VIGOR: Interactive Visual Exploration of Graph Query Results

Robert Pienta, Fred Hohman, Alex Endert, Acar Tamersoy,
Kevin Roundy, Chris Gates, Shamkant Navathe, Duen Horng Chau

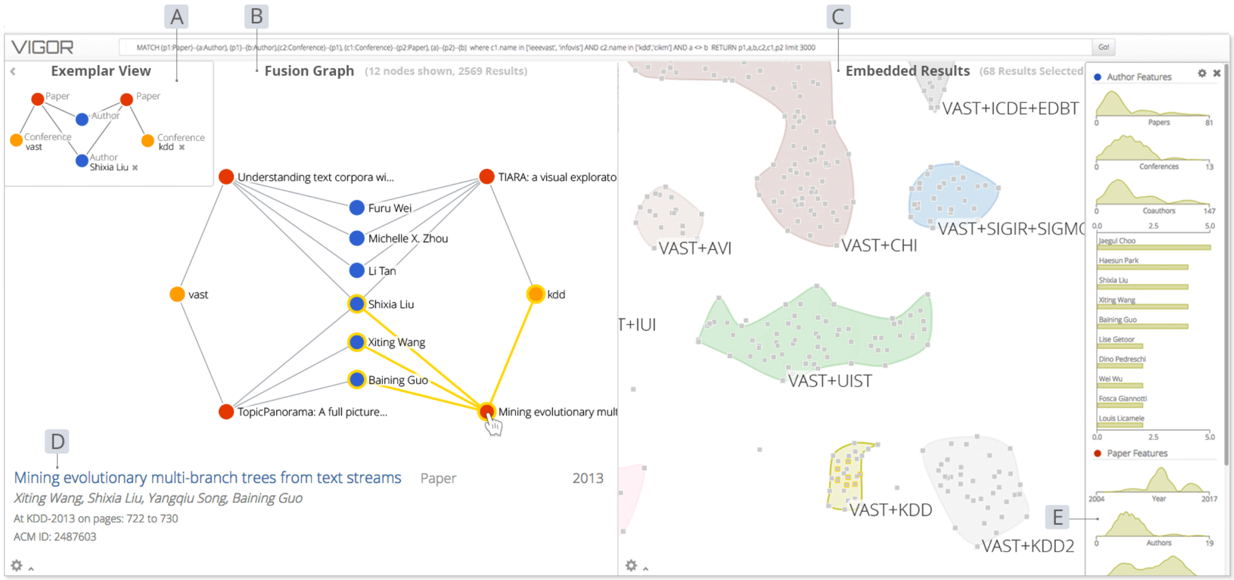

Fig. 1. A screenshot of VIGOR showing an analyst exploring a DBLP co-authorship network, looking for researchers who have co-authored papers at the VAST and KDD conferences. (A) The Exemplar View visualizes the query, and (B) the Fusion Graph shows the induced graph formed by joining all query matches. Picking constant node values (e.g., Shixia) in the Exemplar View filters the Fusion Graph. (C) Hovering over a node shows its details. (D) The Subgraph Embedding embeds each match as a point in lower-dimensional space and clusters them to allow analysts to see patterns and outliers. (E) The Feature Explorer summarizes each cluster's feature distributions.

Abstract—
Finding patterns in graphs has become a vital challenge in many domains from biological systems, network security, to finance (e.g., finding money laundering rings of bankers and business owners). While there is significant interest in graph databases and querying techniques, less research has focused on helping analysts make sense of underlying patterns within a group of subgraph results. Visualizing graph query results is challenging, requiring effective summarization of a large number of subgraphs, each having potentially shared node-values, rich node features, and flexible structure across queries. We present VIGOR, a novel interactive visual analytics system, for exploring and making sense of query results. VIGOR uses multiple coordinated views, leveraging different data representations and organizations to streamline analysts sensemaking process. VIGOR contributes: (1) an exemplar-based interaction technique, where an analyst starts with a specific result and relaxes constraints to find other similar results or starts with only the structure (i.e., without node value constraints), and adds constraints to narrow in on specific results; and (2) a novel feature-aware subgraph result summarization. Through a collaboration with Symantec, we demonstrate how VIGOR helps tackle real-world problems through the discovery of security blindspots in a cybersecurity dataset with over 11,000 incidents. We also evaluate VIGOR with a within-subjects study, demonstrating VIGOR's ease of use over a leading graph database management system, and its ability to help analysts understand their results at higher speed and make fewer errors.

Index Terms—graph querying, subgraph results, query result visualization

1 Introduction

Mining graph patterns, whether suspicious, anomalous, malicious, or just interesting, has become a critical technology for data analytics. For example, in financial transaction networks, analysts may want to flag "near cliques" formed among company insiders who carefully timed their activities [42]. Or in online auctions, analysts may want to uncover "near-bipartite cores" formed among fraudsters and their accomplices [29]. While there is significant research interest and development in graph algorithms, database management systems and even visual graph query construction techniques [2, 7, 32], much less work has focused on helping analysts make sense of the graph structure and rich data that makes up *subgraph results*. Visualizing graph query results (or matches) poses significant challenges, because we must effectively summarize: the underlying data from the nodes, the structure of each subgraph result, a large number of results, and the potential overlap in node and edges among results.

In this work, we visualize the resulting subgraphs from exact graph querying, in which the structure of nodes and edges matches exactly what the analyst specified in their query. Exact graph querying is used in many domains, from bioinformatics [39], cybersecurity [29], social

- *Robert Pienta, Fred Hohman, Alex Endert, Shamkant Navathe, and Duen Horng Chau are with Georgia Tech. E-mail: {pientars, fredhohman, endert, sham, polo}@gatech.edu.*
- *Acar Tamersoy, Kevin Roundy, and Chris Gates are with Symantec Research Labs. E-mail: {acar_tamersoy, kevin_roundy, chris_gates}@symantec.com.*

network analysis [21], to finance [42].

Most graph mining tasks are considered finished when query results have been returned; however, for analysts, seeing initial query results is only the beginning of their sensemaking process. Despite the significant interest in graph database management systems (DBMSs) and querying techniques, little investigation has been done in the space of graph query result visualization and exploration. Contemporary graph querying systems provide only basic methods for displaying results, often using tables or long lists (see examples in Figure 2). Given only the table and list visualizations, it's a challenge to determine what groupings of similar results occur or how a particular node value appears among the results. In the current paradigm, analysts must first find patterns manually in a table before they can rewrite their original queries to do any filtering or grouping. This can be tedious and does not promote the development of an internal representation of the information space [36].

We present a novel visual analytics system, VIGOR, for exploring and making sense of graph querying results. VIGOR uses multiple coordinated views, leveraging different data representations and organizations to streamline analysts' sensemaking process [18, 34]. The important contributions of VIGOR include:

- **Exemplar-based interactive exploration.** VIGOR simultaneously supports *bottom-up* sensemaking [36], where an analyst starts with a specific result and relaxes constraints to find other similar results; and *top-down* sensemaking , where the analyst start with only the structure (i.e., without node value constraints), and add constraints to narrow in on specific results (Figure 1A). VIGOR *supports analysts when investigating how many values are matched to each query-node and how a particular node value filters the results.*

- **Novel result summarization through feature-aware subgraph result embedding and clustering.** VIGOR provides analysts with a *top-down*, high-level overview of all their results which enables analysts to handle complex grouping and comparison tasks to make sense of their data [28, 36]. We introduce an algorithm to group results by node-feature and structural result similarity (Figure 1C) and embed them in a low dimensional representation. By grouping similar results into clusters and making cluster comparison easy, analysts can quickly detect and understand underlying patterns across their results.

- **An integrated system fusing multiple coordinated views.** VIGOR provides multiple brushable linked views to flexibly explore and make sense of subgraph results, by integrating the *Exemplar View*, *Subgraph Embedding View*, and the *Fusion Graph*. The *Fusion Graph* (Figure 1B) shows the subgraph from the underlying network created from combining all the results, in which very common or uncommon nodes will have high and low degree respectively. *The coordinated views make it easier to see how nodes appear together across the many subgraph results.*

- **Real world application to discover cybersecurity blindspots; advancing the state of the art** Through a collaboration with cybersecurity researchers at Symantec, a leading security company, we present the investigative analysis performed in and insights gleaned from using VIGOR to discovering and understanding blindspots in a cybersecurity dataset with over 11,000 real incidents. Through a usability evaluation using real co-authorship network data obtained from DBLP [1], we demonstrate VIGOR's ease of use over Neo4J, a leading graph DBMS, and its ability to help users understand their results at higher speed and with fewer errors.

2 Introducing VIGOR

To illustrate how VIGOR works in practice, we will briefly cover an overview of the system's components (in Section 2.1) and an illustrative scenario where we explore co-authorship in a DBLP network.

[1]DBLP Website: `http://dblp.uni-trier.de/`

Fig. 2. (A) Neo4j, a commercial system, displays subgraph matches in a long table. One match with three nodes is shown here, each gray box describes one nodes' features. (B) VISAGE [32] displays subgraph matches in a list, without revealing connections among results. Even for modest sized queries, these conventional approaches require significant scrolling, and cannot easily reveal broader patterns and relationships among matches.

DBLP Dataset. In this paper we utilize a real co-authorship network drawn from a subset of DBLP's computer science bibliography data. The undirected, unweighted network contains 59,655 authors, 48,677 papers, 7,236 sessions, 417 proceedings, 21 conferences and 1,634,742 relations from the data mining and information visualization communities. We will use this network in both the illustrative scenario (Section 2.2) and in our user study (Section 5.1).

2.1 VIGOR Interface Overview

The VIGOR user interface is composed of four main areas (Figure 1). The *Exemplar View* at the top (Figure 1A) visualizes the user's textual graph query (entered into the text form at the top of Figure 1) and supports quick filtering by value. The *Fusion Graph* (Figure 1B) displays an induced graph of all the result subgraphs from the query, quickly demonstrating which nodes appear often and with which other nodes. The *Subgraph Embedding* view (Figure 1C) summarizes all the results by reducing each result into a square, gray glyph and clustering them (colored, concave clusters) based on feature similarity. Analysts are free to create, name, and compare their own clusters. Clusters are compared in the *Feature Explorer* view (Figure 1E), which provides summary distributions of each node type included in the results. The goal of the VIGOR interface is to enable analysts to detect underlying patterns in their result set as well as explore individual values with as little tedium as possible. The synergy of these techniques across our three views enables analysts to explore their query results with ease.

2.2 Illustrative Usage Scenario

We demonstrate how VIGOR works with an illustrative example exploring a cross-conference co-authorship query. Imagine an analyst, Alexis, is interested in finding authors and papers that bridge the information visualization and data mining communities. This scenario demonstrates some of the interactions and major features in VIGOR.

Because Alexis wants to learn about papers, conferences, and authors, she begins with a query looking for an author who has published two papers with a co-author, where the papers were published

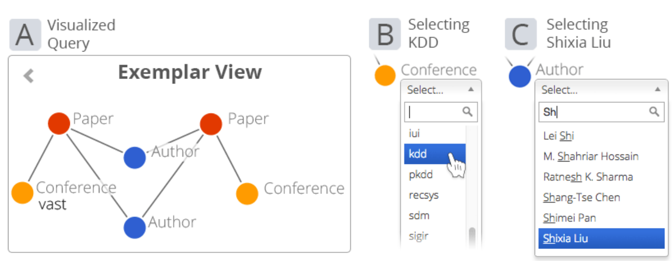

Fig. 3. Exemplar View displaying a query seeking researchers who have coauthored papers at two different conferences. (A) The analyst starts with only the *structure* of the graph query, then incrementally adds node value constraints to narrow in on specific results, (B) first by choosing KDD, which (C) narrows down the remaining choices for authors.

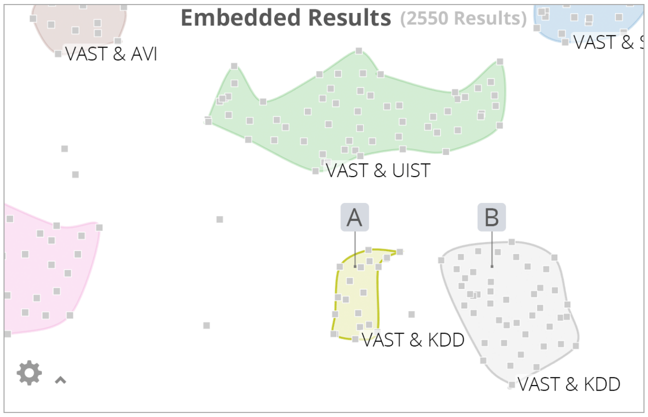

Fig. 4. The Subgraph Embedding provides an overview of the results through the feature-aware subgraph embedding, where results are displayed as points in two dimensions based on node feature similarity. We see the clustered results of a query seeking two co-authors of two papers at VAST and another conference (shown in Figure 3). Nearby clusters (A) and (B) both contain VAST and KDD papers, the features of which are compared in Figure 6. Cluster labels are customized by the analyst during exploration.

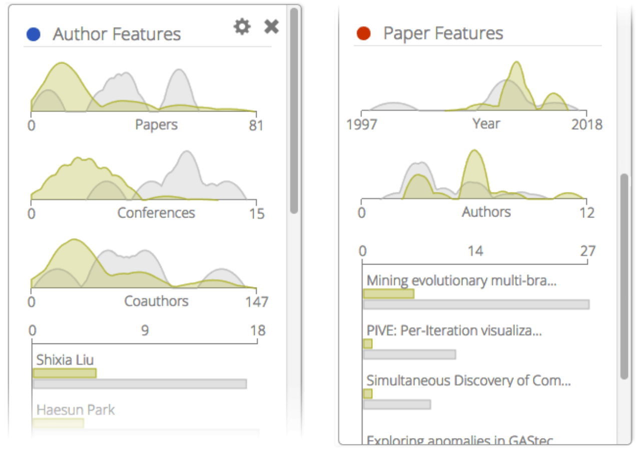

Fig. 6. The Feature Explorer shows common node values and feature distributions for each node type included in two clusters (A and B in Figure 4). The features for each node type in the Fusion Graph view are summarized as distribution charts. The bar chats show the top-k most common values, including those shared between the selected clusters.

to VAST and another conference. In VIGOR, Alexis starts by entering a query written in the Cypher query language from the popular Neo4j (`http://neo4j.com`) DBMS. Her query appears graphically in our Exemplar View, where she verifies that she correctly specified the right structure (Figure 3A).

Alexis has just begun her investigation and she wants to see an overview of her results. She gets over 2,500 results, each with six nodes (from the previously mentioned query), wherein some nodes could be shared among multiple results. She wants a high level overview of her results that allows her to see similarities and groupings.

VIGOR's Subgraph Embedding view provides an overview of all her results in the form of a plot with clusters. Similar subgraph results (gray squares in Figure 4) are placed spatially close together by the feature-aware subgraph result embedding and clustering (Section 4). To help her differentiate among groups, VIGOR uses a density-based clustering technique [22] to detect clusters and automatically creates colored concave hulls for each. Alexis has the option to adjust the embedding and clustering parameters. She may also create and name her own clusters by lassoing groups of points (Figure 5A-C).

She shift-right-clicks two neighboring clusters (Figure 4A and 4B) to compare them in the Feature Explorer (Figure 6). The Feature Explorer shows common node values and feature distributions for each node type included in the clusters, similar to [35, 41]. The color of the plots in the Feature Explorer correspond to the colors of the selected clusters. She can use the value-plots (bar charts in Figure 6) to see what nodes appear most commonly in a cluster. Alexis labels the clusters based on their most common conferences (e.g., "*VAST & UIST*" in Figure 4). She notices that both clusters are composed of authors and their publications at VAST and KDD, a top tier data mining conference. From the author feature distributions in the Feature Explorer (Figure 6-left) she discovers that the gray cluster (cluster B) is likely to contain more senior researchers, because they have higher paper counts, more distinct conferences and greater numbers of co-authors. Her curiosity grows as she wonders what types of papers bridge these two research communities.

After her initial query, Alexis is faced with numerous results, but she wants to find specific authors and papers. What should she do next? She can quickly filter down results by values with which she's comfortable by clicking one of the yellow conference nodes in the Exemplar View window, which displays a searchable dropdown menu with the matching conferences (Figure 3B). She selects KDD. When she clicks on an author node in Figure 3A, the dropdown now contains only those authors who have published together at VAST and KDD. From the list she recognizes Shixia Liu, a VAST'17 Paper Chair, and selects her (Figure 3C).

Alexis' selection in the Exemplar View filters the Fusion Graph (Figure 7), a force-directed graph induced by joining all subgraph matches together (e.g., if a conference is shared among several results, it will appear only once). The Fusion Graph now shows only Shixia Liu's co-authors on at least one paper with her from VAST and one from KDD (e.g., Michelle, Furu, Li, Xiting, and Baining in Figure 7). Alexis discovers that each paper is related to understanding textual data and is potentially valuable to her future research. She is inspired by the combination of the speed, scale, and automation of data mining being combined with the visual, interaction design, and sensemaking of visualization.

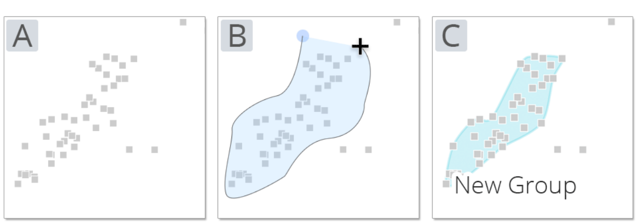

Fig. 5. (A) Starting from a group of results, (B) an analyst lassos the desired results. (C) A concave hull is established forming a cluster with the points. Cluster can be used to: filter the Fusion Graph and compare features and node values in the Feature Explorer.

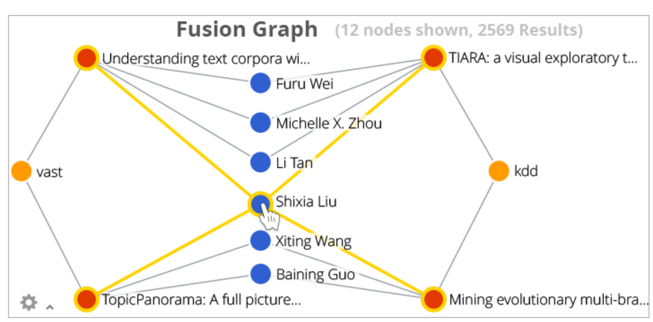

Fig. 7. Shixia Liu's papers and co-authors who have published papers together at VAST and KDD. The Fusion Graph view shows an induced subgraph of all the combined results from the original query, which can be filtered from either the Subgraph Embedding or the Exemplar View.

3 CORE DESIGN RATIONALE

Below, we present the core facets of VIGOR's design and discuss how they support sensemaking for query results.

3.1 Leveraging Examples: Bottom-Up Exploration

Starting with low level details is often referred to as a bottom-up sensemaking [31, 36]. Starting from a known example can greatly improve the development and understanding of a query [49]. We designed the Exemplar View (Figure 3) to provide the following: (1) an arrangeable visualization of the typed input query for fast error-checking; (2) easily accessible information on how many values a particular node from the query finds in the results (e.g., does an author node in a query match to only 3 authors or 3,000?); (3) the ability to start from a familiar result and relax constraints to find other results; and (4) a fast mechanism to add node value constraints to filter down the number of results.

At every step of relaxation in (2) or filtering in (3), the analyst sees real-time updates (in dropdowns in the Exemplar View and as filtering in the Fusion Graph) as the number of possible results changes. Conversely, if the analyst adds new node value constraints

3.2 A View From Above: Top-Down Exploration

High level overviews, like the Subgraph Embedding (see Figure 4), have proven useful in visualization models for sensemaking in other datasets [28, 31]. An overview of subgraph results is challenging, because: the number of subgraphs is large, the subgraphs may share nodes and edges, and each subgraph is made of multiple nodes that each have separate (and often very different) features.

To overcome these challenges we represent each result as a square glyph (to differentiate from the circles used for nodes) rather than nodes and edges, to simplify plotting. The Subgraph Embedding has the strengths of a scatterplot (including concave hulls around clusters) of all the results based on their nodes' features. The Subgraph Embedding allows zooming, panning, jitter, and fine-grain control over embedding and clustering. We group similar results with concave hulls, because there are many cases in which convex hulls overlap unnecessarily. New clusters can be freely created using a freeform lasso tool. Similar results are plotted close to each other and often form clusters as in [45]. The details of our graph embedding algorithm are discussed further in Section 4.

3.3 Feature-centric Sensemaking for Result Clusters

Typically, when an analyst poses a query they have constrained only some of the potential features of their results; the remaining features are free to vary and often form patterns. Feature distributions [41] and node-feature distributions [35] have proven a valuable way to compare results. To compare these features, we created the Feature Explorer (Figure 6), which provides node feature and value distributions by node type for a cluster. The lasso can be used to create new clusters, even from within other clusters or combining them. Multiple clusters can be compared at once by selecting them in the Subgraph Embedding.

3.4 Coordination in Multiple Views

VIGOR utilizes linked highlighting and filtering so that changes made in one view are reflected in the others. The Exemplar View highlights the Subgraph Embedding and filters or highlights the Fusion Graph based on node-value constraints. Clicking squares or clusters in the Subgraph Embedding: allows the selection an exemplar result in Exemplar View for bottom-up exploration, filtering or highlighting the Fusion Graph, and allows for the selection of different clusters in the Feature Explorer. Hover over a node in the Fusion Graph: highlights the node's neighbors and the results containing that node in the Subgraph Embedding. An analyst can choose to filter or highlight the Fusion Graph with the Exemplar View and Subgraph Embedding, with filtering the default.

4 METHODOLOGY & ARCHITECTURE

In the following section we outline our novel feature-aware, subgraph-result embedding for reducing subgraph-results to 2D points. While dimensionality reduction is common in other areas of visualization, visualizing graph query results has seen significantly less advancement. Dimensionally reducing subgraphs requires: (1) a graph embedding to turn each subgraph into a high-dimensional vector and (2) distance-preserving reduction techniques to reduce the dimensionality of each subgraph, without losing underlying similarities. We combine both structural features from the network topology as well as features from the nodes. Often some nodes may have missing values or different types making

4.1 Embedding Subgraphs

For our embedding, we utilize both network topology features as well as the rich domain features from our nodes. The embedding pipeline takes four stages from result set to low-dimensional representation. The steps of the pipeline are (see Figure 8):

- *Extract Features* - Calculate the topological- and node-features.
- *Vectorize* - Merge the common features into per-result vectors.
- *Aggregate & Normalize into Signature* - Reduce the large input vectors into uniform signatures.
- *Reduce & Cluster* - Reduce the signatures using dimensionality reduction to fit them into 2D.

Our Subgraph Embedding reduces query results (each is a subgraph) into points via a subgraph embedding for visual results similar to [45]; however, our approach differs in several key areas outlined below.

Extract Features We use both the node-features f_s and a small set of topologically extracted features f_t as inputs to our embedding (Figure 8A and 8B). There are many different ways to extract features from a graph. We started with the structural features from [45] and NetSimile, [4], for structural features. Based on our experiments using structural features alone is insufficient in our case. Often our subgraph results have significantly fewer nodes than both previous approaches and have exactly the same network structure. Because of the identical structure of our subgraphs the embedding from [4] will project all the results into a single point.

We integrate some of the novel features from NetSimile, but leave several out as they did not perform well on our induced subgraphs. Unlike both approaches we make use of the node features from the results themselves in our embedding. This means that different nodes with similar features will be closer to each other, increasing the chances of semantically meaningful and explainable clusters. In the case of real world data nodes may be missing values, which makes a purely feature-driven comparison between results imbalanced (as some results may have features that others do not). We address this problem by converting the raw features to fixed-length signatures, which we cover in the *Aggregate & Normalize into Signatures* subsection. Which node features to use are chosen by the analyst in a network schema configuration done once during VIGOR setup.

Assume we have received k results, where each result is composed of n nodes. For just the structural features we look at each result in the context of the original network and extract subgraph neighborhood and egonet information from the underlying graph. An egonet of a node, i, is the neighbors of i, the edges to these neighbors and all the edges among neighbors. This performs significantly better for small queries by structurally differentiating them based on their place in the underlying data. The most effective structural features are:

- *Node degree* - or the number of neighbors
  $$d_i = |N(i)|$$
  where $N(i)$ is the set of neighboring nodes of node i.
- *Egonet edges* - the sum of inter-neighbor edges of node i
  $$E(ego(i)) = \sum_{j \in N(i)} \left(\sum_{e_{jk} \in E(j)} \delta_{ik} \right),$$
  $$\delta(ik) = \begin{cases} 1 & \text{if } k \in N(i) \\ 0 & \text{if } k \notin N(i) \end{cases},$$
  where $e_{j,k} \in E(j)$ are the edges at node j to node k.

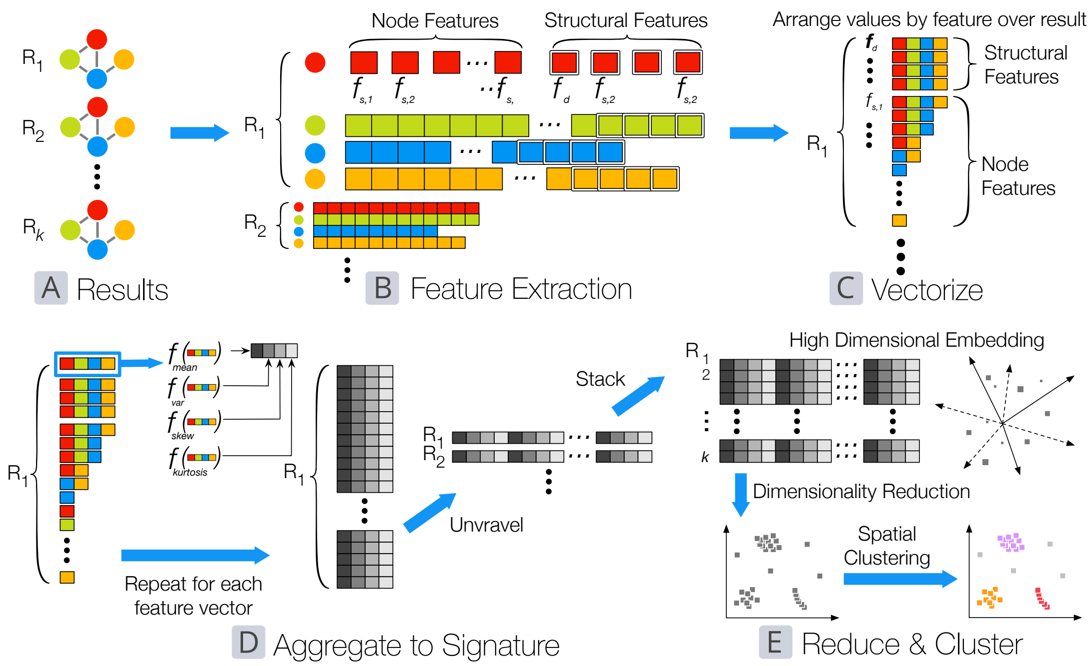

Fig. 8. Given a set of k results (A), we first extract topological features (B) from the neighborhood around each result in the underlying graph, which are combined with features from each node in the result. Next the values are rearranged by feature (C). These feature sub-vectors are run through the moment of distribution functions (mean, variance, skewness, and kurtosis), which collapse the original sub-vectors of different lengths into new uniform-length vectors (D), each is unraveled into a signature for the result, which are unit-normalized and dimensionally reduced (E). The low-dimensional space is clustered before the results are presented (E).

- *Egonet neighboring nodes* - the total number of neighbors across all the neighbors

$$|N(ego(i))| = \left| \bigcup_{j \in N(i)} N(j) \right|,$$

- *Clustering coefficient* - the fraction of closed triples over total triples from the neighbors of node i

$$c_i = \frac{2|e_{jk} \in E(i) : j,k \in N(i)|}{|N(i)| \cdot (|N(i)| - 1)},$$

Vectorize Each node of a result now has the four structural features from above and any non-text features from the nodes themselves (e.g., for an author node in our DBLP graph, we have additional features like the number of coauthors, number of conferences, etc.). This creates an issue, both because a result has k different feature vectors and also the different types of nodes in the result will have different lengths of features (see Figure 8C). The first problem we solve by vectorizing the features per result. We merge common features across the nodes into a single vector per feature for each result (Figure 8B).

Aggregate & Normalize To solve the issue of uneven lengths of the per-result feature vectors we convert them into a signature (see Figure 8D). We aggregate each vector down to a fixed number of values such that the signatures are all the same length. We utilize the moments of distributions to reduce the feature vectors into a fixed length signature. We use the first 4 moments: mean, variance, skewness, and kurtosis. For robustness we cannot use the mean and variance alone, because both structural features and node-features may not be normally distributed. The skew moment measures the lopsidedness of a distribution while the kurtosis gives a measure of how heavy the tail of the distribution is. We perform these for each feature vector per result and wrap them into a single array, yielding a new signature of length $4 \cdot (|f_s| + |f_t|)$, where f_s and f_t are the sets of features from the nodes and the structure respectively.

Reduce & Cluster We then perform dimensionality reduction to reduce the dimensions to two (see Figure 8E). There are many dimensionality reduction techniques both linear and nonlinear. We default to Principle Component Analysis (PCA) [19], but allow the analyst to choose among kernel-PCA [38], multidimensional scaling (MDS) [23], and t-Distributed Stochastic Neighbor Embedding (t-SNE) [25]. We chose to offer PCA first due to its fast performance and simple linear nature.

Both MDS and t-SNE allow arbitrary distance functions rather than the Euclidean distance. For both MDS and t-SNE we compute the Canberra distance (or weighted L_1 Manhattan distance) [24] rather than the Euclidean distance. We chose Canberra because it is sensitive to small changes near zero, which helps preserves small distances in the final reduction. It has also performed well on real datasets [15].

We perform clustering on the dimensionally reduced points (see Figure 8E). There are many density-based clustering algorithms like DBSCAN [37] or OPTICS [22]. We use OPTICS to perform our density-based clustering, because it performs better on clusters with different densities [22]. Because the choice of ε greatly affects the resulting clusters, we allow the user to adjust the value via a slider. The cluster information in encoded as colors in the Subgraph Embedding.

4.2 Architecture

VIGOR uses a client-server architecture using D3 and jQuery for the front-end and python for the back-end. The network data are stored using the popular Neo4j graph database. We chose Neo4j for its cross-platform support, robust querying language, and its scalability to large graph datasets. One of our goals is to offer VIGOR as a flexible sensemaking tool that works on a wide variety of network datasets. Our design separates the underlying network schema from the system, so that VIGOR can easily by used on different network data.

Performance VIGOR is a practical working prototype analytical system; the queries shown in this paper are all returned within 1-2 seconds. We achieve this performance through Neo4j indices and asynchronous computation of dimensionality reductions. Because the different dimensionality reductions techniques have significantly different run times, we return PCA (the fastest) first to maintain the interactivity of the system and subsequently return the others in the background.

5 Evaluation

We performed a two-part user evaluation of VIGOR (Section 5.1). In the first part, we compare VIGOR against Neo4j, a leading graph DMBS. Neo4j is an industry leader among the few free systems that visualize graph query results. In the second part, we performed a think-aloud investigation of the Subgraph Embedding, because there is no analog in Neo4j against which to compare.

To study how VIGOR can help with solving real-world problems, we collaborated with three security researchers at Symantec[2], the leading security company, to identify blindspots in the understanding of critical security incidents. In Section 5.2, we present the investigative analysis performed and insights gleaned from using VIGOR on an cybersecurity incident-network.

The details of the analyzed graphs are outlined in Table 1.

Network Type	Node	Edges	Node Types
DBLP	115,989	1,543,792	5
Cybersecurity	17,651	384,172	3

Table 1. Graph datasets used for evaluation: DBLP dataset for user study; cybersecurity dataset for real-world application to discover security blindspots.

5.1 User Study

To evaluate VIGOR, we conducted a user study to assess how well our new visualization techniques compare to the current state-of-the-art Neo4j interface. Previous research has focused on how analysts can visually construct queries [7, 32, 49]; however, our research focuses on how well analysts can make sense of and solve tasks given a set of query results. We chose a DBLP co-authorship graph, because the concepts are relatively simple and accessible to non-expert participants.

Our protocol has two parts: (I) comparative tasks, (II) a think-aloud exploration study. In Part I, we measured the number of errors and time taken solving a set of tasks for both VIGOR and Neo4j. In Part II, we asked participants to perform some open-ended exploration objectives after giving them a tutorial on the think-aloud protocol.

5.1.1 Participant Demographics

We recruited a total of 12 participants via our institutions local mailing lists. They ranged in age from 21 to 31, with 25 as the average. Of the participants, 7 were female, while the rest were male. Each study lasted on average 70 minutes, for which the participants were each paid $10 for their time.

5.1.2 Protocol

We utilized a within-subjects experimental design with two systems (VIGOR and Neo4j) and two task sets. Each system was tested with one of two sets of tasks (see the subsequent *Task* section). Participants completed the first set of tasks with the first system and the remaining task set with the second system. System order was counterbalanced to ensure experimental fairness. Task sets were also counterbalanced for fairness.

Participants were given an introduction to the dataset and tutorials of each system before being given the tasks. We encouraged participants to ask questions at any time during the study, but especially during the introductory period. For Neo4j we created an interactive Neo4j tutorial tailored to our dataset and instructed participants on Neo4j's interface and its features. For VIGOR we provided an interactive tutorial of the interface, how to filter results, and how to interact with our views.

Once a participant had completed tutorial for their current system, we provided them with context in the form of a scenario based around each query; participants were not asked to write queries. We then instructed them to work quickly and accurately on each task. Each task was allotted five minutes and was timed separately. Participants could only move onto the next task once they had completed the current one, or if time ran out. Incorrect answers were recorded for each task, including if they ran out of time before answering.

[2]We invited our Symantec collaborators to join as coauthors of this work.

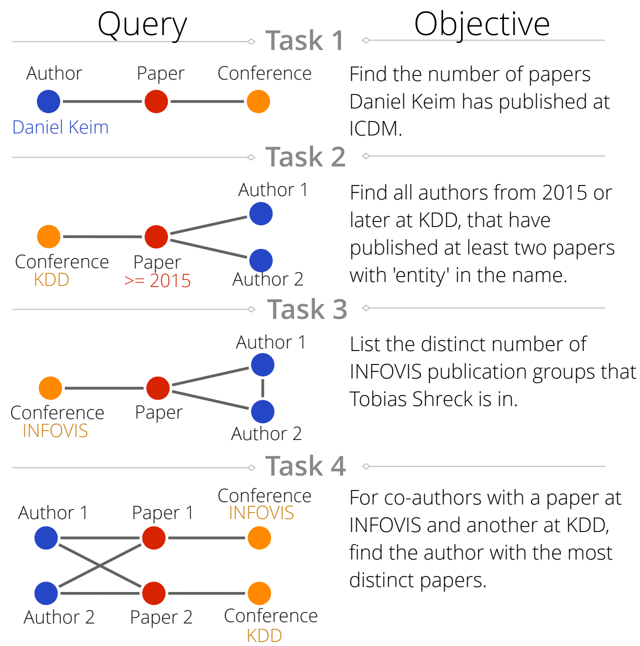

Fig. 9. VIGOR user study comparative tasks. These tasks were provided to create the result sets used in Part I of our user study. Both task sets utilized the same query topologies, but different values, carefully selected to have the same number of results.

Once a participant had completed all the tasks with a system, they would repeat the same process with the next system (including the system demonstration). Participants were not informed which system, if either, was developed by the examiner. After a participant had completed both comparative tasks, we asked them to complete Part II, the think-aloud exploration study. At the end of the study, participants completed a questionnaire that asked for subjective impressions about each software system.

5.1.3 Part I: Comparative Study

Tasks Our interest was in testing the speed of solving simple tasks with a collection of results rather than the speed of writing queries. For each task the participant was provided a short scenario and a pre-written query. The patterns for the tasks were based on common patterns and motifs from prior graph mining research [14, 21, 33]. The tasks from Task Set A (shown in Figure 9) are:

1. Find the count of *ICDM* conference papers by *Daniel Keim* in our dataset.
2. From the last two years of *KDD* publications, find and list the authors who are on more than one paper with "*entity*" in the name.
3. Find the number of distinct groups of researchers that *Tobias Shreck* is in from *INFOVIS* publications.
4. Among coauthors of at least two papers together at *INFOVIS and KDD*, who has the most publications.

The tasks approximately increased in difficulty from 1 through 5. We ranked the difficulty of each task based on the number of nodes, edges, complexity of the query, and size of the results. Our initial intuition was that Neo4j and VIGOR would achieve similar performance for the easier early tasks, while VIGOR would be faster for harder queries.

Error rate and task completion time were the dependent measures. Both measures could be affected by: (1) Software (VIGOR or Neo4j); (2) Task Set (Set A or Set B); (3) Software Order (VIGOR or Neo4j going first). Because of the within-subjects design we utilized a Latin Square design randomizing each participant into one of four groups where we counterbalanced the possible confounding factors (e.g., one group is *(*VIGOR + *Task Set A) then (Neo4j + Task Set B)*).

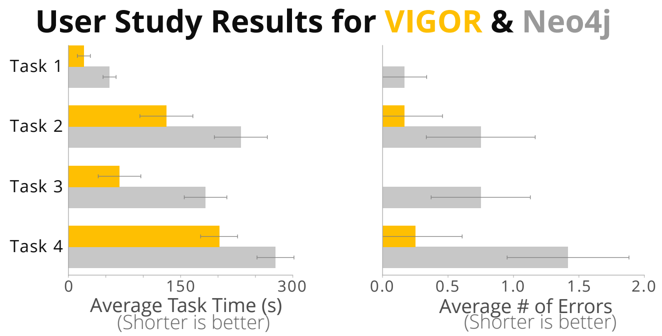

Fig. 10. Average task completion times and error rates for VIGOR (yellow) and Neo4j (gray). VIGOR is statistically significantly faster across all tasks. Error bars represent 95% confidence interval.

Quantitative Results We analyzed task completion times usinng mixed-model analysis of variance (ANOVA) with fixed effects for *software, software order, task set*, and a random effect across *participants*. Mixed-model ANOVA improves over conventional ANOVA as errors are calculated per-subject.

Our task completion times were measured over all combinations of software order and task set. The experiment was successful as the only statistically significant effect was from software system. Figure 10-left demonstrates the average time per task in our study. The software effect was significant for each task: task 1 ($F_{1,11} = 29.79,\ p < 0.0003$), task 2 ($F_{1,11} = 41.02,\ p < 0.0001$), task 3 ($F_{1,11} = 33.68,\ p < 0.0002$), task 4 ($F_{1,11} = 23.89,\ p < 0.0006$). Only task 3 ($F_{1,11} = 12.27,\ p < 0.0057$), and task 4 ($F_{1,11} = 19.6,\ p < 0.0013$), had statistically significant error rates. This is expected as the error rates for the first tasks were very low. The second task in Task Set A came close to significance with ($p < .048$), likely arising from slightly higher number of edges in the induced subgraph than in Task Set B. Participants were both significantly faster and less prone to error with VIGOR versus Neo4j.

Subjective Results At the end of the study we asked participants to rate various aspects comparing both systems using Likert scales. Participants felt that VIGOR was better than Neo4j for all 7 aspects asked (Figure 11). One participants stated, "I enjoyed the clustering features of VIGOR, allowing the user to quickly compare variables (Year, etc.) about any possible combinations of groups.". The participants enjoyed using VIGOR more than a Neo4j and reported that our system was: easier to learn, easier to use, and more likeable overall; although this is a common experimental effect, we find the results encouraging.

5.1.4 Part II: Think-aloud Exploration Study

After the comparative tasks were completed, all participants were asked to perform a think-aloud exploration study. We chose to separate this part of the study from Neo4j as it tests new features that are not present in Neo4j's interface. This part of the study was not timed.

Our goals for the think-aloud study were:

- **Feature interactions:** were our features were working well together, and whether VIGOR met their basic exploration needs.
- **Identify usability issues:** were features usable and if they coordinated in beneficial ways during their exploration.
- **Feature application:** what techniques participants would use with VIGOR and whether its functionality would help streamline their analytics workflow.

High-level Objectives We provided participants with a pair of scenarios and high-level objectives to complete. We asked participants to imagine themselves as researchers interested in:

1. the features from all papers by Jiawei Han or Christos Faloutsos at PKDD and SIGMOD; and
2. understanding the outlier results (results distant from a cluster) for co-authors of papers at VAST and KDD or INFOVIS and KDD.

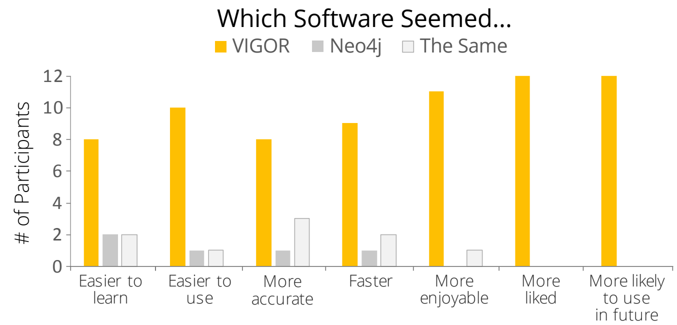

Fig. 11. Participants were asked to qualitatively compare each system at the end of each trial. Overall, they felt that VIGOR was better than Neo4j in all of the 7 aspects asked.

We provided the queries for both tasks. Participants were free to use any features of VIGOR and ask questions during the objectives. We chose the above objectives, because they are common in graph analysis [10, 26].

Key Observations During the first objective, 6 participants began their exploration by searching for PKDD and SIGMOD using the Exemplar View to find the conferences. Another 4 of the participants went directly to using the Fusion Graph to highlight results in the Subgraph Embedding by hovering over specific conferences. The remaining 2 participants used the Feature Explorer's conference type to investigate which clusters contained PKDD and SIGMOD conferences. For 4 of the 12 subjects they had considerable difficulty with their first few lassoing attempts, often completely missing the desired nodes. Only 2 participants failed to adequately complete the objective.

In the second objective, 10 participants started by creating new clusters by lassoing groups of outliers to compare them against the existing clusters. The remaining 2 used the Fusion Graph to highlight results in the Subgraph Embedding for particular nodes. Of 12 participants 3 reported that they had not found any satisfactory explanations for outliers, while the remaining 9 either found specific papers or features not present in the cluster. One participant correctly commented that several of the outliers arise from single-author papers, because multi-author papers have a higher chance of being repeated across the results (and therefore have a higher chance of being similar to other results). Overall participants performed very well using the coordinated views in VIGOR.

5.1.5 Discussion and Limitations

The qualitative and quantitative results of our user study were positive. The results suggest that VIGOR provides useful and effective visual techniques for analyzing and making sense of graph query results. VIGOR achieves this improved performance through: (1) streamlining the filtering process to allow users to quickly narrow down by a particular author (Task 3), or by a particular term in papers (Task 2); (2) the flexibility and customization of the Fusion Graph graph layout (all Tasks); and (3) the Subgraph Embedding, which makes grouping and comparing the results easy (Part II).

While Neo4j is an industry leader, we found two specific design-choices (based on participant feedback) that limited performance with Neo4j: (1) the default *edge-autocomplete*, add any underlying edge from the network (regardless of its inclusion in the query); and (2) the instability of the force directed layout positions during node dragging.

We did not evaluate query creation and modification. Our study did not evaluate query creation and refinement; participants were given the query that corresponds to a scenario investigating co-authorship, which may not be the most natural query that they would like to create. If we allowed participants to create ad hoc task queries, the immense variety of possible queries would make the evaluation extremely difficult. Moreover, query refinement, a challenge that would add additional confounding factors to the study, would also require participants to have more prior knowledge [32]. Even the queries provided were challenging to many participants, as demonstrated by the high error rates in

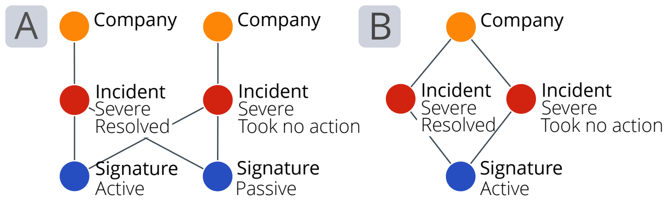

Fig. 12. Example queries from our exploration of cybersecurity blindspots. Query A reveals companies ignoring critical security incidents that their peers resolve, whereas Query B extracts companies responding inconsistently to critical security incidents.

Task 4.

We were pleasantly surprised to see that participants were able to use and compare features using the cluster-based distributions in the Feature Explorer (Figure 6) and that they could use when comparing more than two clusters.

While our evaluation was very positive, the real-world scenarios and initial queries of analysts would be ad hoc. We plan to study this case and better understand how VIGOR can handle tasks in less planned situations. For example, how would analysts utilize the Subgraph Embedding for significantly different domains, transportation networks, intelligence, bioinformatics?

5.2 Real World Application: Discovering Cybersecurity Blindspots

We collaborated with three security researchers from Symantec to identify blindspots in their company company's understanding of critical security incidents. They see strong potential in VIGOR to help them educate their company customers about these weak points in their response to dangerous security situations. We used VIGOR in the following ways to identify company blindspots and bring them into focus: (1) we contrasted companies that tend to ignore critical security incidents with peers that face the same types of incidents but exhibit exemplary incident response (see Figure 12A), and (2) we highlighted instances in which companies do not respond consistently to critical security incidents such as vulnerability scans and malware outbreaks (see Figure 12B).

Symantec Cybersecurity Network To pose these types of queries, we created a cybersecurity network composed of Company nodes (orange), Incident nodes (red) and Signature nodes (blue). In total, this network of security data contains 17,651 nodes and 384,182 edges. All of these entities correspond to real world events and represent the detection of and actions taken against various security threats. Companies are linked to the security incidents that were detected on their systems, and each incident is in turn linked to the signatures that were responsible for triggering it. Signatures that are responsible for the creation of security incidents are designated as "Active Signatures" that typically identify glaring security issues, such as malicious network traffic and computer viruses. Security products also define "Passive Signatures", whose primary purpose is to provide contextual information about such things as login behavior and other system or network events. The active signatures trigger security incidents of various levels of severity, of which critical incidents are the most important, and are the basis of the incidents used in case study, as they should be met with immediate investigation and resolution, but frequently are not.

Query 1 - Comparing Company Incident Behavior Our first query (see Figure 12A) identifies pairs of companies faced with critical security incidents that consist of at least one active and one passive signature, such that one company resolved its incident while the other company ignored it. By posing this VIGOR query and examining its results (ref. Figure 13), we identify a company's (Company 7) blindspots in a way that simultaneously provides interactive graphical evidence that another company (Company 16) is faced with similar security incidents and takes them seriously. The results of this query can also be used as an educational tool and shared directly with companies as evidence of their most glaring blindspots. Doing so helps companies

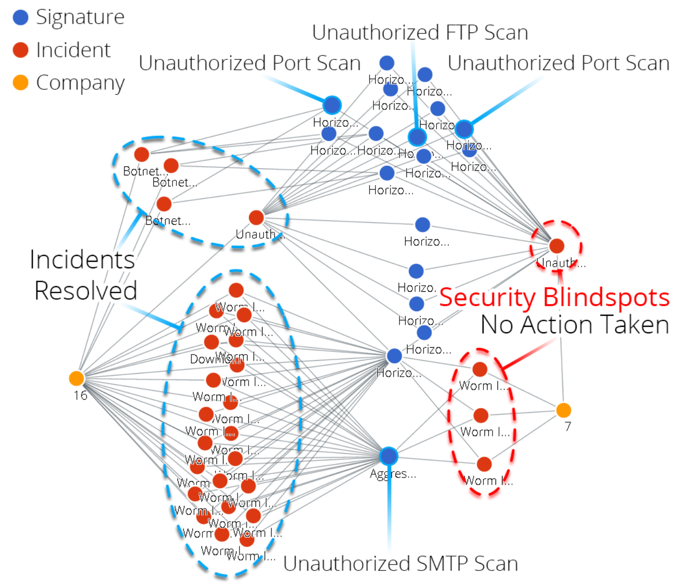

Fig. 13. Results of our first blindspot detection query (see Figure 12A). VIGOR identifies Company 7's blindspots with evidence that Company 16 is faced with similar security incidents and takes them seriously.

re-evaluate and react differently to future instances of incidents that they would have otherwise continued to ignore.

Query 2 - Inconsistent Company Threat-Reactions Of similarly concern are situations in which a company reacts inconsistently to the same type of security incident. By posing the query of Figure 12B, we identify incidents that companies respond to inconsistently. This query provides a company with the ability to identify blindspots at the finer-grained level of its individual incident responders, some of which may understand the perils of a particular type of security incident much better than others do. Example results are shown in Figures 14A and 14B. In both cases, VIGOR identifies malware outbreaks that were not fully eradicated. Further outbreaks of the malware are likely in both cases. The malware of Figure 14A could be spreading by means of the unmitigated unauthorized internal vulnerability scans that happened in a similar timeframe. Similarly, internal machines still infected by the trojan malware of Figure 14B could be used by an attacker to re-establish a firm foothold within the targeted company since the compromised machines were not all cleaned. An additional benefit of this VIGOR query and visualization is that it functions very well as a progress checker after a major security issue, allowing companies to track their progress as they work to ensure that malware outbreak are fully eradicated from the environment. Finally, Figure 14A and 14B both highlight the way in which the Feature Embedding is able to cluster related security blindspots in two dimensions for efficient perusal.

6 Related Work

Graph Visualization and Query Languages Visualizing graphs is a challenging topic that has attracted significant research interest and motivated the development of many tools and techniques. Herman et al. [16] conducted an extensive survey of much of the foundational work. More recently developed techniques for static graph visualization have been covered in [47] and for dynamic graphs in [3]. Graph sensemaking and interaction have grown in popularity [31]. Our work extends this large body of research by providing a visual approach to graph query result understanding and exploration.

Query By Example [49], is an early bottom-up querying system allows users to formulate queries by creating templates from "example queries" rather than writing conventional SQL statements. Another key innovation is to abstract the exact underlying data schema away from analysts as in PICASSO [20], which uses visual glyphs to create visual database queries. Both [1] and [40] avoid complex data schemas

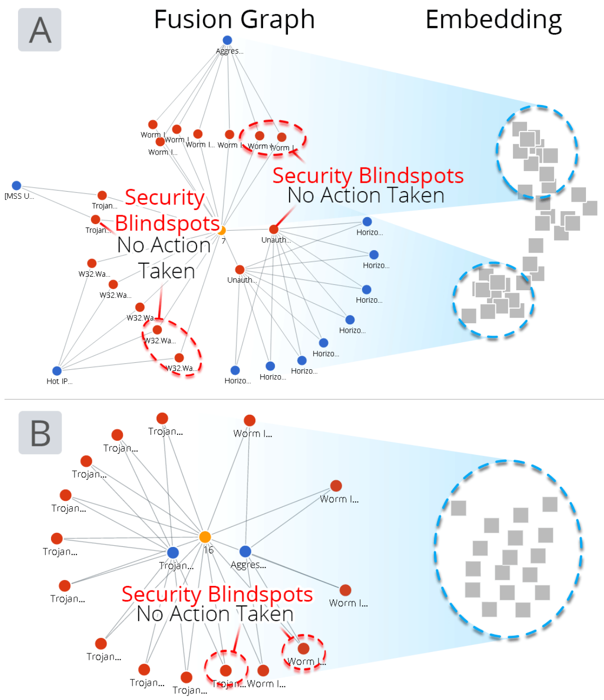

Fig. 14. Results of our second blindspot detection query (see Figure 12B). VIGOR identifies companies that respond inconsistently to critical malware-related security incidents, while majority of the incidents are resolved (circled in green), some received no action (circled in red).

in favor of graphical widgets. For a further detail of visual querying languages on relational databases see [8]. Data storage techniques like the extensible markup language (XML) and resource description framework (RDF) have spurred other querying languages like XQUERY [6], XPATH [12], SPARQL [30]. Both [9] and [27] propose graphical querying languages for XML, while Hogenboom et. al propose one for RDF data [17]. Our work builds on visual querying by using visual metaphors for both the query and the results.

Graph Querying and Graph Databases Algorithmically determining if a given subgraph exists within another graph is referred to as the subgraph isomorphism problem and is *NP-Complete* [13]. Tong et al. proposed to use random walk with restart probabilities to heuristically score approximate matches in G-Ray [44]. The MAGE algorithm follows a similar heuristic and supports a much wider set of possible input queries [33]. Tian et al. utilize approximation indices to provide real-time approximations to a user specified query [43]. While we use Neo4j as our underlying database, most graph matching frameworks work with the ideas we propose in this paper.

Visual Graph Querying There are a few recent visual graph querying systems, many focus on the construction of queries rather than the presentation of results. GRAPHITE [11], allows users to visually construct a graph query over categorically attributed graphs. VOGUE [5], is a query processing system with a visual interface that interleaves visual query construction and processing. Cao et al. created g-Miner, an interactive multivariate graph mining tool that supports template matching and pattern querying [7]. VISAGE [32], is another graph querying tool which guides the user using graph-autocomplete, or pre-fetched results to help guide analysts towards results. GRAPHITE, VOGUE, and VISAGE all use lists to present the results to the user and focus considerably less on the result visualization than the query formulation. Our work aims to fulfill this gap in the current research, by proposing new methods to summarize and explore graph query results.

Summarizing Graphs, Kernels and Embeddings Another line of research focuses on “summarizing” graphs. Koutra et al. [21] propose VoG, which constructs a vocabulary of subgraph-types like stars and cliques to simplify visualization. Dunne and Shneiderman [14] present motif simplification, wherein common patterns or motifs are replaced with easily understandable glyphs (e.g. fans and cliques), which was subsequently applied to biological networks in MAVisto [39]. Rather than replacing structural elements, graph kernels and embeddings allow graphs to be converted in the vectors and scalars. There are numerous types of graph kernels and kernel similarity methods [46]. Both [48] and [46] use the structure to create the embedding while NetSimile, [4], uses extracted features. Van den Elzen et al. used graph embedding to plot the changes in dynamic graph snapshots over time [45]. We draw on some of these ideas in our Embedded Results view to collapse each result down to a point, the details of which will be discussed in the *Methodology & Architecture* section.

7 Discussion and Future Work

When implementing visual graph querying systems, we must grapple with two different scalability concerns; the visual and the computational. The visual scalability of our system is primarily limited by the Fusion Graph, which quickly accumulates large numbers of nodes and edges. By using the Exemplar View, and the Subgraph Embedding, analysts can quickly filter down the Fusion Graph to manageable sizes. The computational scalability of our model is most limited by the dimensionality reduction techniques like t-SNE and MDS, while PCA and kernel-PCA run in under a second. The time to fetch the query results was often trivial compared to the time needed for the embedding pipeline.

We offer several forms of dimensionality reduction, because dimensionality reduction is challenging and the best solution often depends on the underlying data. The choice of which dimensionality reduction method as well as the parameters (ε and n_{neigh}) for OPTICS clustering have been left up to the user. Theses choices vary greatly with the underlying characteristics of the network data and suggest that the best options should come from collaboration between a visualization expert and a domain expert. In our experience, the nonlinear dimensionality reduction techniques worked much better for clustering on most graphs; however, the axes of these approaches are much harder to interpret. Both t-SNE and MDS do a better job at preserving the small distances between the high dimensional points than conventional PCA and this likely leads to better clustering performance. VIGOR might benefit from an approach that automatically detects the dimensionality with the best clustering.

Currently VIGOR applied our system to exact subgraph matches; however, new systems may also produce approximate subgraph matches. Because the approximate results are not identical in shape and content, the result set becomes much more complex. Additional visualization techniques are needed to show where and how approximate results do not match the original query.

8 Conclusions

Visualizing graph query results is challenging, requiring effective summarization of a large number of overlapping subgraph results, each having complex network structure and rich node features. We presented VIGOR, a novel visual analytics system for exploring and understanding graph querying results.

VIGOR supports top-down and bottom-up result sensemaking, through its (1) exemplar-based interaction technique, where an analyst starts with a specific result and relaxes constraints to find other similar results or starts with only the structure (i.e., without node value constraints), and adds constraints to narrow in on specific results; and (2) a novel feature-aware subgraph result summarization. Through our collaboration with Symantec, we demonstrated how VIGOR helps discover security blindspots in a cybersecurity dataset with over 11,000 incidents. We also evaluate VIGOR with a within-subjects study, demonstrating VIGOR's ease of use over a leading graph database management system, and its ability to help analysts understand their results at higher speed and make fewer errors.

Acknowledgments

This research has been supported in part by NSF IGERT grant 1258425, NSF grants IIS-1563816, TWC-1526254, and IIS-1217559.

REFERENCES

[1] C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic queries for information exploration: An implementation and evaluation. In *Proceedings of Conference on Human Factors in Computing Systems*, pp. 619–626, 1992.

[2] R. Angles and C. Gutierrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1):1:1–1:39, Feb. 2008. doi: 10.1145/1322432.1322433

[3] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. The state of the art in visualizing dynamic graphs. In *EuroVis - STARs*, pp. 83–103. Eurographics Association, 2014. doi: 10.2312/eurovisstar.20141174

[4] M. Berlingerio, D. Koutra, T. Eliassi-Rad, and C. Faloutsos. Netsimile: A scalable approach to size-independent network similarity. *CoRR*, abs/1209.2684, 2012.

[5] S. S. Bhowmick, B. Choi, and S. Zhou. Vogue: Towards a visual interaction-aware graph query processing framework. In *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*, 2013.

[6] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu. Xquery 1.0: An xml query language, 2002.

[7] N. Cao, Y.-R. Lin, L. Li, and H. Tong. g-miner: Interactive visual group mining on multivariate graphs. In *Proc. CHI*, pp. 279–288. ACM, 2015.

[8] T. Catarci, M. F. Costabile, S. Levialdi, and C. Batini. Visual query systems for databases: A survey. *Journal of Visual Languages & Computing*, 8(2):215–260, 1997.

[9] S. Ceri, S. Comai, P. Fraternali, S. Paraboschi, L. Tanca, and E. Damiani. Xml-gl: A graphical language for querying and restructuring xml documents. In *Proceedings of the Italian Symposium on Advanced Database Systems (SEBD)*, pp. 151–165, 1999.

[10] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[11] D. H. Chau, C. Faloutsos, H. Tong, J. I. Hong, B. Gallagher, and T. Eliassi-Rad. Graphite: A visual query system for large graphs. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pp. 963–966, 2008.

[12] J. Clark, S. DeRose, et al. Xml path language (xpath) version 1.0, 1999.

[13] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pp. 151–158, 1971.

[14] C. Dunne and B. Shneiderman. Motif simplification: Improving network visualization readability with fan, connector, and clique glyphs. In *Proceedings of the ACM SIGCHI Conference on Human factors in Computing Systems (CHI)*, pp. 3247–3256, 2013.

[15] S. M. Emran and N. Ye. Robustness of chi-square and canberra distance metrics for computer intrusion detection. *Quality and Reliability Engineering International*, 18(1):19–28, 2002. doi: 10.1002/qre.441

[16] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE TVCG*, 6(1):24–43, 2000.

[17] F. Hogenboom, V. Milea, F. Frasincar, and U. Kaymak. Rdf-gl: A sparql-based graphical query language for rdf. In *Emergent Web Intelligence: Advanced Information Retrieval*, Advanced Information and Knowledge Processing, pp. 87–116. Springer London, 2010.

[18] K. J. Holyoak and P. Thagard. *Mental Leaps: Analogy in Creative Thought*. MIT Press, Cambridge, MA, USA, 1995.

[19] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[20] H.-J. Kim, H. F. Korth, and A. Silberschatz. Picasso: A graphical query language. *Softw. Pract. Exper.*, 18(3):169–203, Mar. 1988. doi: 10.1002/spe.4380180302

[21] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. Vog: Summarizing and understanding large graphs. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pp. 91–99, 2014.

[22] H.-P. Kriegel, P. Krger, J. Sander, and A. Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011. doi: 10.1002/widm.30

[23] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964. doi: 10.1007/BF02289565

[24] G. N. Lance and W. T. Williams. Computer programs for hierarchical polythetic classification (similarity analyses). *The Computer Journal*, 9(1):60–64, 1966.

[25] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[26] M. E. Newman. Coauthorship networks and patterns of scientific collaboration. *Proceedings of the national academy of sciences*, 101(suppl 1):5200–5205, 2004.

[27] W. Ni and T. W. Ling. Glass: A graphical query language for semi-structured data. In *Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA)*, pp. 363–370, 2003.

[28] C. North and B. Shneiderman. Snap-together visualization: A user interface for coordinating visualizations via relational schemata. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '00, pp. 128–135. ACM, New York, NY, USA, 2000. doi: 10.1145/345513.345282

[29] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pp. 201–210. ACM, 2007.

[30] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of sparql. In *International semantic web conference*, pp. 30–43. Springer, 2006.

[31] R. Pienta, J. Abello, M. Kahng, and D. H. Chau. Scalable graph exploration and visualization: Sensemaking challenges and opportunities. In *2015 International Conference on Big Data and Smart Computing, BIGCOMP 2015, Jeju, South Korea, February 9-11, 2015*, pp. 271–278, 2015. doi: 10.1109/35021BIGCOMP.2015.7072812

[32] R. Pienta, A. Tamersoy, A. Endert, S. Navathe, H. Tong, and D. H. Chau. Visage: Interactive visual graph querying. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '16, pp. 272–279. ACM, New York, NY, USA, 2016. doi: 10.1145/2909132.2909246

[33] R. Pienta, A. Tamersoy, H. Tong, and D. H. Chau. MAGE: matching approximate patterns in richly-attributed graphs. In *2014 IEEE International Conference on Big Data, Big Data 2014, Washington, DC, USA, October 27-30, 2014*, pp. 585–590, 2014. doi: 10.1109/BigData.2014.7004278

[34] P. Pirolli and S. Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*, vol. 5, pp. 2–4, 2005.

[35] Z. L. J. V. P. T. J. A. G. P. D. H. P. C. Robert Pienta, Minsuk (Brian) Kahng. Facets: Adaptive local exploration of large graphs. In *SIAM International Conference on Data Mining (SDM) 2017*, 2017.

[36] D. M. Russell, M. J. Stefik, P. Pirolli, and S. K. Card. The cost structure of sensemaking. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pp. 269–276. ACM, New York, NY, USA, 1993. doi: 10.1145/169059.169209

[37] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Min. Knowl. Discov.*, 2(2):169–194, June 1998. doi: 10.1023/A:1009745219419

[38] B. Schölkopf and C. J. Burges. *Advances in kernel methods: support vector learning*. MIT press, 1999.

[39] F. Schreiber and H. Schwbbermeyer. Mavisto: a tool for the exploration of network motifs. *Bioinformatics*, 21(17):3572–3574, 2005.

[40] B. Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77, 1994.

[41] J. Stahnke, M. Dörk, B. Mller, and A. Thom. Probing projections: Interaction techniques for interpreting arrangements and errors of dimensionality reductions. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):629–638, Jan 2016. doi: 10.1109/TVCG.2015.2467717

[42] A. Tamersoy, E. Khalil, B. Xie, S. L. Lenkey, B. R. Routledge, D. H. Chau, and S. B. Navathe. Large-scale insider trading analysis: patterns and discoveries. *Social Network Analysis and Mining*, 4(1):1–17, 2014.

[43] Y. Tian and J. Patel. Tale: A tool for approximate large graph matching. In *IEEE International Conference on Data Engineering (ICDE)*, pp. 963–972, 2008.

[44] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 737–746, 2007.

[45] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):1–10, Jan 2016. doi: 10.1109/TVCG.2015.2468078

[46] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borg-

wardt. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, Aug. 2010.

[47] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J.-D. Fekete, and D. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011. doi: 10.1111/j.1467-8659.2011.01898.x

[48] L. A. Zager and G. C. Verghese. Graph similarity scoring and matching. *Applied Mathematics Letters*, 21(1):86 – 94, 2008. doi: 10.1016/j.aml.2007.01.006

[49] M. M. Zloof. Query-by-example: A data base language. *IBM Systems Journal*, 16(4):324–343, 1977.