All visualizations are interactive and support direct manipulation, brushing, linking, zooming, panning, tooltips, among others [65, 69]. Networks may also be searched via textual query (*e.g.*, node name). To analyze the microscopic (local) properties of nodes, links, and subgraphs (*e.g.*, triangles, graphlets, k-core), one can simply select the ones of interest directly in the visualization window. Subgraphs may be directly selected by brushing over interesting regions of the network visually. Multiple selections from different regions of the graph are also supported. Selected nodes, links, and subgraphs may be removed, induced, or even moved by dragging them to the desired location. All macro (global) and microscopic (local) graph properties are automatically updated in an efficient manner after each graph manipulation is performed such as inserting, deleting, or filtering of nodes/links. Node and link information may also be updated easily via double-clicking the node/link. There are many other features including full customization of the visualization (color, size, opacity, background, fonts, and so on), text annotation, graph layouts, collision detection, fish eye, and many others. Visualizations can be exported easily as high-quality images (SVG, PNG) as well as (transformed/filtered) graph data and attributes. Drag-and-drop graph file(s) to quickly visualize and interactively explore networks in seconds. A wide variety of graph formats are also supported by GraphVis including edge-lists (txt, csv, tsv, mtx), XML-based formats (gexf, graphml), and a variety of others.

1.1 Scope of this Article

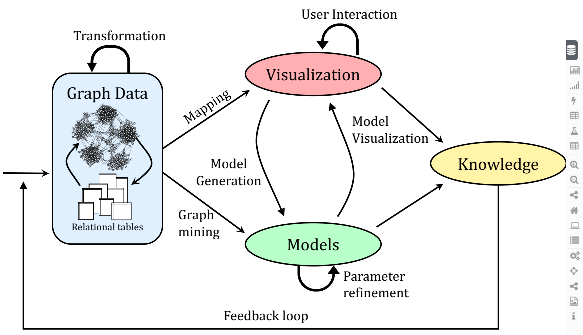
This article focuses on developing an *interactive visual graph analytics* platform that supports real-time interactive visual graph mining and relational machine learning. More formally,

Problem 1 (**Interactive Visual Graph Mining and Learning**). *Interactive visual graph mining and relational learning seamlessly combines visualization, interaction techniques, and state-of-the-art algorithms for graph mining and machine learning. Desiderata for such a platform includes real-time performance, support for dynamic and interactive visual graph queries with direct manipulation and immediate continuous visual feedback as well as support for interactive exploration and modeling using multiple simultaneous interactive visual representations of the graph data.*

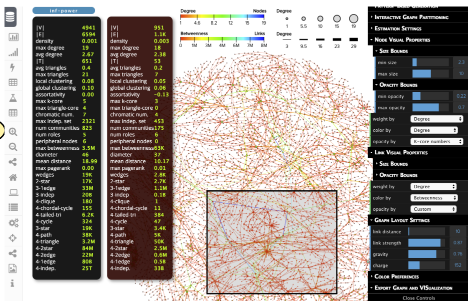
A general overview of the process is shown in Fig. 1(a) [62]. Note existing systems lack many of the key components, properties, and algorithms that lie at the heart of the interactive visual graph mining and relational learning problem. Table 1 provides a comparison of existing systems with respect to the problem above which is the focus of this work. Notably, developing such a system requires many novel components as well as overcoming many algorithmic challenges.

2 RELATED WORK

Table 1 provides a qualitative and quantitative comparison of GraphVis to previous systems. Most systems are mainly focused on either graph visualization [11, 13, 17] or graph algorithms for offline analysis (with no interactive and visual components) [21, 32, 44] and therefore are designed to solve entirely different problems. In comparison, GraphVis is designed for *interactive visual graph*



(a) Interactive visual graph mining & learning overview



|V|	4941	|V|	951
|E|	6594	|E|	1.1K
density	0.001	density	0.003
max degree	19	max degree	18
avg degree	2.67	avg degree	2.38
|T|	651	|T|	53
avg triangles	0.4	avg triangles	0.2
max triangles	21	max triangles	7
local clustering	0.08	local clustering	0.05
global clustering	0.10	global clustering	0.06
assortativity	0.00	assortativity	−0.13
max k-core	5	max k-core	3
max triangle-core	4	max triangle-core	0
chromatic num.	7	chromatic num.	4
max indep. set	2321	max indep. set	453
num communities	823	num communities	175
num roles	5	num roles	6
peripheral nodes	6	peripheral nodes	0
max betweenness	3.5M	max betweenness	63K
diameter	46	diameter	37
mean distance	18.99	mean distance	10.17
max pagerank	0.00	max pagerank	0.01
wedges	19K	wedges	2.8K
2-star	17K	2-star	2.7K
3-1edge	33M	3-1edge	1.1M
3-indep	20B	3-indep	0.1B
4-clique	180	4-clique	1
4-chordal-cycle	155	4-chordal-cycle	11
4-tailed-tri	6.2K	4-tailed-tri	384
4-cycle	324	4-cycle	47
3-star	19K	3-star	3.4K
4-path	38K	4-path	5K
4-triangle	3.2M	4-triangle	50K
4-2star	84M	4-2star	2.5M
4-2edge	22M	4-2edge	0.6M
4-1edge	80B	4-1edge	0.5B
4-indep.	25T	4-indep.	33B

Interactive Graph Partitioning
Estimation Settings
Node Visual Properties
Size Bounds
min size 2.3
max size 10
Opacity Bounds
min opacity 0.22
max opacity 0.7
weight by Degree
color by Degree
opacity by K-core numbers
Link Visual Properties
Size Bounds
Opacity Bounds
weight by Degree
color by Betweenness
opacity by Custom
Graph Layout Settings
link distance 10
link strength 0.87
gravity 0.76
charge 152
Color Preferences
Export Graph and VISualization
Close Controls

(b) GraphVis overview

Fig. 1. (a) Overview of the interactive visual graph mining and learning process and (b) is an overview of GraphVis using the US power grid [64] to explore the vulnerability of the 4,941 power stations with 6,594 lines to various attacks or power failures.

mining and learning problem defined in Section 1.1 which combines visualization and interaction techniques with state-of-the-art graph mining and relational machine learning algorithms.

To the best of our knowledge, GraphVis is the first *interactive visual graph mining and relational learning platform* to seamlessly combine state-of-the-art graph mining and machine learning algorithms with visualization and interaction techniques for real-time exploration, understanding, and modeling of graph data. Previous work has mainly focused on offline software/tools that require downloading and installing a piece of software as well as training (tutorials) to use it [11, 13, 17]. For instance, Gephi [13] is a java-based software package for network visualization built using OpenGL whereas Tulip [11] is a C++ software package. Other examples of offline (non-web) platforms include Osprey [17], among most other platforms summarized in Table 1. However, GraphVis is the first such web-based visual graph analytics platform with the range and scope of interactive visual graph mining *and* machine learning algorithms. The systems closest to GraphVis are Cytoscape [60], Tulip [11], and Gephi [13]. However, these systems are not easy-to-use/intuitive, are focused almost entirely on graph visualization, and are not well-suited for interactive visual graph mining and relational machine learning. In fact, these systems lack support of key components in GraphVis such as interactive visual relational machine learning algorithms (including both classification and link prediction/weighting) and many important interactive visual graph mining algorithms as well as important graph properties such as graphlets for higher-order network analysis. Moreover, GraphVis supports direct manipulation [59] and dynamic (visual) querying [2, 58] in nearly all visual representations of the graph structure, attributes, and features derived from GraphVis directly. In contrast, Gephi [13] and many other systems do not support dynamic visual graph queries with rapid and immediate visual feedback, and have many other key differences summarized in Table 1. Other systems that also focus solely on network visualization include Osprey [17], Pajek [14], and BrainNet [67]. All of these network visualization systems do not focus on interactive visual graph mining & relational learning. Note that other graph mining libraries such as Stanford SNAP [44], iGraph [21] and NetworkX [32] are mainly focused on graph algorithms and are not interactive nor visual and therefore outside the scope of this work. In particular, these systems are not designed for interactive visual graph analytics (Figure 1(a)). One recent system called g-Miner [18] has focused on interactive visual group mining. However, that work focuses only on a subproblem of *community identification* for multivariate graph data.

Table 1. Qualitative and quantitative comparison of GraphVis to other existing systems. For brevity, the prefix "interactive visual" has been removed from each component (column). Note ✓ indicates some support though the system lacks one or more components required for an interactive visual graph analytics system such as interactive visualization, interaction techniques, visual feedback, direct manipulation, real-time performance.

System	Multi-level Features	Higher-order Analysis	Dynamic Network Analysis	Visual Graph Filtering	Community Detection	Role Discovery	Visual Pattern Generation	Graph Generation	Block Models	Relational Classification	Collective Classification	Link Prediction	Link Weighting	Web-based	Interactive Visualization	Direct Manipulation	Dynamic Queries	Rapid Visual Feedback	Multiple Visual Rep.	Estimation Methods	Parallel Algorithms	Node-link diagrams	Scatter plot matrices	Node & link data tables	Global statistics panels	Node & link tooltips	Distribution & line plots
GraphVis	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Pajek [14]	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗
Gephi [13]	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✓	✗	✗	✓	✗	✓	✓	✗	✓
Tulip [11]	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓	✗	✓	✓
Osprey [17]	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗
Cytoscape [60]	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗	✗	✓	✗
BrainNet [67]	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
g-Miner [18]	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✓	✗	✗	✓	✗	✓	✗	✓	✗

"Multi-level features" holds true if the system makes use of both global and local (node/link-based) features as well as global and local statistics and distributions for interactive visual graph mining. "Higher-order Network Analysis" holds true if the system uses higher-order graphlet (subgraph) features. "Dynamic Network Analysis" holds true if the system provides techniques and tools for dynamic network analysis. "Visual Graph Filtering" holds true if the system allows the user to filter the graph via direct manipulation of the visual representation (*e.g.*, simple lasso-selections of nodes/links). "Community Detection" holds true if the system leverages community detection algorithms, whereas "Role Discovery" holds true if the system leverages role discovery for interactive visual graph mining. "Visual Pattern Generation" holds true if users can seamlessly add (complex) patterns to the graph visually using direct interactive manipulation techniques. "Graph Generation" holds true if users can synthetically generate graphs, whereas "Block Models" holds true if users can generate graphs using block model approaches. "Relational Classification" holds true if the system provides interactive visual relational classification capabilities (using input attributes and structural features), whereas "Collective Classification" is true if interactive visual collective classification (semi-supervised learning) capabilities are provided. "Link Prediction" holds true if the system allows users to interactively predict links, whereas "Link Weighting" holds true if users can interactively estimate the weight/strength of existing links in the graph. "Web-based" holds true if the system runs in a web browser. "Interactive Visualization" holds true if the system provides real-time interactive visualization capabilities where users interact directly with the visualization and changes to the visualization are incorporated in a timely fashion. "Direct Manipulation" holds true if users can interact directly on the visualizations with immediate and continuous visual feedback. "Dynamic Queries" holds true if the system allows users to formulate (interactive) queries over the visual representation of the data. "Rapid Visual Feedback" holds true if the system provides rapid visual feedback in real-time after the user initiates a (visual interactive) query. "Multiple Visual Representations" holds true if the system uses multiple visual representations of the graph data. "Estimation Methods" holds true if statistical estimation methods are used for real-time performance. "Parallel Algorithms" holds true if parallel algorithms are used. The remaining rows hold true if the system uses the particular interactive visual representation.

Besides systems, there has also been a significant amount of research into visualization and interaction techniques for graphs [33, 36, 38, 42]. Most work has focused on representing graph data using node-link diagrams which require layout algorithms for computing the position of nodes (and links) [27, 28, 66]. Recently, fast approximation techniques have also been proposed for visualizing large graphs, see Hu and Shi [38] for a recent survey. Other work has focused on reduce the visual clutter (of a graph layout) arising from the size of the graph by transforming the graph layout using filtering [39, 63] and bundling [22, 29, 35] techniques. Notably, any of the above graph layout or clutter reduction algorithms can be used by GraphVis. Besides the interactive node-link visualization, GraphVis also leverages multiple interactive visual representations of the graph data including a variety of interactive scatter plot matrices for both the node and link features, interactive data tables, interactive distribution plots, and others. See Table 1 for a summary.

GraphVis is also the first such interactive visual graph analytics platform that leverages statistical estimation for approximating computationally intensive graph algorithms to ensure real-time performance on large graphs. Most estimation methods use some form of sampling [5, 61]. For instance, Stefani et al. [61] proposed an approach that uses reservoir sampling to estimate triangles in an edge stream. More recently, Ahmed et al. [4] proposed order-based reservoir sampling to estimate a variety of properties from massive graph streams. However, any of the above or future estimation methods that are faster or more accurate can also be easily leveraged by GraphVis to further improve performance.

Previous work in graph generation has mainly focused on *algorithms* for constructing graphs that preserve one or more graph properties such as the degree distribution, triangle counts, and other graph properties [9, 10, 25, 43]; and has largely ignored the problem of interactive visual graph generation. In this work, we introduce and incorporate three general classes of interactive visual graph generators in GraphVis that combine graph generation with interactive visualization and interaction techniques for real-time visual exploration of the generated graphs and models with rapid immediate visual feedback. This allows the user to quickly understand and uncover key insights into the generated graphs and models as well as provides them with a way to quickly add subgraph/probabilistic patterns to existing graphs for simulation and other important use cases.

Temporal graph visualization has also been the focus of recent research [57, 68] The majority of this work focuses mainly on visualization approaches for temporal networks [15, 16, 47] and not on statistical techniques for dynamic network analysis [1, 34].

3 INTERACTIVE GRAPH MINING

3.1 Multi-level Graph Features

Visual analytic tools need to allow for interacting and reasoning across multiple simultaneous scales of data representations [23]. Thus, we developed GraphVis with a multi-scale visual analytics engine to support (visual) interactive network exploratory analysis at both the global macro-level as well as the local microscopic level. Visual graph mining and machine learning techniques lie at the heart of GraphVis and provide the analysts with a set of powerful tools to discover key insights and reveal important structural patterns interactively in real-time. Such an approach is vital for interactively exploring big data in real-time by summarizing its patterns, statistics (binning, distributions, etc.), as well as spotting anomalies. Statistical techniques are used to find interesting nodes, allowing the user to sort through the top-k most interesting nodes for further investigation. Table S4 provided in the supplementary material summarizes the computational properties of many multi-level graph features used in GraphVis.

Every update, insertion, or deletion of a node, edge, or subgraph is immediately reflected in the visualization window. Furthermore, the visualization and analytics are also updated immediately upon any parameter change via sliders or other interface controls. This allows to quickly test a hypothesis as well as investigate the impact of certain actions on the network structure and its properties/statistics. For instance, suppose we use betweenness to filter the graph, as we adjust the slider, the analyst receives visual feedback immediately at each change in the slider (in contrast to adjusting the slider to the desired value, then receiving feedback on the selection).

3.1.1 Macro-level Interactive Graph Analysis. At the macroscopic (global) level, we use a variety of key network properties. A few of these include max/mean degree, total triangles, global clustering, max k-core number, diameter, mean distance, approx. chromatic number, number of communities/roles, and max k-truss (triangle-core) number [50]. To help guide the interactive exploration, we display many of the important macro properties that help characterize the global structure of the network and any selected subgraphs (see the left-most panels in Fig. 1(b) and Fig. 6(a)). Statistical aggregates (mean, max, mode, sum, var.) are also provided to aid in understanding the graph.

3.1.2 Micro-level Interactive Node & Edge Analysis. To facilitate the discovery process, GraphVis provides interactive exploration at the microscopic (local) level, *e.g.*, using edge and node degree, eccentricity, k-core number, and k-truss (triangle-core) number. In addition, many important social network analysis measures are used in the interactive visual analytics including betweenness, number of triangles, clustering coefficient, path lengths, PageRank, and many others. These node and link properties are displayed in visual form and can be explored/manipulated directly by the user in a free-flowing manner (*e.g.*, using brushing, linking, zooming, mouseover, filtering, and so on). For instance, the neighborhood of a node (link) can be highlighted as well as its micro-level statistics and properties. See Fig. 6(c) and Fig. S11(a) for node-centric examples and Fig. 7(b) for links. Multiple visual representations of the graph data are also provided. For example, GraphVis leverages an interactive scatter plot matrix tool for analysis of the correlation between pairs of link statistics (Fig. 3) or pairs of node statistics (Fig. S11(b)). Users can also interactively highlight interesting nodes (links) across the various measures, interactively filter them, color and weight them using various properties, among other possibilities. Furthermore, semantic zooming can be used to drill-down in order to understand the differences between individual nodes and links. Many other important measures are also available including k-truss numbers, graphlet counts (*e.g.*, number of 4-cliques incident to a link/node), as well as a variety of others. GraphVis also provides interactive data tables for exploring and searching the graph data (Fig. S4).

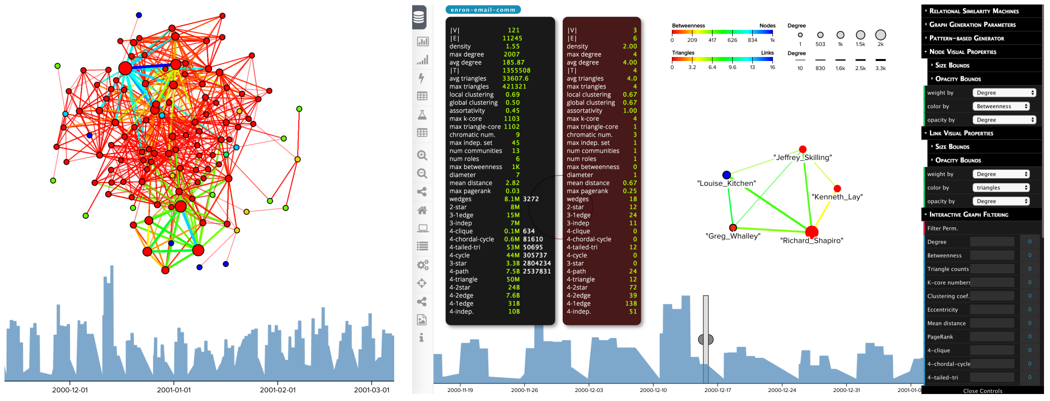
3.1.3 Distributions. Node and link summarization techniques (*e.g.*, binning/histograms, statistical distributions) are used to obtain fast, meaningful and useful data representations. For example, we leverage binning methods to interactively compute and maintain the frequency distribution of some graph properties (*e.g.*, degree distribution) upon any update, insertion, or deletion of a node, edge, or subgraph. Furthermore, we also interactively plot the cumulative distribution function (CDF) and the complementary CDF, which are easily computed from the frequency distribution. These are known to be important for networks, capturing interesting structural properties (*e.g.*, heavy-tailed distributions). Furthermore, we also utilize sampling [3] as well as fast ranking algorithms for displaying top-k nodes, links, and subgraphs to the user for further exploration. In addition to distributions, the macro-level measures are also useful for big graph data and vital to the multi-level strategy offered by GraphVis.

3.2 Interactive Dynamic Network Analysis

Dynamic Network Analysis (DNA) [19] has become increasingly important as most networks are naturally evolving over time [20, 40]. To understand and explore the temporal dependencies that lie at the heart of these networks, we propose interactive visual dynamic network analysis tools. Further, GraphVis provides tools to interactively analyze the evolution and dynamic patterns that arise in graphs over time. See Fig. 2 for an example. We also provide tools to easily and intuitively filter the graph based on time. In particular, the time scale (resolution; *e.g.*, 10 minutes, 1 day) can be selected via brushing and adapted based on the application or data properties. This controls the temporal graph being visualized as shown in Fig. 2(b). Using these tools, analysts can begin to understand the dynamics and trends present in the network (*e.g.*, seasonality, spike, trends).

4 INTERACTIVE GRAPH QUERYING AND FILTERING

Data is selected in GraphVis using two general approaches that are both intuitive and flexible for a wide variety of use cases and important applications. Nodes, edges, and subgraphs are easily selected by visual inspection (Section 4.1) or by an important graph property or statistic (Section 4.2). Nevertheless, both approaches leverage the real-time interactive visualization to help guide the user to the warranted strategy. As an aside, once the user selects a subset of nodes and links via



|V|	121	
|E|	11245	
density	1.55	
max degree	2007	
avg degree	185.87	
|T|	1355508	
avg triangles	33607.6	
max triangles	421321	
local clustering	0.69	
global clustering	0.50	
assortativity	0.45	
max k-core	1103	
max triangle-core	1102	
chromatic num.	9	
max indep. set	45	
num communities	13	
num roles	6	
max betweenness	1K	
diameter	7	
mean distance	2.82	
max pagerank	0.03	
wedges	8.1M	3272
2-star	8M	
3-1edge	15M	
3-indep	7M	
4-clique	0.1M	634
4-chordal-cycle	0.6M	81610
4-tailed-tri	53M	50695
4-cycle	44M	305737
3-star	3.3B	2804234
4-path	7.5B	2537831
4-triangle	50M	
4-2star	24B	
4-2edge	7.6B	
4-1edge	31B	
4-indep.	10B	

|V|	3
|E|	6
density	2.00
max degree	4
avg degree	4.00
|T|	4
avg triangles	4.0
max triangles	4
local clustering	0.67
global clustering	0.67
assortativity	1.00
max k-core	4
max triangle-core	1
chromatic num.	3
max indep. set	1
num communities	1
num roles	1
max betweenness	0
diameter	1
mean distance	0.67
max pagerank	0.25
wedges	18
2-star	12
3-1edge	24
3-indep	11
4-clique	0
4-chordal-cycle	0
4-tailed-tri	12
4-cycle	0
3-star	0
4-path	24
4-triangle	12
4-2star	72
4-2edge	39
4-1edge	138
4-indep.	51

Node Visual Properties: weight by Degree; color by Betweenness; opacity by Degree

Link Visual Properties: weight by Degree; color by triangles; opacity by Degree

Degree	0
Betweenness	0
Triangle counts	0
K-core numbers	0
Clustering coef.	0
Eccentricity	0
Mean distance	0
PageRank	0
4-clique	0
4-chordal-cycle	0
4-tailed-tri	0

(a) Full network (b) Temporal window (1 day)

Fig. 2. Interactive dynamic network analysis of an email communication network (multigraph) consisting of 121 Enron employees whom send 11245 emails to each other (representing links) over a 3 month period [7]. This data was released to the public by the Federal Energy Regulatory Commission during its investigation into fraud. (a) The full network without using a temporal window to filter the graph. (b) Using a small temporal window, we immediately see many of the major players involved in the Enron scandal.

one or more queries, they can be removed or explored interactively in real-time using a variety of interactive visual graph mining and machine learning techniques.

4.1 Visual Selection and Filtering of Subgraphs

One may select multiple nodes, links, and subgraphs in a visual and completely interactive manner. The process is both easy, intuitive, and fast, and unlike other approaches, one may select a single node or a subgraph of interest with a simple drag of the mouse (Fig. 1(b) and Fig. 6(a)). The simplest means of selecting nodes, links, or subgraphs is to simply press and hold shift key while selecting any arbitrary set of nodes and links. Multiple selections are also easily possible using the same method. For instance, once a subgraphs has been selected, release all keys, then make additional selections by pressing and holding the shift key once again. Nodes (links) can be visually selected using other intuitive visual representations of the data. In particular, besides selecting nodes/edges visually using the node-link diagram directly, one can also leverage other interactive visual representations of the graph data such as the interactive scatter plot matrix tool in GraphVis as shown in Fig. 3. Note that this approach can be viewed as a hybrid between visual filtering and filtering based on graph properties as it combines an intuitive visual representation allowing visual queries based on node/link properties of interest.

4.2 Selection and Filtering via Properties

Besides the intuitive visual filtering (or selection) from Section 4.1, one may also *interactively filter* the graph using a wide range of graph properties, statistics, and queries (as shown in Fig. 6(b)). Notice that a key difference between visual filtering and filtering via node/edge properties is that previously the filtering has been performed manually, *i.e.*, the user manually decided which nodes to select by visual inspection of the graphs general structural patterns (and connectivity) or by investigating the properties of each node to determine its fate or fitness *w.r.t.* the individual's objective. Nodes and edges can be filtered (or selected) interactively based on the node/edge properties. In Fig. 6(b), the user interactively filters nodes and edges from the network visualization by changing a simple slider. In particular, Fig. 6(b) interactively filters all nodes that the model is highly confident about (with uncertainty ≤ 0.50) and therefore leaving only the nodes that the model is most uncertain. Note that any new feature constructed by the user is automatically included in the list of properties that can be used to filter various visual representations of the data.

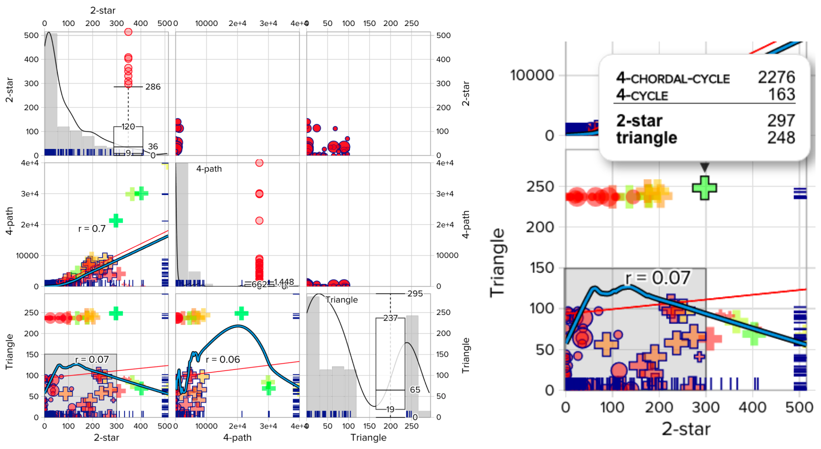
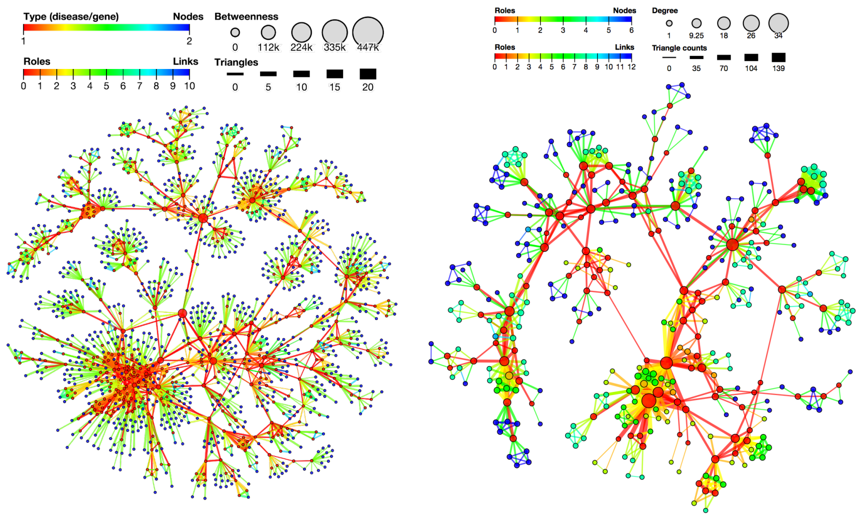


Fig. 3. The *interactive scatter plot matrix* tool is useful for exploring and understanding the data through real-time interactive queries. A rectangle is drawn which corresponds to the data points that fall in the user-defined range. Note that interactive dynamic queries are simple and intuitive to use (by drawing a rectangle via a mouse-click, and drag to adjust the size and position). In this use case we interactively explore a co-authorship network from the Arxiv High Energy Physics category (ca-HepPh) consisting of 8,361 authors with 15,751 co-author links indicating that the two authors collaborated on at least one paper [48]. In particular, the user selects all co-author links with $E_S = \{x_1 \leq 150 \wedge x_2 \leq 300\}$ — *i.e.*, all links with triangle count less than 150 and 4-node stars less than 300. The selected co-author links are highlighted across the interactive scatter plots. Multiple such interactive queries can be configured by simply drawing multiple regions. Data points representing co-author links are colored and sized based on the number of tailed-triangles. For large graphs, sampling is used to select a representative set of points to render (without degrading quality as many points overlap and thus can be removed).

5 INTERACTIVE GRAPH PARTITIONING

GraphVis provides a diverse collection of visual interactive graph partitioning methods. Intuitively, graph partitioning can be categorized into role discovery [51] (Section 5.1) and community detection [26] (Section 5.2). Roles and communities are fundamentally different but complimentary graph concepts. Informally, communities are sets of nodes with more connections inside the set than outside, whereas roles are sets of nodes (or edges) that are more structurally similar to nodes (edges) inside the set than outside. The fundamental difference is that communities are based on connection density (or cohesion, proximity; see [26]) whereas roles are based on structural similarity or equivalence [45] (*e.g.*, nodes that are the center-of-stars/hubs are all assigned to one role). Thus, nodes that share roles (are structurally similar) may be in different communities whereas nodes in the same community are generally close to one another but may have different structural roles. The other difference is that roles capture general node-level (or edge-leve) structural patterns and therefore generalize across networks whereas communities do not. Note that there are other types of graph partitioning methods such as graph coloring that are also incorporated into GraphVis. All graph partitioning methods in GraphVis are designed to be efficient taking at most linear time in the number of edges to compute.



(a) Edge roles correlated with class label (b) Node and edge roles

Fig. 4. **Interactive Role Discovery**. **(a)** We explore the behavioral roles from a network of disorders/diseases and genes linked by known associations [31]. Edges are colored by the role with largest membership. Nodes are colored by class label (disease ● / gene ●). **(b)** Interactive visual mining using node and edge roles of a network science co-authorship graph [49] with 379 scientists from the network science community with 913 links indicating co-authorship, *i.e.*, individuals who authored at least one paper together [49]. For example, authors (nodes) and co-author links that are gate-keepers (bridge roles) connecting different groups of authors are shown in red (*e.g.*, Mark Newman). See text for further discussion.

5.1 Interactive Role Discovery

Using the notion of feature-based roles as a basis [51], we design and implement a few interactive role discovery methods. In particular, discovering feature-based roles has the following key steps: Given the graph G, we first derive an initial feature-based representation (node/edge features) and then group/cluster the features into roles using matrix factorization or k-means. As initial features, we use a variety of graphlet features of size 2, 3, and 4. We also include other features such as in/out/total degree/egonet, k-core number, betweenness, among many others. While GraphVis currently only exploits the above features, we can also discover more complex deeper features using recent graph-based deep learning and graph embedding approaches [56]. Given this feature matrix, we then use k-means or a (non-negative) matrix factorization approach to group the features. The appropriate number of roles can be selected using a model selection criterion or interactively by the user (*e.g.*, by adjusting a simple slider). Fig. 4 demonstrates the effectiveness of the interactive role discovery approach. In particular, Fig. 4(a) visualizes the edge roles learned from a network of disorders/diseases and genes linked by known associations [31]. We observe that edge roles are clearly correlated with the class label of the node. Intuitively, this makes sense since they capture the structural behavior surrounding each edge in the network. The node and edge roles of a network science co-authorship network [49] are shown in Fig. 4(b) and can be easily interpreted.

For instance, the first node and edge role (*i.e.*, the red nodes and edges in Fig. 4(b)) represent a type of gatekeeper/central role where the nodes/edges have high betweenness and are crucial for maintaining the connectivity of the graph (*i.e.*, removing such nodes would break apart the graph into many disconnected components).

5.2 Interactive Community Detection

We designed a fast and effective interactive community detection approach. In particular, the approach begins with each node (or edge) belonging to its own community. For each node $v_i \in V$ (or edge), we assign it to the community $C_k \in C$ that has the maximum number of neighbors $\Gamma(v_i)$ in it. More formally,

$$\arg\max_{C_k \in C} \sum_{v_j \in \Gamma(v_i)} \mathbb{I}_{jk} \tag{1}$$

where $\mathbb{I}_{jk} = 1$ if $v_j \in C_k$, and 0 otherwise. Notice that the above criterion in Eq. 1 can be easily replaced/modified to take into account other important aspects. The algorithm converges when an iteration results in no further changes (*i.e.*, no new assignments are made) or if the max number of iterations is reached which can be interactively tuned by the user. Upon each iteration, we compute a random permutation and use this ordering to assign nodes (or edges) to communities. To further speedup the approach, we leverage the number of previous iterations that the community assignment of a node (or edge) remained unchanged (*i.e.*, the community of v_i remained stable over the last t iterations). In particular, let δ denote a hyperparameter that controls the number of previous iterations that the community assignment of a node or edge must remain unchanged before it is declared as final. Thus, each iteration of the approach can be defined over the set S of graph elements (nodes/edges) that are still active, *i.e.*, $T_i < \delta$ where T_i denotes the number of subsequent iterations that v_i has remained unchanged (*w.r.t.* community assignment). Fast and efficient localized updates are performed when new nodes/edges are added by the user (*e.g.*, using a pattern generator proposed in Section 7.1). In Fig. S10 provided in the supplementary material, links and nodes are colored according to their community membership and can be interactively explored using the wide variety of interactive visual graph mining techniques. For instance, the user can select a community of interest by drawing a rectangular region around it. Once selected, the macro-level/global properties of that community are immediately computed and displayed next to the properties of the full graph to make it easy for the human analyst to interpret and compare. The user can also interactively tune a variety of parameters (*e.g.*, max number of iterations or number of communities) and see the impact immediately in real-time allowing the user to quickly find and explore other community assignments in real-time. Other more computationally intensive methods such as spectral clustering or NMF-based community detection can be easily incorporated into GraphVis.

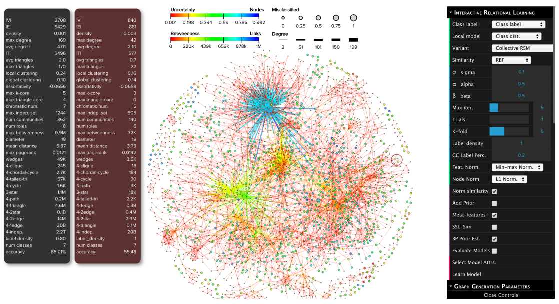
6 INTERACTIVE RELATIONAL MACHINE LEARNING

6.1 Interactive Relational and Collective Classification

Relational Machine Learning (RML) [30] methods exploit the relational dependencies between nodes to improve predictive performance. However, these approaches can often fail in practice due to low relational autocorrelation, noisy links, sparsely labeled graphs, and problems with the graph data representation. To overcome these problems, *interactive relational machine learning* (iRML) techniques are developed in which users interactively specify relational models and data representation (via transformation techniques for the graph structure and features), as well as perform evaluation, analyze errors, and make adjustments and refinements in a closed-loop [54]. In particular, humans interact with relational learning algorithms such as wvRN [46] and Relation Similarity Machines (RSM) [55] by providing input (in the form of labels, similarity/kernel function, hyper-parameters, priors, confidence/uncertainty about particular instances, learning rate, corrections, rankings, probabilities, evaluation) while observing the output (in the form of predictions, uncertainty, feedback, and any useful visual representation of the data). Other algorithms including a relational variant of decision trees and SVM have also been implemented in GraphVis.

GraphVis combines fast, flexible, and powerful relational learning techniques with interactive visualization to aid in evaluation, tuning, analysis of errors, model selection, regularization, semi-supervised information, among others. It is designed for rapid interactive visual exploration and learning through visual representations and interaction techniques at each stage; and enables users to trade off competing goals, encode prior knowledge, understand the model, and analyze errors and uncertainty. These techniques facilitate the design and selection of a relational model (from the space of potential models), their evaluation, error and uncertainty analysis, as well as enable users to interactively refine them in a closed-loop. An overview of the proposed iRML system in GraphVis is shown in Fig. 5. In that example, we first interactively learn a model (for the cora network), then select the misclassified nodes for further analysis. The global statistics of the selected subgraph are shown in the right-most panel. Node color represents the model's uncertainty using an entropy-based measure, whereas the size of the node indicates whether it was correctly classified or not. In Fig. 5, misclassified nodes are given a larger size so that they can easily be identified for further exploration. Uncertainty (and learned class prob. distribution, statistics) of a node or set of nodes may also be displayed by selecting or mousing over the nodes of interest as shown in Fig. 6(c). The iRML system also supports interactive real-time visual graph filtering, *e.g.*, remove all uncertain nodes above a user-specified threshold as shown in Fig. 6(b) and Fig. 6(a). Moreover, all visualizations are interactive and support brushing, linking, zooming, panning, tooltips, and others (Fig. 6(a)). Efficient update rules are also derived to avoid relearning the entire model after each user interaction/visual query. For example, after the deletion (or insertion) of a node, we can update the global relational model via a fast localized update. These local updates enable real-time exploration capabilities by leveraging fast exact or approximate solutions (*e.g.*, to support real-time interactive queries; see Fig. 6(a)).

Many of the iRML components in GraphVis can be quickly explored in real-time using interactive visualization and analytic techniques including the attribute to predict, initial features to use (non-relational and graph-based features), local model for estimation, kernel function (RBF, polynomial), hyperparameters (for selected kernel), node- and feature-wise normalization scheme (*e.g.*, L_1, min-max), as well as whether to use Semi-Supervised Learning (SSL), meta-features (based on current estimates), among others (*e.g.*, see the right panel in Fig. 5). Interactive link prediction methods proposed in Section 6.2 and other important learning components in GraphVis can be leveraged



|V|	2708
|E|	5429
density	0.001
max degree	169
avg degree	4.01
|T|	5496
avg triangles	2.0
max triangles	170
local clustering	0.24
global clustering	0.10
assortativity	-0.0656
max k-core	5
max triangle-core	4
chromatic num.	7
max indep. set	1244
num communities	362
num roles	8
max betweenness	0.9M
diameter	19
mean distance	5.87
max pagerank	0.0121
wedges	49K
4-clique	245
4-chordal-cycle	2.7K
4-tailed-tri	57K
4-cycle	1.6K
3-star	1.1M
4-path	0.2M
4-triangle	4.6M
4-2star	0.1B
4-2edge	14M
4-1edge	20B
4-indep.	2.2T
label density	0.80
num classes	7
accuracy	85.01%

|V|	840
|E|	881
density	0.003
max degree	42
avg degree	2.10
|T|	577
avg triangles	0.7
max triangles	22
local clustering	0.16
global clustering	0.14
assortativity	-0.0658
max k-core	3
max triangle-core	0
chromatic num.	5
max indep. set	505
num communities	140
num roles	6
max betweenness	32K
diameter	19
mean distance	3.79
max pagerank	0.0142
wedges	3.5K
4-clique	16
4-chordal-cycle	184
4-cycle	90
4-path	9K
3-star	18K
4-tailed-tri	2.2K
4-1edge	0.3B
4-2edge	0.4M
4-2star	2.9M
4-triangle	0.1M
4-indep.	20B
label_density	1
num classes	7
accuracy	55.48

INTERACTIVE RELATIONAL LEARNING

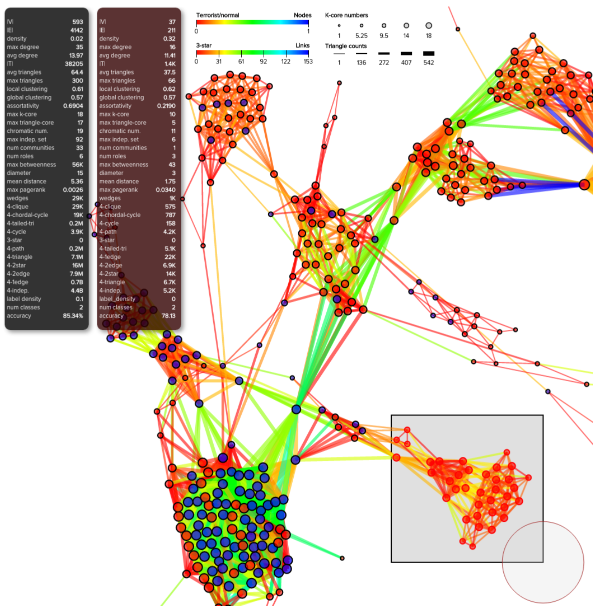
Class label	Class label
Local model	Class dist.
Variant	Collective RSM
Similarity	RBF
σ sigma	0.1
α alpha	0.5
β beta	0.5
Max iter.	5
Trials	1
K-fold	5
Label density	1
CC Label Perc.	0.2
Feat. Norm.	Min-max Norm.
Node Norm.	L1 Norm.
Norm similarity	✓
Add Prior	☐
Meta-features	✓
SSL-Sim	☐
BP Prior Est.	✓
Evaluate Models	☐

Select Model Attrs.
Learn Model
GRAPH GENERATION PARAMETERS
Close Controls

Fig. 5. Interactive Relational ML. We use the cora citiation network [46] consisting of 2,708 papers with 5,429 citation links between them. See text for discussion.

Table 2. Average time per test instance. Note $|V|$ is the number of nodes and V^{ℓ} is the number of labeled nodes.

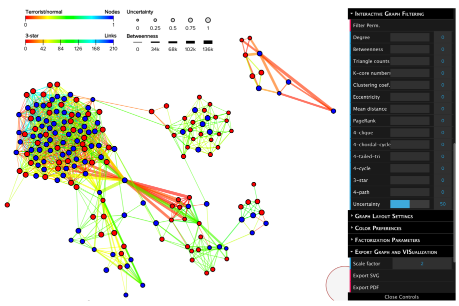
GRAPH	$\|V\|$	$\|E\|$	$\|V^{\ell}\|$	$\|C\|$	AVG. TIME
soc-terror	881	8.5K	90	2	0.18 ms
aff-polbooks	105	441	12	3	0.12 ms
cora	2.7K	5.4K	272	7	0.97 ms
DD6	4.1K	10.3K	417	20	3.10 ms



|V|	593
|E|	4142
density	0.02
max degree	35
avg degree	13.97
|T|	38205
avg triangles	64.4
max triangles	300
local clustering	0.61
global clustering	0.57
assortativity	0.6904
max k-core	18
max triangle-core	17
chromatic num.	19
max indep. set	92
num communities	33
num roles	6
max betweenness	56K
diameter	15
mean distance	5.36
max pagerank	0.0026
wedges	29K
4-clique	29K
4-chordal-cycle	19K
4-tailed-tri	0.2M
4-cycle	3.9K
3-star	0
4-path	0.2M
4-triangle	7.1M
4-2star	16M
4-2edge	7.9M
4-1edge	0.7B
4-indep.	4.4B
label density	0.1
num classes	2
accuracy	85.34%

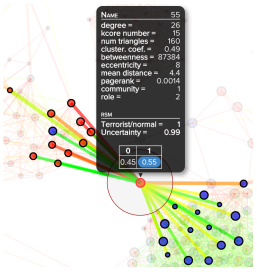
|V|	37
|E|	211
density	0.32
max degree	16
avg degree	11.41
|T|	1.4K
avg triangles	37.5
max triangles	66
local clustering	0.62
global clustering	0.57
assortativity	0.2190
max k-core	10
max triangle-core	5
chromatic num.	11
max indep. set	6
num communities	1
num roles	3
max betweenness	43
diameter	3
mean distance	1.75
max pagerank	0.0340
wedges	1K
4-clique	575
4-chordal-cycle	787
4-cycle	158
4-path	4.2K
3-star	0
4-tailed-tri	5.1K
4-1edge	22K
4-2edge	6.9K
4-2star	14K
4-triangle	6.7K
4-indep.	5.2K
label_density	0
num classes	2
accuracy	78.13

(a) Interactive relational learning



Filter Perm.	
Degree	0
Betweenness	0
Triangle counts	0
K-core numbers	0
Clustering coef.	0
Eccentricity	0
Mean distance	0
PageRank	0
4-clique	0
4-chordal-cycle	0
4-tailed-tri	0
4-cycle	0
3-star	0
4-path	0
Uncertainty	50
Scale factor	2

(b) Interactive filtering via uncertainty



Name	55
degree =	26
kcore number =	15
num triangles =	160
cluster. coef. =	0.49
betweenness =	87384
eccentricity =	8
mean distance =	4.4
pagerank =	0.0014
community =	1
role =	2

RSM

Terrorist/normal =	1
Uncertainty =	0.99

0	1
0.45	0.55

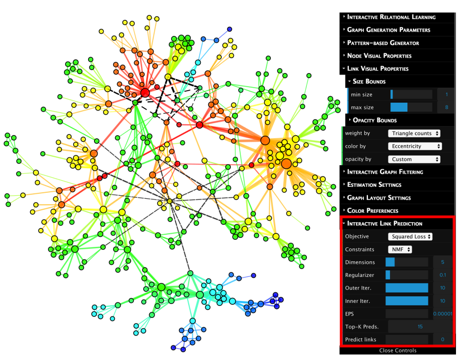
(c) Exploring local confidence

Fig. 6. At the heart of GraphVis lies the ability to interactively learn models in real-time and adjust them accordingly, as well as tune parameters, explore/construct features, among many other possibilities. In this use case, we use a network of 881 individuals including known terrorists, their family members, and other known contacts with 8,592 links between them indicating known associations (soc-terror) [70]. In (a) the user interactively learns a global model (see the leftmost black side panel for stats and accuracy) and then selects a subgraph H of individuals visually by drawing a region around the individuals of interest. A local RSM model is immediately learned for H (in real-time) and stats and accuracy (using same k-fold CV) are reported in the red side panel on the left side. Individuals (nodes) are colored by their class label (terrorist/normal). (b) Interactive filtering via uncertainty. All individuals (nodes) and associations with high confidence are filtered (≤ 0.50), leaving only those individuals (terrorists, family members) that are most uncertain. Relative entropy is used to measure the uncertainty of the learned probability distributions. (c) Interactive visual exploration of the local probability distribution learned for individuals, as well as the uncertainty, and other model and graph features via simple and intuitive local dynamic queries initiated by simply lasso-selecting a node or group of nodes. In (b)-(c) nodes are colored by class label and weighted (sized) by uncertainty.

directly to improve learning and inference of the user-defined iRML method. Fig. 6(a) demonstrates a few other useful features. In particular, the iRML platform in GraphVis is shown to be fast, parallel, space-efficient, amenable to streaming and dynamic queries/updates, and most importantly, naturally supports real-time interactive learning and inference, and provides rapid immediate (and visual) feedback to the user at real-time interactive response times.

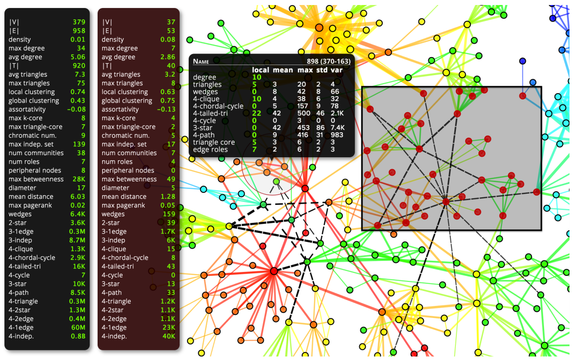
6.2 Interactive Link Prediction and Weighting

The quality and utility of the visualizations and interaction techniques depend on the underlying data representation(s). Moreover, graph data is often unreliable, noisy, and uncertain, and therefore techniques are needed to explicitly model and account for such uncertainty and noise. In this section, we introduce the interactive link prediction and link weighting problems and propose interactive techniques and computational models for discovering an appropriate representation of the data for a given task that reveals the important patterns/information of a large, complex, and potentially dynamic graph. These models are designed to be fast, flexible, and operate at multiple levels of abstraction giving rise to a variety of useful data representations that facilitate visual analysis (by a human analysts in a timely fashion). Furthermore, the visual representations and interaction techniques for link prediction and weighting are shown to be useful for robustifying the graph data, or finding and removing noisy and/or spurious links through efficient exploration of the space of models. All of these techniques facilitate the human analysts ability to understand and reason about complex graph data in a timely fashion.



min size	1
max size	8
weight by	Triangle counts
color by	Eccentricity
opacity by	Custom
Objective	Squared Loss
Constraints	NMF
Dimensions	5
Regularizer	0.1
Outer Iter.	10
Inner Iter.	10
EPS	0.00001
Top-K Preds.	15
Predict links	0

(a) Interactive link prediction



|V|	379	|V|	37
|E|	958	|E|	53
density	0.01	density	0.08
max degree	34	max degree	7
avg degree	5.06	avg degree	2.86
|T|	920	|T|	40
avg triangles	7.3	avg triangles	3.2
max triangles	75	max triangles	8
local clustering	0.74	local clustering	0.63
global clustering	0.43	global clustering	0.75
assortativity	-0.08	assortativity	-0.13
max k-core	8	max k-core	4
max triangle-core	7	max triangle-core	2
chromatic num.	9	chromatic num.	5
max indep. set	139	max indep. set	17
num communities	38	num communities	7
num roles	7	num roles	4
peripheral nodes	8	peripheral nodes	0
max betweenness	28K	max betweenness	49
diameter	17	diameter	5
mean distance	6.03	mean distance	1.28
max pagerank	0.02	max pagerank	0.05
wedges	6.4K	wedges	159
2-star	3.6K	2-star	39
3-1edge	0.3M	3-1edge	1.7K
3-indep	8.7M	3-indep	6K
4-clique	1.3K	4-clique	15
4-chordal-cycle	2.9K	4-chordal-cycle	8
4-tailed-tri	16K	4-tailed-tri	43
4-cycle	7	4-cycle	0
3-star	10K	3-star	13
4-path	8.5K	4-path	33
4-triangle	0.3M	4-triangle	1.2K
4-2star	1.3M	4-2star	1.1K
4-2edge	0.4M	4-2edge	1.1K
4-1edge	60M	4-1edge	23K
4-indep.	0.8B	4-indep.	40K

Name 898 (370-163)

	local	mean	max	std	var
degree	10				
triangles	5	3	20	2	4
wedges	0	8	42	8	66
4-clique	10	4	38	6	32
4-chordal-cycle	0	5	157	9	78
4-tailed-tri	22	42	500	46	2.1K
4-cycle	0	0	3	0	0
3-star	0	42	453	86	7.4K
4-path	0	9	416	31	983
triangle core	5	3	6	2	3
edge roles	7	2	6	2	3

(b) Interactive exploration of the predicted links

Fig. 7. **Interactive link prediction.** In (a) the user can interactively learn link prediction models using the easy-to-use interface on the lower-right (red rectangle). Notably, we learn a model using squared-loss with non-negativity constraints and use it to predict the top-k links with largest weight in a network science co-authorship network (ca-netscience) [49]. The black dashed links (⇢) above are predicted links which represent authors whom are likely to publish together as well as share similar interests. Moreover, the predicted links can be viewed as co-author recommendations. All of the various components of the link prediction and weighting models can be interactively explored in real-time. Notice that one can interactively predict links and then use the new graph representation to improve the accuracy of classification by using the interactive relational learning/classification component of GraphVis. (b) Uses the previous interactive visual graph techniques in GraphVis to explore and understand the predicted links in real-time. For instance, in the above visualization, the user hovers over a link to display its properties and visually selects a subgraph of interest that contains many of the predicted co-authorship links.

Link prediction is the task of predicting the existence of a new (unobserved) link $(v_i, v_j) \notin E$ between two nodes in G, whereas *link weighting* is the problem of estimating the true weight Φ_{ij} of a link $(v_i, v_j) \in E$. The link weighting problem is also called relationship strength estimation. Estimating the strength of relationships is an important problem and may be useful for improving predictive models by reducing the impact of noisy or spurious links, or to identify important or influential individuals in social networks, among other applications.

We solve both problems by finding $\mathbf{W} \in \mathbb{R}^{m \times d}$ and $\mathbf{H} \in \mathbb{R}^{n \times d}$ where $d \ll \min(m, n)$ such that $\mathbf{A} \approx \mathbf{W}\mathbf{H}^T$.[5] More formally, find:

$$\min_{\mathbf{W},\mathbf{H}} \sum_{(i,j) \in E} \left\{ (A_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2 + \lambda_w \|\mathbf{w}_i\|^2 + \lambda_h \|\mathbf{h}_j\|^2 \right\} \tag{2}$$

[5]Note that for the link weighting problem one can hold-out a fraction of the links, learn a model on the other (observed) links, and then use the model to estimate weights for the links in the hold-out set. Other techniques that leverage the above low-rank matrix approximation are also possible.

Intuitively, each row $\mathbf{w}_i^T \in \mathbb{R}^d$ of $\mathbf{W}$ represents a d-dimensional embedding of the i-th row in $\mathbf{A}$ ($\mathbf{a}_i^T \in \mathbb{R}^n$) whereas each row $\mathbf{h}_j^T \in \mathbb{R}^d$ of $\mathbf{H}$ represents a d-dimensional embedding of the j-th column in $\mathbf{A}$ ($\mathbf{a}_j \in \mathbb{R}^m$). The *regularization parameters* $\lambda_w > 0$ and $\lambda_h > 0$ in Eq. (2) are scalars that trade off the loss function with the regularizer. The above low-rank matrix factorization maps each row i or column j of $\mathbf{A}$ to a feature vector $\mathbf{w}_i$ or $\mathbf{h}_j$ in a d-dimensional latent space. Notice the above uses a nonzero squared loss: $(A_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2$, though, any arbitrary separable loss may be used. Furthermore, we can impose hard constraints C on $\mathbf{W}$ and $\mathbf{H}$ such as non-negativity constraints $\mathbf{W}, \mathbf{H} \geq 0$:

$$\min_{\mathbf{W}\geq 0, \mathbf{H}\geq 0} \sum_{(i,j)\in E} \left\{ (A_{ij} - \mathbf{w}_i^T \mathbf{h}_j)^2 + \lambda_w \|\mathbf{w}_i\|^2 + \lambda_h \|\mathbf{h}_j\|^2 \right\} \tag{3}$$

GraphVis uses a recent scalar (block) coordinate descent optimization scheme with fast and efficient element-wise updates along the columns [53]. We can estimate the likelihood of a link between i and j where $(i,j) \notin E$ using $\mathbf{w}_i^T \mathbf{h}_j$ directly. This can be repeated for each pair $(i,j) \notin E$ (or a sample of such pairs) and the top-k links with largest weights can be maintained efficiently (see discussion below for details). Furthermore, we estimate the link strength/weight of existing edges as $\mathbf{w}_i^T \mathbf{h}_j$ where $(i,j) \in E$. In Fig. 7, we predict the top-k links with largest weight and visualize these links immediately as the user interactively explores various models, model dimensionality, objective functions, constraints, regularizers, regularization hyperparameters, among others. To efficiently compute and maintain the set of top-k links, we maintain a priority queue (implemented as a min-heap) allowing us to check in $o(1)$ time whether a given link with an arbitrary weight should be included in the set of top-k links or not. A link is added to the set of top-k links if its weight is larger than the link with the smallest weight existing in the set (assuming k links already exist and thus it is at max capacity). The previous link with smallest weight is discarded first. Furthermore, it takes only $O(\log k)$ time to add a link to the min-heap. Obviously, it takes only $O(k)$ space. Estimating the strength of a single link takes $O(d)$ time. Therefore, it takes $O(nmd)$ time to estimate all pairs in $\mathbf{A}$. Both can be efficiently computed in parallel.

The sliders and other interface elements on the right in Fig. 7(a) allow the user to interactively tune the link prediction and weighting models based on the objective function, constraints, dimensionality, regularization, number of inner and outer iterations in the optimization scheme, the error tolerance before convergence, the number of links to predict, and other factors. After each model is learned, the user can understand and explore the models in real-time by coloring, weighting, or filtering the graph using features derived from the model such as the estimated link weights.

Improving Classification: We may also use interactive link prediction to learn a more effective relational representation for a variety of relational learning tasks. In particular, Fig. S6 in the supplementary material shows the impact on the network structure when the approach is used. Strikingly, the approach creates edges between nodes of the same class, making them significantly closer compared to the original relational data. Another example is shown in Fig. S7 where we find that links are predicted among the misclassified nodes of the same class. The predicted links reinforce the misclassified nodes by connecting them up to nodes with the same class. Hence, learning a model using the new graph representation often leads to better predictive performance.

7 INTERACTIVE GRAPH GENERATION

Graph generators are useful for simulations, testing algorithms, assumptions, benchmarks, among others [12]. The *interactive graph generators* developed in this work are broadly categorized into:

1) Pattern-based graph generators that allow users to easily add subgraph patterns (such as k-cliques, k-stars) to an existing graph by interacting directly with the network visualization.

2) Interactive model-based graph generators (Section 7.2.1), as well as novel block model approaches that capture community structure and other properties better (Section 7.2.2).
3) Hybrid approaches that allow users to add realistic probabilistic patterns (based on a graph model) to existing graphs via direct manipulation with the visualization (Section 7.3).

7.1 Pattern-based Graph Generators

This section first introduces basic concepts used to create and control pattern-based graph generators, then we introduce the main types of pattern-based generators. Given a graph $G = (V, E)$ and a subgraph $H = (W, F)$ representing an arbitrary graph pattern (*e.g.*, clique) selected by the user, how should we connect the selected subgraph pattern H to the original graph G?

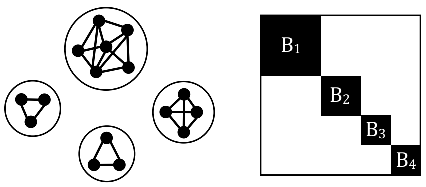
The goal is to make it as easy and intuitive as possible while also being the most generally useful for a wide range of applications and use cases. We satisfy this goal by allowing users to quickly select nodes in G for which the subgraph pattern H will be connected based on a lasso-type selection or user-specified region, *e.g.*, a circle centered at (x, y) — the current position of the cursor; and a radius specified by the user that controls the nodes under consideration *w.r.t.* a particular (x, y) position. Once nodes are selected, the subgraph pattern H is added to G by simply clicking directly on the visualization, *e.g.*, nodes in G that lie within the region (defined by the cursor (x, y) location in the visualization window and its size/radius) are connected to the new nodes from the newly generated pattern H. Intuitively, this makes it easy and fast for users to generate synthetic graphs for simulations, exploration, hypothesis testing, among others. More formally, given the set of selected nodes $U \subseteq V$ (*e.g.*, that lie in the region centered at the mouse cursor), we create a link for each pair of nodes (u, w) such that $u \in U$ (node in the region) and $w \in W$ (node from pattern). Finally, all pattern-based graph generators have an additional parameter that controls the effective size of the subgraph pattern in terms of nodes added to the graph (*e.g.*, clique of size k).

There are two main types of pattern-based graph generators: (i) simple edge and node patterns and (ii) fundamental subgraph patterns such as k-stars, k-cliques, k-cycles, and k-chains that serve as fundamental building blocks of complex real-world networks. For a summary of the main pattern-based generators, see Table S3 provided in the supplementary material.

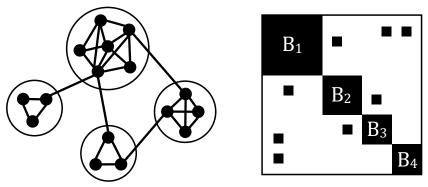
7.2 Interactive Model-based Generators

This section presents two fundamental classes of *interactive model-based generators*: (i) interactive probabilistic generators (Section 7.2.1) and (ii) interactive block model generators (Section 7.2.2).

7.2.1 Interactive Probabilistic Generators. GraphVis also provides users with a wide variety of interactive probablistic generators based on models such as Chung-Lu (CL) [9], preferential attachment (PA) [10], Erdős-Rényi (ER) [25], Kronecker product graph model (KPGM) [43] and random geometric (RG) graphs. A few probabilistic generators are now discussed. Let $\textsc{er}(n, p)$ denote an ER graph [25] that arises from fixing N nodes and generating edges independently with probability p. Thus, the expected degree for each node is simply $p(N - 1)$. Other random graph models were proposed for other types of networks and/or objectives. For instance, CL [9] and PA models [10] match expected scale-free degree distributions. More specifically, the probability of adding the edge (i, j) in CL is: $P(i, j \mid d) = {}^{d_i d_j}/_{\sum_k d_k}$. All graph models are completely interactive and easily explored visually in real-time by adjusting the various model parameters. Interactive graph models enable users to *quickly* understand the fundamental processes governing each of the graph models through interactive visualizations, and are useful for simulations, testing algorithms, among other applications. Furthermore, all interactive visual graph mining techniques as well as interactive visual filtering are useful for interactively exploring, querying, and making sense out of the graph models in real-time.



(a) Step 1: Add local edges within each block



(b) Step 2: Add global edges between the blocks

Fig. 8. Overview of the general block model approach. (a) generates local edges within each block whereas (b) generates global edges between each block. The user can interactively select the graph model $M \in \mathcal{M}$ to use for each step above. In addition, the user can interactively tune the number of nodes, number of blocks, as well as the parameters of the local and global graph models used in Step 1 and 2 above.

7.2.2 Interactive Block Model Generators. For capturing community structure, we propose a number of interactive block model approaches that combine multiple one-stage probabilistic models such as CL, ER, or PA by creating K graphs using these one-stage probabilistic models and then connecting them up by adding edges between the graphs (inter-community/block edges) via another graph model (*e.g.*, an ER model) which we denote as the global graph model. As an example, a user can select a CL model to generate K graphs and then add edges between each of them using an ER model according to some probability p_c. More formally, the process begins with N isolated nodes which are then distributed among the K blocks in some fashion (*e.g.*, using a uniform discrete distribution $\mathbb{F}_b$) which is performed as a preprocessing step. Afterwards, the general interactive block model approach has two main steps (shown in Fig. 8):

Step 1: This step generates local edges within each block $B_i \in \{B_1, B_2, \ldots, B_K\}$ (intra/within-block edges) according to the user-defined local graph model $M \in \mathcal{M}$ where $\mathcal{M}$ is the set of graph models such as ER, CL, or PA. See Fig. 8(a) for an intuitive illustration.

Step 2: This step creates global edges between the K blocks (inter-block edges) according to the user-defined global graph model $M_G \in \mathcal{M}$. See Fig. 8(b) for intuition.

All interactive block models are easily explored in real-time by adjusting a few simple parameters via sliders and other simple visual interface elements such as dropdown boxes to select the local or global graph models to use. After each user interaction GraphVis generates the resulting block model specified by the user and visualizes the graph in real-time (within seconds) which the user can then use to further refine. The user can interactively generate and visualize the resulting graphs in real-time (within seconds) by adjusting a number of important parameters and models used in the generation process. First, the user can select the number of nodes N to use as well as the number of communities (blocks) K to use in the block model generation process. Second, the user can also interactively select the local graph model $M \in \mathcal{M}$ to use for generating edges within each of the K blocks. Third, once the local graph model M is selected the user can adjust the local graph model parameters used to generate the edges within each block $B_i \in \{B_1, B_2, \ldots, B_K\}$ (*e.g.*, if ER is used then one can adjust the probability of an edge).

We have included a number of useful block models which can be selected via a dropdown menu. Examples of three block models including Block ER, Block PA, and Block CL are shown in Fig. S9. For Block ER, Block PA, and Block CL we use ER, PA, and CL for the local graph models, respectively. For the global graph model used to create edges between blocks we use an ER model for Block ER, Block PA, and Block CL. Nodes are distributed into blocks using a uniform discrete distribution $\mathbb{F}_b$. Similarly, one may also generate the initial graphs using multiple one-stage graph models (such as CL and ER) and then combine them as done previously in Section 7.1.

7.3 Hybrid approaches

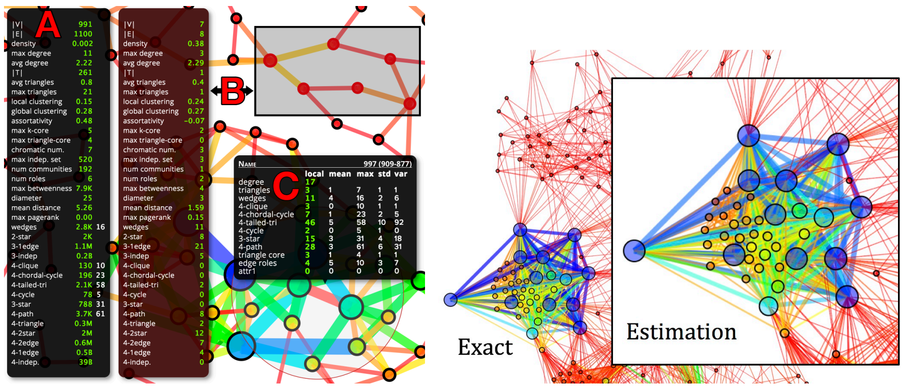
We also propose hybrid approaches that allow users to generate hybrid patterns of a fixed size by interpreting probabilistic model-based generators as probabilistic patterns that can be interactively added by the user in the same fashion as the simple patterns defined in Section 7.1. For instance, the user can easily add a Chung-Lu pattern of a fixed size (*i.e.*, number of nodes) and connect this pattern up to any of the currently visualized nodes using the same simple visual interactive process proposed in Section 7.1. This reinterpretation provides an easy and intuitive way to add realistic probabilistic patterns to existing graphs by interacting directly with the visualizations.

8 REAL-TIME INTERACTIVE PERFORMANCE

Interactive visual graph mining and learning requires efficient, fast, and parallel algorithms to achieve real-time interactive rates. Section 8.1 presents a general estimation framework for provably accurate approximation of subgraph counts and other important graph properties whereas Section 8.2 discusses parallelization of GraphVis.

8.1 Leveraging Statistical Estimation Methods for Real-time Performance

A key challenge when designing interactive graph mining and learning tools is speed and real-time performance. To overcome these challenges, we leverage *fast* and *accurate estimation methods* for many of the computational intensive graph parameters such as induced subgraph counts and statistics (graphlets, network motifs), roles/communities, diameter, betweenness, eccentricity, and other distance-based measures.



|V|	991	
|E|	1100	
density	0.002	
max degree	11	
avg degree	2.22	
|T|	261	
avg triangles	0.8	
max triangles	21	
local clustering	0.15	
global clustering	0.28	
assortativity	0.48	
max k-core	5	
max triangle-core	4	
chromatic num.	7	
max indep. set	520	
num communities	192	
num roles	6	
max betweenness	7.9K	
diameter	25	
mean distance	5.26	
max pagerank	0.00	
wedges	2.8K	16
2-star	2K	
3-1edge	1.1M	
3-indep	0.2B	
4-clique	130	10
4-chordal-cycle	196	23
4-tailed-tri	2.1K	58
4-cycle	78	5
3-star	788	31
4-path	3.7K	61
4-triangle	0.3M	
4-2star	2M	
4-2edge	0.6M	
4-1edge	0.5B	
4-indep.	39B	

|V|	7
|E|	8
density	0.38
max degree	3
avg degree	2.29
|T|	1
avg triangles	0.4
max triangles	1
local clustering	0.24
global clustering	0.27
assortativity	−0.07
max k-core	2
max triangle-core	0
chromatic num.	3
max indep. set	3
num communities	1
num roles	2
max betweenness	4
diameter	3
mean distance	1.59
max pagerank	0.15
wedges	11
2-star	8
3-1edge	21
3-indep	5
4-clique	0
4-chordal-cycle	0
4-tailed-tri	2
4-cycle	0
3-star	0
4-path	8
4-triangle	2
4-2star	12
4-2edge	7
4-1edge	4
4-indep.	0

Name 997 (909-877)

	local	mean	max	std	var
degree	17				
triangles	3	1	7	1	1
wedges	11	4	16	2	6
4-clique	3	0	10	1	1
4-chordal-cycle	7	1	23	2	5
4-tailed-tri	46	5	58	10	92
4-cycle	2	0	5	1	0
3-star	15	3	31	4	18
4-path	28	3	61	6	31
triangle core	3	1	4	1	1
edge roles	4	5	10	3	7
attr1	0	0	0	0	0

Fig. 9. Application of the accurate estimation methods for real-time interactive visual graph mining and learning. **Left:** A subset of the Western US Power Grid network [64] consisting of substations connected by high-voltage transmission lines. Substations and transmission lines are colored/weighted by the estimated local 4-path count. (A) Estimated global graphlet counts and other statistics; (B) summary statistics of the selected subgraph (rectangular region); and (C) local graphlet statistics (including frequency, mean, max, standard deviation, ...) of the selected link. **Right:** *Exact and Estimated Graphlet Features are Nearly Identical.* In this use case we explore a router-level graph with 2,113 routers and 6,632 connections (links) between the routers [52]. Routers and links are colored and weighted by the local 4-cliques. The *exact local 4-clique counts* are shown on the left whereas the right is *estimated counts* (using the estimators in Alg. S1 of the supplementary material with $\epsilon = 0.05$ and $\delta = 0.05$). See text for further discussion.

Table 3. Global estimates of expected value and relative error using 100K samples. The graphlet statistic for the full graph is shown in the first column.

	GRAPH	$\lvert E\rvert$	X	$\hat{X}$	$\frac{\lvert X-\hat{X}\rvert}{X}$
4-CLIQUE	ca-citeseer	814K	18.7M	18.7M	0.0004
	ca-dblp-2012	1M	16.7M	16.7M	0.0004
	soc-gowalla	950K	6M	6M	0.0009
	soc-pokec	22.3M	42.9M	42.9M	0.0002
	socfb-Berkeley	852K	26.6M	26.6M	0.0007
	socfb-Indiana	1.3M	60.1M	60.1M	0.0004
	socfb-MIT	251K	13.6M	13.6M	0.0004
	socfb-Texas84	1.5M	70.7M	70.7M	0.0002
	web-wikipedia09	4.5M	1.4M	1.4M	0.0004

Table 4. Local graphlet estimation experiments. We estimate both the counts and graphlet frequency distribution (GFD) for the top-1000 edges with largest degree ($d_v + d_u$) using a sampling probability of $\pi_r = 0.01$ and report the average relative error.

GRAPH							
soc-gowalla	GFD	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$
	COUNT	0.008	0.008	0.001	0.006	$<10^{-4}$	0.0003
ca-dblp12	GFD	0.0001	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$	$<10^{-4}$
	COUNT	0.003	$<10^{-4}$	0.0003	$<10^{-4}$	0.0004	$<10^{-4}$
socfb-Texas84	GFD	0.001	0.001	0.001	0.001	0.001	0.001
	COUNT	0.031	0.075	0.013	0.042	0.002	0.002

8.1.1 General Estimation Approach: From Global to Local Statistics and Distributions. We describe a general approach for estimating both global and local statistics (and distributions), and mention any modifications that may be required for computing either global or local statistics and local distributions. We also discuss a number of statistics that illustrate a variety and range of statistics that can be estimated using such an approach. For generality, we use the term graph element to refer to either a node or link. Let $\theta(s)$ and $\Theta(G)$ denote the local and global single-valued statistic for graph element s and graph G, respectively. For estimating local distributions (*e.g.*, degree distribution), we use $\Theta(s, t)$ for the value at s with parameter t. We now present the general approach:

(1) Sample K graph elements to obtain the set $\mathcal{S}$ using an arbitrary sampling method[6]
(2) For each graph element $s \in \mathcal{S}$, compute local property $\theta(s)$
(3) Derive $\Theta(G)$ from the set $\{\, \theta(s) \mid s \in \mathcal{S} \,\}$
(4) Scale $\Theta(G)$ using the inverse sampling probability $\sigma = 1/p$ (if appropriate)

where p denotes the sampling probability and thus $p = K/|G|$ if uniform random sampling is used and $|G|$ is the total number of graph elements in G (*i.e.*, $|G| = |V|$ total nodes; or $|G| = |E|$ links). Notice that step (4) is optional and depends on the statistic being estimated. We often use $\sigma = 1/p$ to scale the statistic to the full graph (or full neighborhood in the case of local node/link/subgraph statistics), especially when estimating global or local count-based statistics (*e.g.*, number of 4-cliques in G or at a given graph element s). This corresponds to the Horvitz-Thompson construction [3, 37]. However, σ in step (4) can be modified depending on the type of statistic estimated. Furthermore, step (1) uses simple random sampling for count-based statistics; however, a weighted sampling strategy is often useful for other types of statistics such as estimating *extremal statistics*, *e.g.*, max k-truss number in G. Nevertheless, the approach given above (steps 1-4) clearly results in a better approximation as the number of samples K increases.

Now we briefly discuss estimating local statistics such as the number of k-cycles or k-cliques at a given graph element s (node, link). Notice that the runtime of such local statistics depend largely on the size of the local neighborhood surrounding a graph element s, and thus it is most useful to estimate only the top-k elements with the largest such neighborhoods (as the vast majority of other graph elements in sparse graphs have relatively small neighborhoods and can be computed fast using exact methods). Many local statistics begin by computing the set of neighbors to explore. In this case, we simply use a sampling method to select a subset of neighbors to explore. Recall that the sampling will largely depend on the type of statistic being estimated. For estimating extremal

[6]For example, using uniform random sampling or weighted sampling or other non-uniform sampling technique [5].

statistics like max, we select the neighbors that will most likely lead to such extreme values, whereas for count-based statistics we use uniform random sampling with a local scaling (step 4).

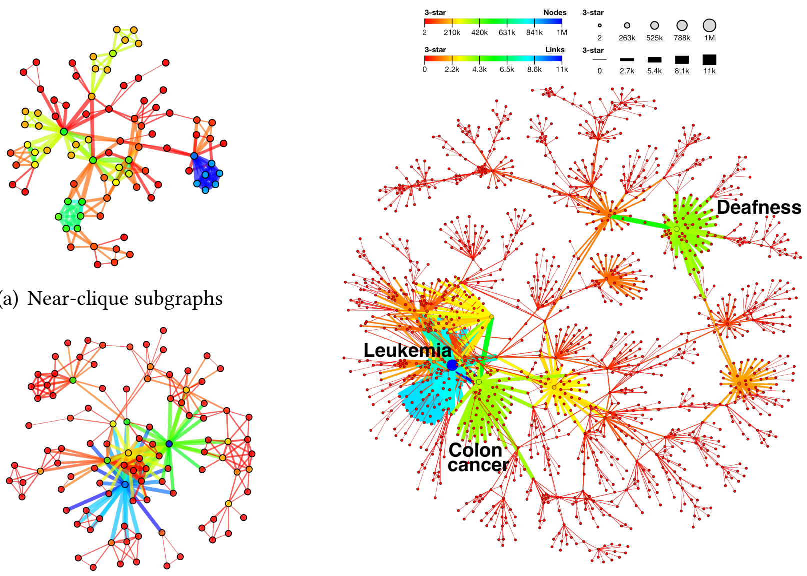
Estimation techniques are exploited for computing and maintaining many of the computationally expensive global statistics (Fig. 9A). Many local statistics are estimated on-the-fly, *e.g.*, when a user hovers over a node or link, we immediately estimate the local statistics for the selected graph element. Afterwards these statistics are stored for later use and simply retrieved as needed and therefore we avoid having to recompute them. Notice that many such local statistics are computed from the immediate neighborhood of a graph element (*e.g.*, many graphlet statistics such as triangles), and thus, we determine and update the graph elements and the corresponding local statistics that are impacted by adding, removing, or changing one or more graph elements (nodes, links). Similar estimation techniques are used when the user selects a subgraph of interest (Fig. 9B).

8.1.2 Estimation of Subgraph Properties. Techniques for estimating both global (macro) and local (micro) subgraph counts are incorporated into GraphVis. For instance, estimating the counts of all 3-node subgraphs (such as triangles), 4-node subgraphs (such as 4-cliques, 4-cycles), among others. Such techniques are crucial for interactive real-time performance.

Estimation Experiments: We proceed by first demonstrating the effectiveness of the methods for estimating the local frequency of both connected and disconnected graphlets up to size $k = 4$. Given an estimated count $\hat{X}_{ij}$ of an arbitrary graphlet H_i for edge $e_j \in E$, we consider the relative error: $\mathbb{E}(\hat{X}_{ij} \parallel X_{ij}) = \left|X_{ij} - \hat{X}_{ij}\right| / X_{ij}$ where X_{ij} is the actual count of H_i for e_j. The relative error indicates the quality of an estimated graphlet statistic relative to the magnitude of the actual statistic. We first demonstrate the effectiveness of estimating local graphlet statistics. Table 4 shows the relative error for the top 1000 links with highest cumulative degree $(d_v + d_u)$ for each link $e_j = (v, u)$. Notably, these links are also the most computationally challenging and therefore most useful to estimate. We also demonstrate the effectiveness of estimating global statistics. Experimental results for estimating the global 4-clique count on a variety of graphs is shown in Table 3.

Impact on Revealing Higher-order Structures: In Fig. 9 (Right), notice the visualization that uses the estimated graphlet features (to encode the color and size of the links and nodes) is strikingly similar to the one using exact graphlet features; and the differences are trivial and insignificant. Hence, the graphlet estimators are extremely accurate and effective for revealing higher-order structure and organization in large networks. Strikingly, we also observe in Fig. 10 that GraphVis can be used to detect and spot large cliques and stars visually in real-time using small induced subgraph patterns (graphlets, motifs). Intuitively, Fig. 10 uses GraphVis to immediately reveal higher-order structures by encoding the color and size of nodes and links in the network using the counts of a few network motifs/graphlets. In this way, we can leverage graphlets to find and rank large stars, cliques, and other larger higher-order structures (Fig. 10) that are of fundamental importance in many types of networks [8].

Impact on Relational Classification: We also investigate the difference in predictive performance when using graphlet features that are estimated (as opposed to exact graphlet features that are significantly more expensive to compute). Using the terrorist & known association network [70] (see Fig. 6 for data description) with 5-fold cross-validation and a label density of 10%, we observe 92.29% accuracy with the *exact* graphlet features and 92.16% with the *estimated* graphlet features. Notably, the difference in accuracy between the exact and estimated local graphlet features is *insignificant.* As an aside, we used the relational similarity machines (rsm) classifier for prediction [55]. Similar results were found using the cora and citeseer citation networks.



(a) Near-clique subgraphs

(b) Large star-like subgraphs

(c) Revealing important phenotypes

Fig. 10. **Higher-order Network Analysis: Spotting, Detecting, and Ranking Higher-order Structures.** Graphlets/network motifs are fundamental building blocks of complex networks. (a)-(b) For this use case we leverage the network science co-authorship graph [49]. Strikingly, we observe that higher-order structures such as large cliques (which finding the largest clique is NP-hard) and stars can be detected and spotted visually in real-time using small induced subgraph patterns (graphlets, motifs). Nodes and links are colored by 4-cliques and 3-paths in (a) and (b), respectively. We observe that graphlet frequencies can be used as a proxy to understand and detect large subgraph patterns as shown above for clique and star-like subgraph patterns. As shown above, we can easily spot visually and quantitatively rank the larger subgraph patterns by size. (a) Large dense/near-clique subgraphs are revealed above and naturally form a ranking; ordered from largest (blue) to smallest (red) as can be easily spotted in the above visualization. (b) Similarly, GraphVis can be used to quickly spot large star-like subgraphs visually. (c) In addition, using the network of known disorders and genes linked to them [31] we immediately uncover the largest stars in the graph which correspond to important phenotypes such as leukemia, color cancer, and deafness (ordered by size). These phenotypes correspond to hubs (large stars) that connect to a number of distinct disorders which is consistent with [31].

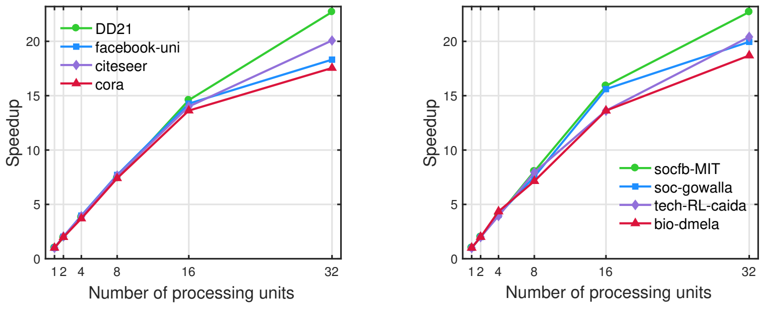
8.2 Parallelization

To support the real-time interactive requirements of GraphVis, many components are parallelized:

- Interactive relational learning and classification algorithms are parallelized. For instance, the model is recomputed in parallel upon any user interaction that requires significant work such as the addition/deletion of a subgraph; or whenever a user interactively refines the model hyperparameters, selects a different kernel function to use, or includes new attributes, etc.
- Global and local graphlet (network motif) statistics & properties are parallelized and updated whenever a new graph is loaded, or any interactive (visual) query is made by the user (*e.g.*, selection of subgraphs to explore).
- Interactive graph partitioning algorithms including the discovery of roles and communities are computed in parallel whenever appropriate. These are often updated in parallel after any significant insertion, deletion, or changing of the graph structure, or whenever a user changes the specific community/role discovery method or hyperparameters.

We also use hardware acceleration and leverage GPUs for rendering visualizations.

8.2.1 Parallel Performance Experiments. Given p parallel workers (cores, processors), speedup is defined as $S_p = T_{\text{seq}}/T_p$ where T_{seq} is the runtime of the sequential algorithm and T_p is the runtime of the parallel algorithm on p workers. Results are shown in Fig. 11 for two components of GraphVis. In particular, we observe strong scaling results in both cases.



(a) Interactive Relational Learning (b) Interactive Local Graphlets

Fig. 11. Parallel performance. Strong parallel scaling results are observed across a variety of networks.

9 CONCLUSION

This paper proposed an interactive visual graph mining and machine learning platform called GraphVis for deriving important insights in a timely fashion from large, complex, dynamic, and uncertain graph data. It is designed to be fast, flexible, and completely interactive. Most importantly, GraphVis enables users to quickly explore, understand, and obtain key insights into graph data (for decision-making, planning) with minimum effort by taking advantage of state-of-the-art *graph mining and relational learning techniques* paired with a variety of useful *visual representations of the graph data* along with easy-to-use and intuitive *interaction techniques*. Furthermore, we also proposed techniques for interactive relational learning (*e.g.*, node/link classification), interactive link prediction and weighting, interactive role discovery and community detection, higher-order network analysis (via graphlets, network motifs), among others. Human analysts can refine and tune the various graph mining and relational learning methods in a timely and easy-to-use fashion for specific tasks and constraints via an end-to-end interactive visual analytic pipeline that learns, infers, and provides rapid interactive visualization with immediate feedback at each change/prediction in real-time. Other key aspects of GraphVis include interactive visual filtering, querying, ranking, manipulating, exporting, as well as tools for interactive dynamic network analysis and visualization, interactive graph generators/models (including new block model approaches), and a variety of multi-level network analysis techniques.

REFERENCES

[1] Charu Aggarwal and Karthik Subbian. 2014. Evolutionary network analysis: A survey. *CSUR* 47, 1 (2014), 10.

[2] Christopher Ahlberg, Christopher Williamson, and Ben Shneiderman. 1992. Dynamic queries for information exploration: An implementation and evaluation. In *Proc. of SIGCHI*. 619–626.

[3] Nesreen K Ahmed, Nick Duffield, Jennifer Neville, and Ramana Kompella. 2014. Graph Sample and Hold: A Framework for Big-Graph Analytics. In *SIGKDD*. 1–10.

[4] Nesreen K. Ahmed, Nick Duffield, Theodore L. Willke, and Ryan A. Rossi. 2017. On Sampling from Massive Graph Streams. In *VLDB*. 1430–1441.

[5] Nesreen K. Ahmed, Jennifer Neville, and Ramana Kompella. 2014. Network Sampling: From Static to Streaming Graphs. *TKDD* 8, 2, Article 7 (June 2014), 56 pages.

[6] Nesreen K. Ahmed and Ryan A. Rossi. 2015. Interactive Visual Graph Analytics on the Web. In *ICWSM*. 566–569.

[7] Nesreen K. Ahmed, Ryan A. Rossi, Theodore L. Willke, and Rong Zhou. 2017. Edge Role Discovery via Higher-order Structures. In *PAKDD*. 291–303.

[8] Nesreen K. Ahmed, Theodore L. Willke, and Ryan A. Rossi. 2016. Estimation of Local Subgraph Counts. In *BigData*.

[9] William Aiello, Fan Chung, and Linyuan Lu. 2001. A random graph model for power law graphs. *Exp. Math.* 10, 1 (2001), 53–66.

[10] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys* 74 (2002), 47.

[11] David Auber. 2004. Tulip – A huge graph visualization framework. In *Graph Drawing Soft.* 105–126.

[12] A.L. Barabási and RE Crandall. 2003. Linked: The new science of networks. *A. J. of P.* 71, 4 (2003), 409–410.

[13] Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. 2009. Gephi: an open source software for exploring and manipulating networks. *ICWSM* 8 (2009), 361–362.

[14] Vladimir Batagelj and Andrej Mrvar. 2004. Pajek – analysis and visualization of large networks. In *Graph drawing software*. Springer, 77–103.

[15] Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. 2014. The state of the art in visualizing dynamic graphs. *EuroVis STAR* 2 (2014).

[16] Skye Bender-deMoll and Daniel A McFarland. 2006. The art and science of dynamic network visualization. *JoSS* 7, 2 (2006), 1–38.

[17] Bobby-Joe Breitkreutz, Chris Stark, and Mike Tyers. 2003. Osprey: a network visualization system. *Gen. bio.* 4, 3 (2003).

[18] Nan Cao, Yu-Ru Lin, Liangyue Li, and Hanghang Tong. 2015. g-Miner: Interactive visual group mining on multivariate graphs. In *HCI*. ACM, 279–288.

[19] Kathleen M Carley. 2003. *Dynamic network analysis.* Citeseer.

[20] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. 2012. Time-varying graphs and dynamic networks. *Int. J. of Par., Emergent and Dist. Sys.* 27, 5 (2012), 387–408.

[21] Gabor Csardi and Tamas Nepusz. 2006. The igraph software package for complex network research. *InterJournal* Complex Systems (2006), 1695. http://igraph.org

[22] Weiwei Cui, Hong Zhou, Huamin Qu, Pak Chung Wong, and Xiaoming Li. 2008. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008), 1277–1284.

[23] David Ebert, Kelly Gaither, Yun Jang, and Sonia Lasher-Trapp. 2014. Cross-Scale, Multi-Scale, and Multi-Source Data Visualization and Analysis Issues and Opportunities. (2014), 353–360.

[24] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C North, and Gordon Woodhull. 2001. Graphviz — open source graph drawing tools. In *International Symposium on Graph Drawing*. Springer, 483–484.

[25] Paul Erdős and A Rényi. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci* 5 (1960), 17–61.

[26] S. Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3-5 (2010), 75–174.

[27] Thomas MJ Fruchterman and Edward M Reingold. 1991. Graph drawing by force-directed placement. *Software: Practice and experience* 21, 11 (1991), 1129–1164.

[28] Emden R Gansner, Yifan Hu, and Stephen North. 2013. A maxent-stress model for graph layout. *IEEE Transactions on Visualization and Computer Graphics* 19, 6 (2013), 927–940.

[29] Emden R Gansner, Yifan Hu, Stephen North, and Carlos Scheidegger. 2011. Multilevel agglomerative edge bundling for visualizing large graphs. In *Pacific Visualization Symposium (PacificVis)*. 187–194.

[30] L. Getoor and B. Taskar (Eds.). 2007. *Introduction to Statistical Relational Learning.* MIT Press.

[31] Kwang-Il Goh, Michael E Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-László Barabási. 2007. The human disease network. *PNAS* 104, 21 (2007), 8685–8690.

[32] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*. Pasadena, CA USA, 11–15.

[33] Ivan Herman, Guy Melançon, and M Scott Marshall. 2000. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics* 6, 1 (2000), 24–43.

[34] Petter Holme and Jari Saramäki. 2013. *Temporal networks.* Springer.

[35] Danny Holten. 2006. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 741–748.

[36] Danny Holten and Jarke J Van Wijk. 2009. Force-Directed Edge Bundling for Graph Visualization. In *Computer graphics forum*, Vol. 28. Wiley Online Library, 983–990.

[37] Daniel G Horvitz and Donovan J Thompson. 1952. A generalization of sampling without replacement from a finite universe. *JASA* 47, 260 (1952), 663–685.

[38] Yifan Hu and Lei Shi. 2015. Visualizing large graphs. *Wiley Interdisciplinary Reviews: Comp. Stat.* 7, 2 (2015), 115–136.

[39] Yuntao Jia, Jared Hoberock, Michael Garland, and John Hart. 2008. On the visualization of social and other scale-free networks. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008), 1285–1292.

[40] Hyojung Kang and David Munoz. 2015. A dynamic network analysis approach for evaluating knowledge dissemination in a multi-disciplinary collaboration network in obesity research. In *Proc. of the Winter Simulation Conf.* 1319–1330.

[41] Daniel Keim, Gennady Andrienko, Jean-Daniel Fekete, Carsten Gorg, Jorn Kohlhammer, and Guy Melançon. 2008. Visual analytics: Definition, process, and challenges. *Lecture notes in computer science* 4950 (2008), 154–176.

[42] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. 2006. Task taxonomy for graph visualization. In *AVI Workshop on BEyond Time and Errors: Novel Eval. Methods for Info. Visualization.* 1–5.

[43] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. 2010. Kronecker graphs: An approach to modeling networks. *JMLR* 11, Feb (2010), 985–1042.

[44] Jure Leskovec and Rok Sosič. 2016. SNAP: A General-Purpose Network Analysis and Graph-Mining Library. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 1 (2016), 1.

[45] F. Lorrain and H.C. White. 1971. Structural equivalence of individuals in social networks. *JAMS* 1, 1 (1971), 49–80.

[46] Sofus A Macskassy and Foster Provost. 2007. Classification in networked data: A toolkit and a univariate case study. *JMLR* 8 (2007), 935–983.

[47] James Moody, Daniel McFarland, and Skye Bender-deMoll. 2005. Dynamic network visualization. *Amer. J. Sociology* 110, 4 (2005), 1206–1241.

[48] Mark EJ Newman. 2001. The structure of scientific collaboration networks. *PNAS* 98, 2 (2001), 404–409.

[49] Mark EJ Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical review E* 74, 3 (2006), 036104.

[50] Ryan A. Rossi. 2013. Fast Triangle Core Decomposition for Mining Large Graphs. In *PAKDD.* 1–12.

[51] Ryan A. Rossi and Nesreen K. Ahmed. 2015. Role Discovery in Networks. *TKDE* 27, 4 (2015), 1112–1131.

[52] Ryan A. Rossi, Sonia Fahmy, and Nilothpal Talukder. 2013. A Multi-Level Approach for Evaluating Internet Topology Generators. In *Networking.* 1–9.

[53] Ryan A. Rossi and Rong Zhou. 2016. Parallel Collective Factorization for Modeling Large Heterogeneous Networks. In *Social Network Analysis and Mining (SNAM).* 30.

[54] Ryan A. Rossi and Rong Zhou. 2016. Toward Interactive Relational Learning. In *AAAI.*

[55] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2016. Relational Similarity Machines. In *MLG.* 1–8.

[56] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2018. Deep Inductive Network Representation Learning. In *3rd International Workshop on Learning Representations for Big Networks (WWW BigNet).* 8.

[57] Lei Shi, Chen Wang, and Zhen Wen. 2011. Dynamic network visualization in 1.5 D. In *PacificVis.* IEEE, 179–186.

[58] Ben Shneiderman. 1994. Dynamic queries for visual information seeking. *IEEE software* 11, 6 (1994), 70–77.

[59] Ben Shneiderman. 1997. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *Proceedings of the 2nd international conference on Intelligent user interfaces.* ACM, 33–39.

[60] Michael E Smoot, Keiichiro Ono, Johannes Ruscheinski, Peng-Liang Wang, and Trey Ideker. 2010. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics* 27, 3 (2010), 431–432.

[61] Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. 2017. TRIÈST: Counting Local and Global Triangles in Fully Dynamic Streams with Fixed Memory Size. *TKDD* 11, 4 (2017), 43.

[62] James J. Thomas and Kristin A. Cook. 2005. *Illuminating the Path: the research and development agenda for visual analytics.* IEEE Computer Society.

[63] Frank Van Ham and Martin Wattenberg. 2008. Centrality based visualization of small world graphs. In *Computer Graphics Forum*, Vol. 27. 975–982.

[64] Jian-Wei Wang and Li-Li Rong. 2009. Cascade-based attack vulnerability on the US power grid. *Safety Science* 47, 10 (2009), 1332–1336.

[65] Michelle Q Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. 2000. Guidelines for using multiple views in information visualization. In *AVI.* 110–119.

[66] Colin Ware and Robert Bobrow. 2005. Supporting visual queries on medium-sized node–link diagrams. *Information Visualization* 4, 1 (2005), 49–58.

[67] Mingrui Xia, Jinhui Wang, and Yong He. 2013. BrainNet Viewer: a network visualization tool for human brain connectomics. *PloS one* 8, 7 (2013), e68910.

[68] Kevin S Xu, Mark Kliger, and Alfred O Hero. 2013. A regularized graph layout framework for dynamic network visualization. *DMKD* (2013), 1–33.

[69] Ji Soo Yi, Youn ah Kang, and John Stasko. 2007. Toward a deeper understanding of the role of interaction in information visualization. *TVCG* 13, 6 (2007), 1224–1231.

[70] Bin Zhao, Prithviraj Sen, and Lise Getoor. 2006. Event Classification and Relationship Labeling in Affiliation Networks. In *ICML Workshop on Statistical Network Analysis (SNA).*

Received May 2017; revised November 2017; accepted XXXX 2018