Structure-Based Suggestive Exploration: A New Approach for Effective Exploration of Large Networks

Wei Chen, Fangzhou Guo, Dongming Han, Jacheng Pan, Xiaotao Nie, Jiazhi Xia, and Xiaolong Zhang

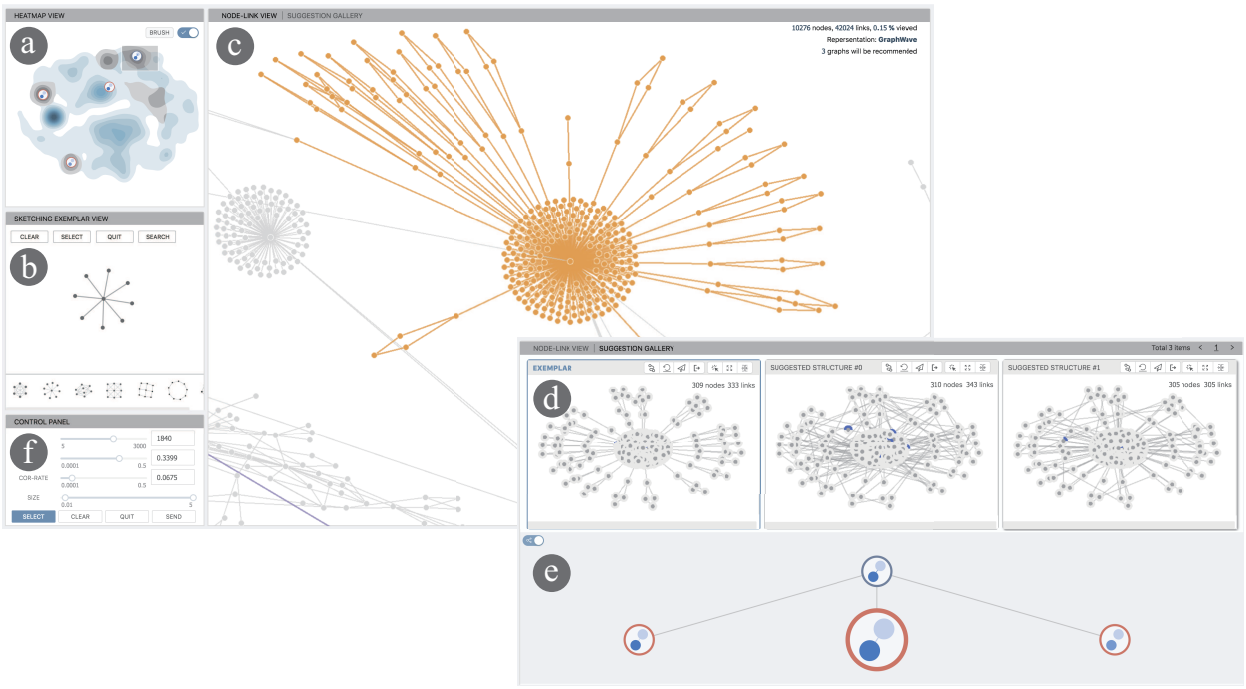


Figure 1. The user interface of our prototype system. (a) a heatmap view to show the rough shape of a network; (b) a sketching exemplar view where users can specify a structure exemplar manually; (c) a node-link view to present the structure of the network; (d) a suggestion gallery to visualize similar structures detected by a query engine; (e) an exploration history view to show exploration history; and (f) the control panel to enable users to adjust parameters of querying structures.

Abstract— When analyzing a visualized network, users need to explore different sections of the network to gain insight. However, effective exploration of large networks is often a challenge. While various tools are available for users to explore the global and local features of a network, these tools usually require significant interaction activities, such as repetitive navigation actions to follow network nodes and edges. In this paper, we propose a structure-based suggestive exploration approach to support effective exploration of large networks by suggesting appropriate structures upon user request. Encoding nodes with vectorized representations by transforming information of surrounding structures of nodes into a high dimensional space, our approach can identify similar structures within a large network, enable user interaction with multiple similar structures simultaneously, and guide the exploration of unexplored structures. We develop a web-based visual exploration system to incorporate this suggestive exploration approach and compare performances of our approach under different vectorizing methods and networks. We also present the usability and effectiveness of our approach through a controlled user study with two datasets.

Index Terms—Large Network Exploration, Structure-Based Exploration, Suggestive Exploration

1 Introduction

Network data has been widely used in many fields to describe relationships among entities, such as social relationships between people in sociology, interactions between proteins in biology, and transactions between companies in finance [77]. However, the efficiency and accuracy of analyzing a network are greatly influenced by its size because analysts often have little knowledge about where to start the analysis and where to find interesting patterns. Visual exploration offers an interactive means to sense the underlying network and gain insight in an exploratory way [47, 65, 67, 75].

Various techniques to support large network exploration have been

- *W. Chen is with State Key Lab of CAD and CG, Zhejiang University. E-mail: chenwei@cad.zju.edu.cn.*
- *F. Guo, D. Han, J. Pan, and X. Nie are with State Key Lab of CAD and CG, Zhejiang University. E-mail: {guofangzhou, dongminghan, anxis, and net}@zju.edu.cn.*
- *J. Xia is with Central South University. E-mail: xiajiazhi@csu.edu.cn.*
- *X. (Luke) Zhang is with Pennsylvania State University. E-mail: lzhang@ist.psu.edu.*
- *W. Chen and F. Guo contribute equally to this paper. They are sorted by the alphabetic order of their last names.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

developed [16, 64, 70]. They usually follow two general strategies. A top-down strategy [8, 33] provides an overview of network structures and guides analysts to local details by filtering or querying. The bottom-up strategy [50], on the other hand, shows local details upon the request of analysts and supports network exploration by following nodes and edges of interest. Under both strategies, analysts have to narrow down to local regions frequently and investigate the details. It is also a common practice for analysts to switch between different levels of details to navigate the entire network, or traverse the network stepwise along nodes or edges. Thus, automatic recommendation of appropriate views or structures has been suggested as a more effective and user-friendly approach for large network exploration [14, 16, 57].

In particular, exemplar-based structure suggestion can assist users to analyze and compare multiple similar structures quickly in examining a large network. Here, we define a structure, or a subgraph, of a network as a relationship among a set of connected nodes and an exemplar as a structure of interest, which is specified by users. A structure describes a certain local network pattern. The main challenge for suggesting structures by exemplars lies in the lack of computationally-efficient methods to support real-time structure-based exploration in large networks. Existing methods that detect structures in networks, such as subgraph matching [15] and motif discovery [28], require high computational cost or constraints on networks, such as networks must be labeled [17]. In this paper, we present a novel visual exploration approach that suggests appropriate structures upon user-specified exemplar. Our approach employs a representation-and-querying scheme: prior to exemplar-based query among all structure candidates, a vectorized representation of each structure is pre-computed. We design and implement a web-based system for the exploration of structures of interest in a network. Starting from an exemplar, our system supports interactive query, identification, comparison, and analysis of one or a set of structures scattered in a network. In summary, the contributions of this paper include:

- A novel structure querying algorithm that leverages a vectorized representation for nodes in a network. This algorithm is essential to interactive visual exploration of large networks;
- An efficient suggestive exploration scheme that supports visual exploration of structures in large-scale networks; and
- A web-based exploration system to support efficient exploration of large-scale networks.

2 Related Work

Our research work concerns the development of a novel network-representation approach to support interactive exploration of large networks. Thus, we review research literature in the areas of network representation and visually-guided large-network exploration.

2.1 Representations of Networks

The term of representation can refer to two different but related concepts: visual representation and data representation.

Visual representations of networks can be classified into three major categories: node-link diagram, matrix representation, and hybrid methods. The key issue in node-link diagram is the spatial out of nodes, and numerous methods have been proposed [24, 31, 45, 46]. Purchase et al. [60] discussed the importance of keeping the mental map for users to understand the evolution of networks. Force-directed layout algorithms are most widely used. Algorithms like FM^3 [30] and ForceAtlas2 [36] can process the layout of large networks with fast speed. Matrix representation uses an adjacency matrix to visualize a graph, and usually each non-diagonal matrix cell represents an edge. Ordering rows and columns appropriately can effectively reveal typical netwrok patterns such as clusters [51]. Ghoniem et al. [23] compared node-link diagram and matrix representation, and concluded that node-link diagrams are more intuitive and more suitable for path-based tasks, while matrix representations are more compatible with dense graphs. However, matrix representation faces a space scalability issue. Hybrid methods, such as NodeTrix [34], use matrix and node-link diagram to represent different components in a network (e.g., matrix for local communities and node-link for connections among communities). In this paper, we adopt node-link diagram to enable interactively specifying structures.

Data representation of a network concerns ways to describe nodes and edges of a network mathematically. Recently, vectorized representation, an approach to embed nodes or structures into a high-dimensional space, has been popular in data mining [27]. Various structure-preserving approaches have been proposed. Feature-based methods [6, 57, 69] can measure a set of features and formulate a vector representation of a node. Van den Elzen et al. [69] flattened the adjacency matrix as a vector and attached derived attributes at the end of it to construct the final representation. Pienta et al. [56] aggregated the node attributes and topological information of the neighborhoods of a structure into a vectorized signature. The graphlet kernel method [63] vectorized a graph by counting the frequencies of graphlets, which are small, induced, and non-isomorphic subgraph patterns [59]. Kwon at al. [40] used graphlet kernels to calculate similarities among large graphs. In addition, some learning-based methods [52, 54] have been developed by machine-learning researchers. Such algorithms as Node2Vec [29], Struc2Vec [61], and GraphWave [18] use representation learning to vectorize a node or structure.

2.2 Visual Exploration of Large Networks

Visual exploration techniques are widely used in large network analysis and network sensemaking [48, 55, 74]. These techniques usually follow one of the two major strategies: top-down exploration or bottom-up exploration.

Providing an overview of the entire network is an intuitive way to help users explore the data [8, 33, 76]. However, the escalating size of networks increases the computational cost to generate an overview of large networks, as well as the cognitive burden of users to explore the networks. At the data level, methods such as clustering [3, 66], sampling [43], and filtering [37] have been used to reduce the number of objects in an overview. At the visualization level, techniques like edge bundling can help to reduce view cluttering [35, 80]. Interaction tools, such as expansion [41, 70] and zooming [20, 68], are usually combined with these methods. However, users still have to perform many interaction activities to explore very large networks and may not always know where exploration should start in an overview due to the lack of necessary details about network structures.

Bottom-up techniques provide another way to explore a large network. Under this approach, users often start from a single node or a small structure of the network, and explore other nodes connected or relevant to the shown nodes. Some techniques in this category, such as Link Sliding and Bring & Go [50], are purely based on network topology, and support topology-based exploration. Some designs supported the exploration of large networks in focus+context visualization with degree-of-interest (DOI) functions [2, 10, 22, 25, 38]. For example, Van Ham et al. [70] used a DOI function to extract a maximal interest subgraph around a searched node and enable users to expand the subgraph in any direction. Recently, Srinivasan et al. [64] designed Orko, a system to support network exploration with multimodal interactions.

For networks with nodes having properties, node similarity-based methods can be used to support user navigations by identifying relevant nodes automatically [14, 16, 26, 57]. Various visual analytics systems support the exploration of large networks with query mechanism, such as path analysis [12, 39, 53], visual query and query results analysis [7, 13, 56, 58]. Zhao et al. [78] showed a technique to support exploring explicit and implicit relations in datasets. In addition to node similarity-based methods, subgraph-based methods [9] have also been proposed to support large-network exploration. A representative way is to support users analyze graphs by motifs [19, 72, 73], which are predefined graph patterns. Von Landesberger et al [72] presented a system that supports analysis of directed, weighted networks by filtering and aggregating structures based on pre-defined or user-specified motifs. However, finding motifs in a network is a time-consuming process and thus is not ideal for supporting analysis of large networks (e.g., a network with more than 10,000). Lenz et al. [42] proposed a visual analytics system for exploration of motifs in directed acyclic networks, which is not designed for free exploration in simple graphs. By measuring the

similarities among graphs, clustering of graphs [73] and finding relevant graph in a set of graphs [71] can be achieved. Similarly, Behrisch et al. [5] compared matrices with varying sizes by measuring the distances among matrices in a low-dimensional space. However, these methods are designed for exploring of a set of networks instead of a single large network.

Yet, existing solutions typically require users to perform a large amount of interactive activities in discovering interesting patterns or exploiting the entire structure. In this paper, we present a novel structure-based suggestive exploration scheme to reduce the workload on free navigation and frequent investigation of local structures.

3 Overview of Our Approach

In this paper, we focus on visual exploration of unlabeled simple networks, i.e., undirected unweighted networks without self-loops and multiple edges. We address this challenge with a new representation-and-querying scheme: prior to online network exploration, a vectorized representation of the network structure is pre-computed. With this representation, nodes are regularized into a multi-dimensional space. This conversion actually facilitates effective analysis and comparison of nodes, and consequently makes it amenable for structure-based query and exploration. Different from previous subgraph matching algorithms [15], which either require user-defined labels or have relative high time complexity for unlabeled networks, our representation-and-query scheme requires no label information, and is suitable for online structure-based visual exploration of large-scale networks.

In this paper, we call a structure specified by users for query an *exemplar*. Our approach is designed to support rapid exploration of a large network by suggesting structures on exemplars. Users can simultaneously explore, analyze, and compare multiple regions of a network triggered by simple exemplar-specification interaction activities. Meanwhile, this suggestion scheme facilitates the propagation of user specifications on a node to other related nodes, effectively reducing the workload in exploring multiple regions. With the detected structures, users can be further guided to unexplored regions.

The basic workflow of structure-based suggestive exploration is shown in Figure 2. The workflow can be summarized into three steps: (1) specifying an exemplar through user interaction; (2) providing users with suggested structures that are topologically similar to the exemplar; users can explore and verify the suggested structures; and (3) modifying or revising the suggested structures by iteratively exploring the network.

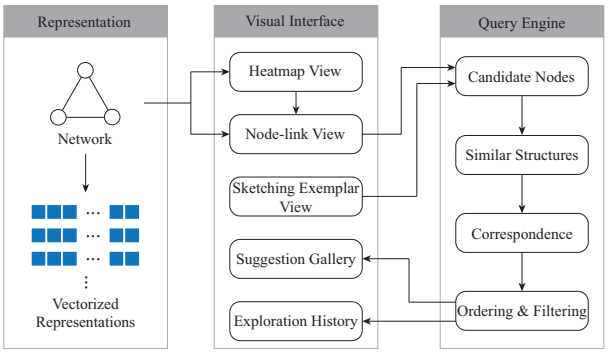


Figure 2. Overview of our approach. Vectorized representations are first calculated in a preprocessing step. The visual interface shows the network topology in the heatmap view and the node-link view and vectorized representations in the node embedding view. After an exemplar is given, the query engine searches similar structures, which are further analyzed and explored in the visual interface.

Following the above process, users can explore a large-scale network efficiently iteratively. In such an exploration, multiple regions, which are possibly far apart in the network, are shown simultaneously to users.

4 Detection of Structures

Before a detailed description of the representation and query of structures, we define the terms in Table 1. An unlabeled simple network can be denoted as $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is a set of n nodes, and $E = \{e_1, e_2, ..., e_k | e_i = (v_m, v_n), v_m, v_n \in V\}$ is a set of edges linking nodes in V.

Table 1. Definition of symbols

Symbol	Description
$N(v)$	Neighbors of node v
$Ego_i(v)$	i-hop ego-network of v
g_s	A specified exemplar
$kNN(g_s)$	The set of k nearest neighbors of nodes in g_s
G_{kNN}	The subnetwork formed by nodes in $kNN(g_{sel})$
C	A connected component in G_{kNN}
P_s	Clusters of nodes in g_s partitioned based on similarity
P_C	Clusters of nodes in C partitioned based on similarity and P_s
p_s	A cluster in P_s
p_C	A cluster in P_C

4.1 Vectorized Representation

The key idea of our approach is to generate a regularized representation for each node (and its relevant local structure) and to enable vector-based similarity measure and query. It is crucial for a representation to provide the structure information of nodes (e.g., the structural similarities among nodes). We choose five types of techniques from existing representations, including GraphWave [18], Graphlet Kernel [49], Node2Vec [29], Struc2Vec [61], and Feature-based method [6, 56, 69]. The impacts of different vectorized representations on the quality of suggested exemplars is evaluated in Section 6.2.

4.2 Specifying Exemplars

We offer two modes for users to specify exemplars of their interests, including selection and sketching. The selection mode enables users to specify exemplars in the exploring network using lasso tool or node-wise selection. The sketching mode allows users to sketch their aiming structures in the sketching panel. While we have the vectorized representation of the whole network, the vectorized representations of selected exemplars can be directly obtained. However, in the sketching mode, because the sketched exemplar is newly created, we need to generate its vectorized representation online.

4.3 Querying Structures

After an exemplar is specified, similar structures are queried through a four-step process: 1) constructing the set of candidate nodes in the vector space according to similarities among the vectorized representations of nodes; 2) detecting connected components in the set of candidate nodes according to the topology of the original network; 3) calculating the correspondences among nodes in the specified exemplar and candidate; and 4) ordering the detected components based on their structural similarities to the exemplar. Next, we describe these steps in detail.

Constructing the set of candidate nodes. We build the set of candidate nodes by selecting nodes that are similar to nodes in the exemplar. Specifically, given a structure containing m nodes, we check k nearest neighbors of each node and combine them as the candidate nodes set $\mathbb{N}$. Noting that the nearest neighbors of different nodes may overlap, the size of $\mathbb{N}$ would be less than $m \times k$. The k nearest neighbors of a node is formed according to the similarity among the vectorized representation of nodes. We compute the cosine distance with Graphlet kernel, Node2Vec, and Struc2Vec methods, and obtain Euclidean distance with the feature-based and GraphWave methods. Generally, a small similarity between a pair of nodes indicates that they have similar local structures and vice versa.

Detecting structures Given k nearest neighbors of m nodes in an exemplar, the size of the exact search space is k^m. Fully traversing this space to find similar structures faces two challenges. First, the time complexity is too high to support interactive exploration. Second, it only supports exact matching, i.e., the nodes of a structure must have a

one-to-one mapping to the exemplar. In reality, structures are seldom exactly the same, so query must tolerate certain differences in structure.

Based on the above observations, we propose to detect connected components in the subgraph (see Algorithm 1), which are composed by the candidate nodes and their edges. The detected components are considered as candidate target structures. The assumption here is that a suggested structure must be a connected component in the original network, and each node in it must be similar to a certain node in the exemplar. It is possible that connected or overlapped structures are detected as one component. Visually presenting composted structures can help users understand the relationships among them. On the contrary, explicitly generating separated structures may yield redundant structures and lead to heavy perception burden to users. Thus, we keep the connected or overlapped structures unchanged.

Algorithm 1 Query Similar Structures

Input: G: the network; g_s: the exemplar; ε: the minimum similarity between two nodes; k: value of k in kNN search; *Sim*: the ordered similarity matrix
Output: $\mathbb{C}$: detected connected components

```
for all n_i in g_s do
    Count_{n_i} = 0
    for all n_j in Sim[n_i], n_j ∈ G do
        if Count_{n_i} > k then
            Break
        end if
        if Sim[n_i, n_j] < ε and n_j ∉ N_explored and n_j ∉ g_s then
            Count_{n_i} += 1
            Add n_j into G_sim
            Add n_j into N_explored
        end if
    end for
    Add n_i into N_explored
end for
ℂ ← connected_components(G_sim);
```

Constructing correspondences. For a detected structure, the mapping from nodes in each connected component C to the exemplar g_s is constructed in three steps. First, nodes in g_s are categorized into clusters P_s based on the similarity in the vector space with DBSCAN [21]. We choose DBSCAN because it detects clusters without a pre-defined cluster number. Next, nodes in each C are categorized into clusters P_C. For a node in a specific C, it is assigned to the cluster to which its most similar node in g_s belongs. In this way, a cluster in P_C also corresponds to a cluster in P_s. Third, correspondences among nodes in a cluster p_C in P_C and nodes in the corresponding cluster p_s in P_s are calculated. Because nodes in two corresponding clusters are similar in the vector space, we map every node in p_C to a node in p_s (see Algorithm 2). Specifically, we select two most similar nodes in p_C and p_s, establish the correspondence between them, and remove them from the node set awaiting for being processed.

Algorithm 2 Find Correspondence among Nodes in Structures

Input: g_s: the exemplar; C: a detected connected component; *Sim*: ordered similarity matrix; P_s: clusters of nodes in g_s;
Output: *Corr*: the correspondence between nodes in g_s and C; P_C: clusters of nodes in $\mathbb{C}$

```
for all n in C do
    p ← argmin(Σ_{n*∈p*} Sim[n, n*]), where p* ∈ P_s
    Append n into P_C[p]
end for
for all p in P_s do
    n_i, n_k ← argmin_{n_i∈p, n_k∈P_C[p]} Sim[n_i, n_k]
    Corr[n_k] ← n_i
end for
```

Ordering and filtering detected structure. Detected structures are ordered by structure similarity scores calculated with the Weisfeiler-Lehman graph kernel [62]. Then, connected components that are most similar to the exemplar are displayed. Filtering out those connected components that are not similar to the exemplar can reduce the number of components to be explored. In this paper, a connected component is filtered out automatically if its size is too small ($|C|/|g_s| < 50\%$) or it cannot be mapped to the exemplar properly ($|P_C|/|P_s| < 50\%$) by default. These two parameters are empirically set and can be adjusted interactively by users.

5 Visual Exploration

We develop a visual exploration system to support structure-based suggestive exploration in large networks. Figure 1 shows the overall user interface of the system (Figure 1). The system consists five major views: 1) a node-link view to show the detailed structure of a large network and allow users to identify interested structures; 2) a heatmap view to present a heatmap with 2D kernel density estimation of network layout in (1); 3) a sketch panel for exemplar sketching; 4) a suggestion gallery to present suggested structures; and 5) an exploration history tree to record the explored structures for later review.

5.1 The Node-link View

The node-link view (Figure 1(c)) is the major working space for network exploration and exemplar specification. This view presents the detailed structures of a large network in a pre-computed force-directed layout. It has panning and zooming tools to support navigation. A lasso tool for node selection is offered for the specification of exemplars.

5.2 The Heatmap View

To help users understand the rough shape of a network and locate interesting structures, a KDE (kernel density estimation)-based heatmap and its extracted contour lines are shown. The explored regions are marked in blue and the unexplored are in grey. The heatmap view provides a set of interactive tools, including zooming in/out, Region-Of-Interest (ROI) selection and panning, and suggestion locating.

- **Zooming** To support free navigation at different level-of-details, multi-level contour lines are extracted with different bandwidths.
- **ROI selection and panning** Users can select the ROI by drawing a rectangle in the overview. Users can panning the ROI by dragging it and exploring the details in the node-link view.
- **Suggestion locating** When an exemplar is specified, it and the associated suggestions are encoded as glyphs in the heatmap view. Users can locate the suggestions by clicking the glyphs.

Glyph design Glyph is used to indicate structure of interest. A glyph should abstract the structure of suggestions and inform users the similarity between a suggestion and the specified exemplar. Our glyph design consists of two components (see Figure 3): an outer circle and an inner node-link diagram. The outer circle is color-coded to specify whether a structure is user-specified (in blue) or is system-suggested (in orange). In the inner node-link diagram, each node represents a cluster that is identified in the query step (see Algorithm 1). Two nodes are connected if two corresponding clusters are connected by at least one edge. The node-link diagram is generated with respect to the user-specified exemplar. In the node-link diagram of a suggestion, a missing node indicates that the corresponding cluster is not found. When users hover on a glyph, missing nodes are shown as dashed circles. The tone of a cluster node encodes the number of nodes in this cluster. The size of glyphs scales along with zooming of the heatmap view. We also set minimum and maximum size of glyphs to improve the scalability of the heatmap view.

Design Alternatives. We also considered other three glyph design alternatives in the design process (Figure 4). In the first design (Figure 4(a)), the total number of nodes is encoded by the color of an inner circle and the radial line chart surrounding the circle encodes the number of nodes in each cluster. However, users can hardly understand the exact number of nodes in the radial line chart. The second design (Figure 4(b)) removes the inner circle and employs a radial bar chart to encode the node number of each cluster. Its color represents the number of all nodes. This design is inefficient when the number of clusters is

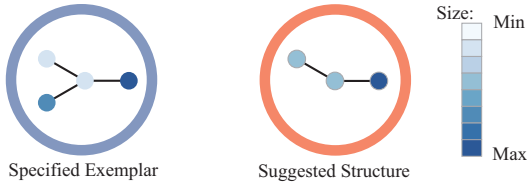


Figure 3. Glyph design used in the heatmap view and the exploration history view. Nodes in a selected structure and a suggested structure are categorized into clusters. The glyph consists of an outer circle and an inner node-link diagram. The color of the circle shows the type of the node. The node-link diagram encodes each cluster with a node, the color of which indicates the size of the cluster.

large. Accordingly, the third design (Figure 4(c)) removes the axes that correspond to clusters that are not found. It is not adopted because it lacks intuitiveness and does not scale well with the number of clusters.

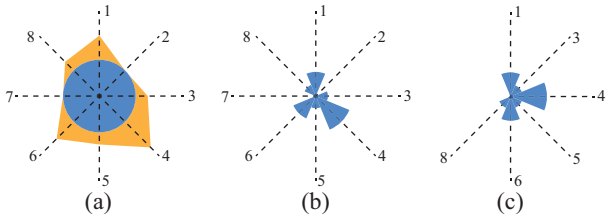


Figure 4. Three design alternatives for the structure glyph. Each axis represents a node in an exemplar with 8 nodes: (a) a design to encode the size of suggested structure by the color of the inner circle and the correspondence of nodes by a star glyph; (b) a design to encode size by color and correspondence by bar length with a radial bar chart; and (c) a design that revises (b) by removing nodes have no correspondence in the suggested structure.

5.3 The Sketching Exemplar View

The sketching exemplar view (see in Figure 1b) supports a set of interactive tasks related to an exemplar, including adding an exemplar based on templates, adding nodes/edges in an exemplar, and deleting nodes/edges. This view is designed for situations in which users have a target in mind and want to sketch a desired exemplar. Basically, adding nodes/edges of an exemplar allows users to start from scratch. The templates, which are summarized by Bach et al. [4], provide suggestions for novice users to begin their sketching. For experienced users, adding an exemplar based on templates can be more efficient. The combination of these interactive tasks offers a flexible and efficient way for exemplar sketching. For instance, a precise exemplar can be created by first selecting a template structure and then adding/deleting nodes and edges.

5.4 The Suggestion Gallery

The suggestion gallery juxtaposes the suggested structures in a descending order based on their similarities to the specified one. Users can click the "Page Down" and "Page Up" buttons to explore the suggestion galleries. The exemplar is always positioned at the left most of the gallery for comparison. It is crucial to layout the presented structures consistently for comparison. The layout is computed by considering the correspondences among structures through 3 steps: (1) Laying out the specified exemplar by using the force-directed layout algorithm. (2) Setting the initial layout of each suggested structure. Each node in a suggested structure is mapped to a node in the exemplar. The initial position of a node is set to the position of its corresponding node in the exemplar. (3) Adding perturbation into the initial layout to avoid overlapping. When there are multiple nodes that map to one node in the exemplar, visual clutter occurs. To avoid this problem, positions of nodes are jittered by taking a few iterations with the force directed layout algorithm. In our implementation, three iterations have been proven to be effective.

Expanding structures consistently. We design a consistent structure expanding technique to support simultaneous exploration the suggestions. Expanding the neighboring structures are useful for: 1) verifying the similarity in a larger scale; and 2) exploring new structures. A consistent expanding can help to build a consistent mental model during the exploration. The main challenge is the consistency of the layouts of the expanded neighbors. Our solution for this challenge consists of three steps, as illustrated in Figure 5. First, we cluster the expanded nodes in the vectorized space using DBSCAN. The number of clusters is automatically decided by the clustering algorithm. Second, when representing each cluster as a node, we layout the cluster nodes by using the force-directed algorithm. Third, for each structure, we layout its expanded nodes. We use positions of corresponding cluster nodes as the initial positions, and perform three iterations with the force-directed layout algorithm to jitter nodes in the same cluster. Users can recursively expand a structure and merge adjacent expansions.

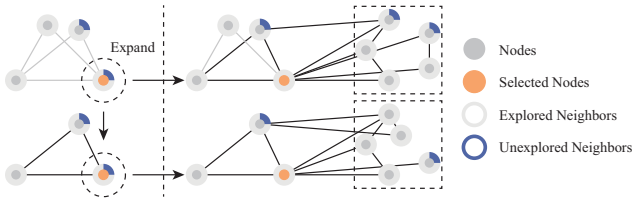


Figure 5. Encodings of nodes in the suggestion gallery and layout after node expansion. The layout of expanded nodes are firstly calculated by force-directed layout algorithm and then placed on the right of explored nodes. The color of the inner circle indicates if the node is selected. The outer ring encodes the ratio of explored neighbors (the grey part) and unexplored neighbors (the green part).

To denote where potential neighbors are for exploration, we design a glyph (see Figure 5). The glyph consists of an inner circle and an outer ring. The color of the inner circle represents the status of nodes, including previous nodes (grey), newly expanded nodes (blue), or selected nodes (orange). The outer ring is divided into two halves: the grey half represents the expanded neighbors and the green half represents the remaining neighbors. The angle of a half encodes the ratio of corresponding neighbors.

5.5 The Exploration History View

Once an exemplar is specified, the suggested structures are shown in the exploration history view (Figure 6) with a forest structure. This view provides an overview of suggested structures and helps users interactively review exploration history at any time. When users click a tree node, the corresponding suggestion history will be shown in the suggestion gallery view.

Construction of the exploration history forest. There are two types of nodes in the tree. The first type represents exemplars, called exemplar nodes. The second type represents suggested structures, called suggested nodes. Initially, the forest is empty. An exemplar node is inserted as a root into the forest, once an exemplar is specified. The suggested structures are appended to the exemplar node. If a new exemplar is specified when users exploring around suggested exemplars, a new exemplar node is appended as a child of the node being explored. If a newly specified exemplar is irrelevant to any nodes in the forest, a new root is generated. Eventually, a forest is dynamically generated. Edges between nodes in the forest can be classified into two categories. The first category represents the suggestion relation (Figure 6(a)), i.e., a suggested node is generated based on an exemplar node. The second category represents the exploration relation (Figure 6(b)), i.e., an exemplar node is interactively identified around another node.

Visual Design. The forest is visualized by a series of dendrograms (Figure 6). The dendrograms are placed vertically along the y axis, which represents time (see in Figure 6). Nodes are depicted with the same glyph design used in the heatmap view. To distinguish two types of edges in the forest, nodes that are connected by suggestion relations

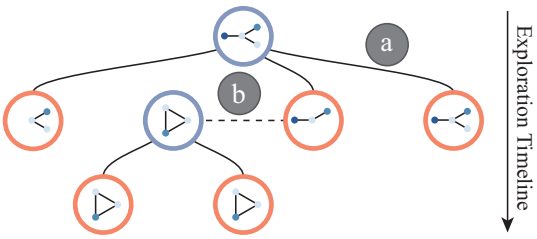


Figure 6. Visual design used in the exploration history view: (a) the suggestion relation from an exemplar structure to a suggested structure; and (b) the exploration relation from a suggested structure to an exemplar structure. The structures are encoded with the glyph described in Section 5.2.

are arranged vertically, while nodes that are connected by exploration relations are arranged horizontally, as shown in Figure 6.

5.6 The Control Panel

The control panel enables users to control the quality of suggested exemplars (Figure 1(f)). Users can set four parameters, k, ε, $|C|/|g_s|$, and $|P_C|/|P_s|$, before the suggestions are generated. By adjusting k and ε, users can change the search scope of the query algorithm in the high dimensional space. By adjusting $|C|/|g_s|$ and $|P_C|/|P_s|$, users can filter out low quality suggestions. With $|C|/|g_s|$, users can adjust the number of nodes in suggested exemplars, avoiding the size of exemplars being too large or too small compared to the specified exemplar. With $|P_C|/|P_s|$, users can adjust the correspondence between suggested exemplars and the specified exemplar, ensuring that suggested exemplars can be properly mapped to the specified exemplar. To present the effect of current combination of parameters, the number of suggested exemplars is shown on the top of the system whenever users adjust parameters.

5.7 System Use Scenario

In this section, we illustrate how our system works with a real-world network. The network used here is a Bitcoin trading network extracted from the open data on [1]. The network is a subset of trading logs on Jan 01, 2018, with 207689 nodes and 547500 edges.

Imagine a financial analyst who wants to gain some insight into trading patterns of Bitcoin, but does not have any prior knowledge on Bitcoin trading. He loads the network into our system and starts from The heatmap view, which shows three regions with higher node densities than other regions. Therefore, he begins the exploration in these regions. In the node-link view, he finds that a small number of nodes has high degree centralities, i.e., some entities, people or organizations, directly traded with a huge number of entities. Furthermore, five nodes are connected through a significant number of intermediary nodes (Figure 7). Based on his experience, he suspects that these trading activities may be related to money laundering, so he decides to explore whether there are similar patterns in other regions with lower density.

Subsequently, he examines the border of the layout by brushing in the heatmap view and finds a structure (Figure 8(a)) similar to the previously seen structure and with a reasonable number of nodes for manipulation. He selects this structure as an exemplar and then gets the suggested structures by the system (see Figure 8(b)). These structures appear in other regions and indicate similar trading patterns (Figure 9). Studying a structure glyph in the heatmap view (see Figure 7), the analyst identifies a node involved in several structures, implying repeated involvements of an entity in similar suspicious tradings. Thus, the analyst files a report to suggest further investigation on the entity.

The analyst continues to investigate other trading patterns. He has knowledge on social network analysis and knows that a star-shaped structure indicates a node with a high local centrality in the structure, which may be related to entities who are the centers of trading activities. Instead of searching for a star-shaped structure, he draws a star exemplar in the sketching exemplar view, and as expected, the system suggests a series of star structures in the suggestion gallery. By analyzing node glyphs, he finds that some nodes in these structures have many

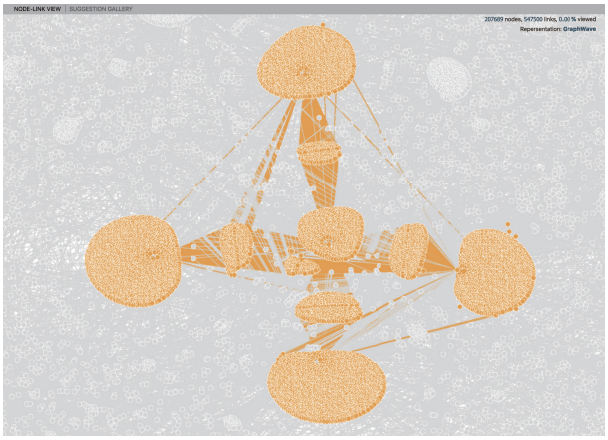
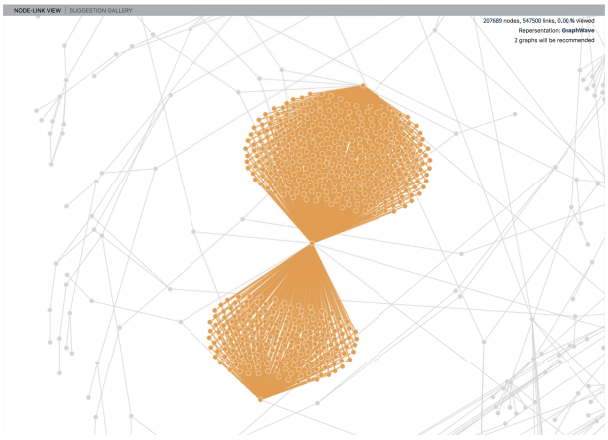


Figure 7. Structure of one of the regions with the highest density in the network. Five center nodes trade with a large amount of nodes, which only trade with one or two other nodes. Meanwhile, the center nodes are connected through a huge number of intermediate nodes.



(a)

(b)

Figure 8. Structures identified based on an exemplar: (a) an exemplar is found in a low density region, which is similar to the trading pattern in the high density region; and (b) suggested structures in other regions are shown after the exemplar is specified.

unexplored neighbors. To verify the centrality of those nodes identified as the centers in a larger scale, he expands a node in a structure of the gallery, and corresponding nodes in all other structures are simultaneously expanded. After several expansions, he finds an interesting trading pattern: the centers of two star structures are connected. This pattern appears in multiple structures (Figure 10(a)). By observing the structure in the node-link view, he realizes that this trading pattern is a frequent one in the Bitcoin trading network (Figure 10(b)). He is then interested in structures with multiple centers in the trading network, so

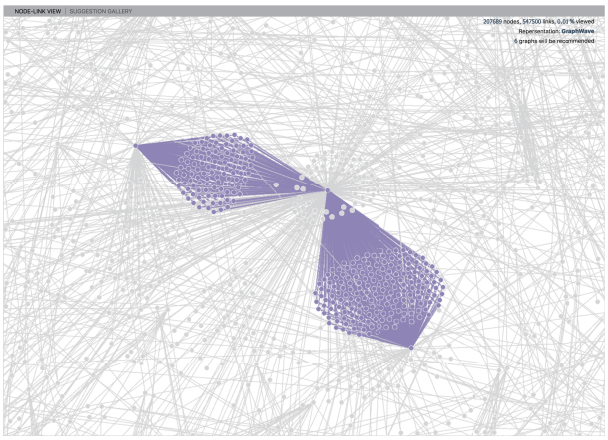
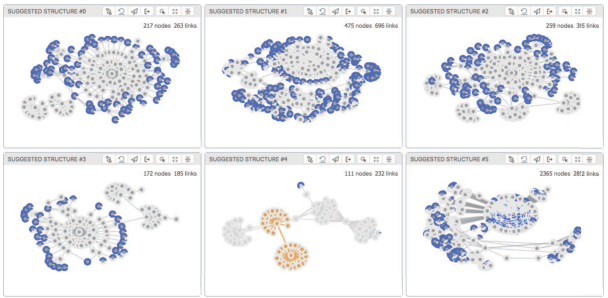
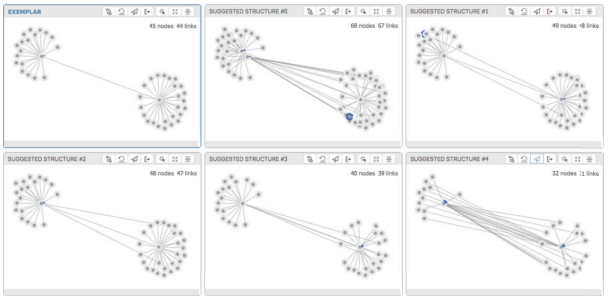


Figure 9. A suggested structure, which is very similar to the specified exemplar.

sketches an exemplar as Figure 11a and the system suggests a series of such structures (Figure 11(b)).



(a)



(b)

Figure 10. (a) Connected star structures are found in multiple suggested structures after several expansions; (b) connected star structures are suggested after a connected star structure is specified as an exemplar.

5.8 System Implementation

We implemented a web-based system in browser-server architecture. We adopted React, D3.js, and PixiJS to implement the front-end application. PixiJS was used in the heatmap view, the node-link view, and the suggestion gallery view; and D3.js was used in the heatmap view, the sketching exemplar view, and the exploration history view. We used Python 3.0 to implement the backend server. Scipy and Numpy were used for data processing. MongoDB is used for data storage.

6 Experiments

We conducted several experiments to evaluate the effectiveness of our approach with different vectorized representations of nodes in different large networks. The experiments were conducted on a PC with an Intel

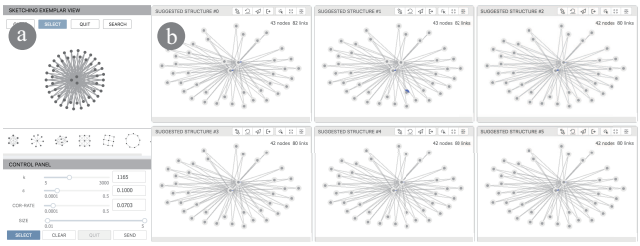


Figure 11. (a) The exemplar sketched by the user that contains three centers and a series of surrounding nodes; (b) a series of suggested exemplars similar to the sketched exemplar.

i7-4790 CPU (3.60GHz) and 16 Gigabyte RAM. The vectorized representations we used include GraphWave, Graphlet Kernel, Struc2Vec, Node2Vec, and the feature-based method. All vectorized representations of nodes were performed offline on a PC with an Intel Xeon CPU E7540 (2.00 GHz) and 188 Gigabyte RAM. We implemented the feature-based method and used open-source implementation of other four methods to calculate the vectorized representations. For the Bitcoin trading network with 207689 nodes and 547500 edges, Struc2Vec takes longest time (for about 7 days) and other methods take similar time to finish calculation (for several hours).

6.1 Datasets

Our experiments used a set of synthetic networks and two kinds of real networks: a twitter network and a set of Bitcoin trading networks.

- **Synthetic networks** consist of a series of typical structures, including stars, cliques, bipartite cores, and king's graph [11]. We generate three networks, which contains 2122, 5581, and 10926 nodes and 20944, 78505, and 192631 edges, respectively. The number of each type of structure is set to 10, 20, and 30 and the number of nodes in structures is randomly set to $[40, 60]$, $[60, 80]$, and $[80, 100]$, respectively. We conducted performance tests on all three datasets.
- **The Twitter network [44]** consists of 'circles' from Twitter. It contains 81306 nodes and 1342296 edges. This network is used in the algorithm performance evaluation.
- **Bitcoin trading networks** used in the experiments were again extracted from the open data on [1]. In addition to the network we described in Section 5.7, three other networks were extracted from the transaction data on Jan 01, 2018. These networks have 10276 nodes and 42024 edges, 104134 nodes and 75560 edges, 207689 nodes and 547500 edges, respectively.

6.2 Performance

We evaluated our method in two aspects. First, we evaluated the quality of suggested exemplars with different vectorized representations and different ways of specifying exemplars. Based on the first evaluation, we were able to choose the best vectorized representation. Then we evaluated the search speed of our method with different network sizes and different exemplar sizes based on the chosen representations and two ways of specifying exemplars.

Due to the lack of ground truth, we evaluated the performances of five vectorized representations introduced in Section 4.1 on a synthetic network with 10926 nodes. We evaluated the performances using recall, which is the ratio of found target structures to all the target structures, and precision, the ratio of found target structures to all found structures. We examined the performances of our approach on different representations with different values of k and ε based on the two ways of specifying exemplars. Results are shown in Figure 12.

The recalls and precisions of the selection mode are shown by solid lines in Figure 12. The query recalls increase with the increase of k, while ε is fixed at 0.05. When k is larger than 1600, GraphWave, Graphlet kernel, Struc2Vec and feature-based methods all achieve a precision larger than 0.8. When k is fixed at 2500, recalls of Node2Vec and Struc2Vec increase significantly with the increasing of ε, while

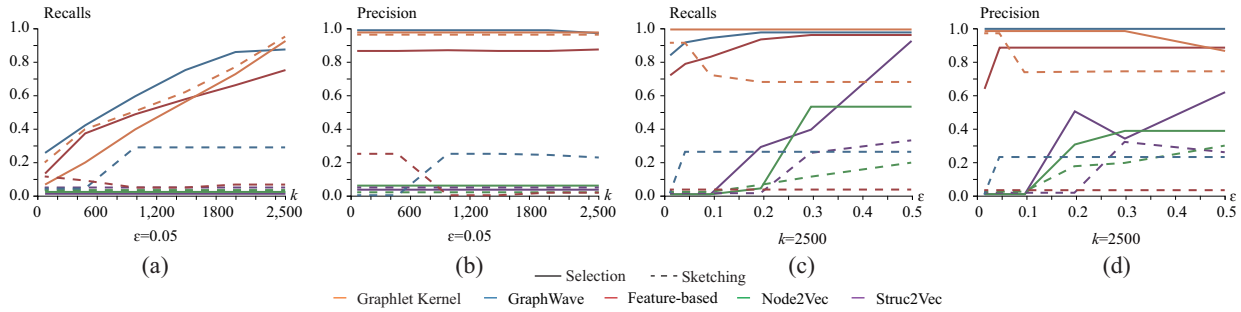


Figure 12. Recalls and precisions of our approach based on different vectorized representations when exemplars are specified by selections (solid lines) and sketching (dashed lines). (a) recalls increase with the increasing of k; (b) precisions are stable with the increasing of k; (c) recalls of GraphWave, Graphlet Kernel, and Feature-based methods are stably high while recalls of Struc2Vec and Node2Vec increase with the increase of ε when exemplars are specified by selection; recalls of Graphlet Kernel is high while recalls of other methods are low when exemplars are specified by sketching; (d) precisions of GraphWave, Graphlet Kernel, and feature-based method are stable while precisions of Struc2Vec and Node2Vec increase with the increasing of ε-neighbors in the selection mode. Precisions of Graphlet Kernel are high while others are low in the sketching mode.

the recalls of GraphWave, Graphlet Kernel, and feature-based methods change slightly. In terms of precisions, GraphWave and Graphlet Kernel outperform the other three approaches, and GraphWave is slightly better than Graphlet Kernel.

The recalls and precisions of the sketching mode are shown by dashed lines in Figure 12. When ε is fixed at 0.05, the recalls and precisions of Graphlet Kernel and Graphwave increase with the increasing of k. Graphlet Kernel achieves good recalls (> 0.6) when k is larger than 1500 and has a good precision (> 0.9). When k is fixed at 2500, recalls of Graphlet Kernel outperform than other representations. When ε is small (< 0.05), Graphlet Kernel has high recalls (> 0.9) and precisions (> 0.9). However, recalls of Graphlet Kernel decrease to around 0.7 when $\varepsilon > 0.1$. The precisions of Graphlet Kernel decrease from around 1.0 to around 0.75 when $\varepsilon > 0.1$. This is because when ε is large, in some cases, detected candidates form a large connected network, which is filtered out by our algorithm. In terms of recalls and precisions, Graphlet kernel outperforms the other approaches.

The recalls and precisions show that our method can suggest appropriate exemplars with proper parameters and vectorized representations. Also, We conclude that GraphWave works best in the selection mode, and Graphlet kernels works best in the sketching mode. Thus, we used GraphWave as the basic representation for experiments described later when exemplars are specified by selection, and Graphlet kernels as the basic representation for experiments described later when exemplars are specified by sketching.

Because the suggestion generation procedures of the two modes of specifying exemplars are different, we measured the query time based on different specification methods separately, as shown in Figure 13. The solid line shows the time to return suggested exemplars when specify exemplars by selection. In general, the time to return query results is lower than 1 second. For example, to query a network with 207689 nodes and 547500 edges, it takes less than 1 second to produce query results, when the size of the exemplar is smaller than 50. When the size of the exemplar is 100, the query time in a large network is only about 1.2 seconds. The dashed line shows that the query time to return suggested exemplars when exemplars are specified by sketching is longer than selection, which is reasonable because more calculation need to be done. To query a network with 207689 nodes and 547500 edges, the query time is less than 2 seconds, when the size of the exemplar smaller than 50. When querying an exemplar with 100 nodes, the query time is smaller than 3.5 second. In general, although the query time of the sketching mode is longer than the selection mode, the query results can be returned in less than 3.5 seconds.

Computational Complexity. The fast query speed can be explained by the good computational complexity of our algorithm. The time complexity of querying similar structures is $O(m \times |V|)$, where m is the node size of the exemplar. The time complexity of finding correspondence among nodes in structures is $O(m \times k)$, where k is the number of candidate nodes, which is smaller than $|V|$. Thus, the total time

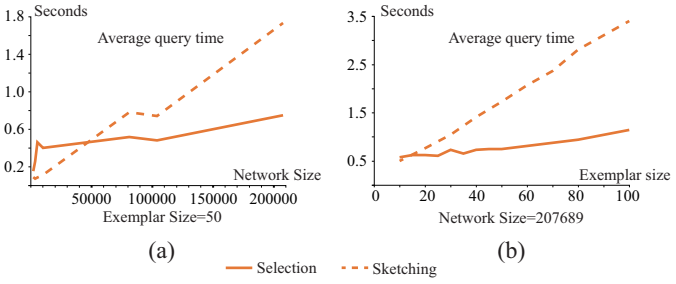


Figure 13. The query time of our approach. (a) The exemplar size is fixed to be 50. The average and max query time increase linearly with the increase of the network size; (b) the network size is fixed to be 207689. The average query time increases linearly with the increase of the exemplar size.

complexity of our method is $O(m \times |V|)$. Finding suggested structures at the level of second is important to interactive exploration. This low-delay feedback on the specification of exemplars allows users to continue their exploration activities without the interruption of their cognitive activities that is often caused by computational delay.

7 User Study

We conducted a user study to compare our approach with manual exploration. The study was a between-subject design. The manual exploration treatment was constructed with our system by removing the suggestive exploration functionality. Our hypothesis is that the structure-based suggestive exploration scheme greatly improves exploration efficiency.

Datasets. Two datasets were used in the user study: the synthetic network with 10926 nodes, and the Bitcoin trading network with 10276 nodes. The Bitcoin trading network was used for the tutorial. The synthetic network was used for the task because we had the ground truth about the network and could accurately assess user performance.

Participants. We recruited 12 student participants (7 male, 5 female). They were all familiar with visualization techniques when recruited. No participants had prior knowledge of the synthetic network nor network exploration. Participants were randomly divided into two groups. The experiment group had 4 males and 2 females, and the control group had 3 males and 3 females.

Task. The task of participants was to find some structures inside the synthetic network. We provided each participant a sheet, on which a set of structures were presented. Some listed structures appeared in the synthetic data but some do not. Subjects were asked to search for structures similar to each structure on the sheet and also to count the number of each structure they found.

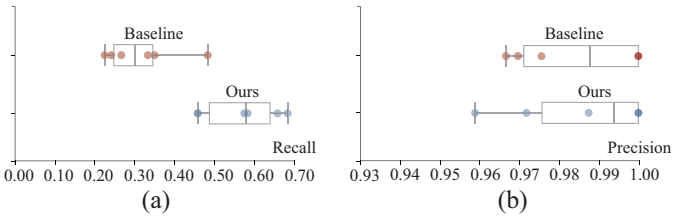


Figure 14. Recall and precision of our approach and the baseline system. (a) Our approach has higher recalls than the baseline system; (b) the precision of ours and the baseline are close.

Procedure and Apparatus. Each trail had three steps. In the first step, a participant was given a 5-minute tutorial with the Bitcoin trading network. The functions of the system they would use were introduced and participants then explored the Bitcoin trading network freely to get familiar with the system. In the second step, users were asked to complete the task with the synthetic network and to record what they found when they explored the network in the system . This step had a fixed time length, 20 minutes, but a participant could stop exploration before the time ran out if they thought they had completed the task. The final step was an interview after the task was completed. In this step, we asked participants for feedback about the system and the task. We recorded the feedback of participants. We used the same PC described in Section 6 for all trials.

Results. We compared the recall and precision between two groups. As Figure 14 shows, both approaches lead to high precision: 0.98 for both treatments and no significant difference found (p=.45). However, there is a significant difference in recall between two treatments, 0.56 for our system vs. 0.32 for the baseline ($p < .001$).

Our interview data showed participants were more positive about our system than the baseline. Participants with the baseline system complained tasks were tediousness and time-consuming. In comparison, participants using our system indicated that the exemplar-based suggestion provides an efficient query interface and speeds up the exploration process dramatically. A participant told us the sketching exemplar view was very helpful because he could directly draw a specific structure for suggestions when he could not find the structures on the answer sheet.

Some participants expressed concerns with our system, however. A participant said that he did not trust the suggestions generated by the system and had to confirm the suggested structure one by one in the node-link diagram. Examining his logs, we found that his performance was still impressive, with a recall higher than most participants in the baseline group. Two participants told us that the two parameters k and ε-neighbors were hard to understand, but they could still find structures with the help of our system because it showed the number of suggested structures every time when they adjusted the parameters.

8 Discussions

Value. Our system can be used as the first step towards exploring a large graph with an exemplar paradigm. Our design maintains a balance between flexibility and compatibility in specifying exemplars. Currently, our system provides two modes to define exemplars. These modes are compatible with existing techniques in graph definition and interaction design. For example, specifying exemplars is compatible with the vectorization framework. Meanwhile, providing structure templates follows a design principle in interaction design: recognizing an object is easier than recalling it.

Quality of suggestions. Our method can provide high-quality suggestions. The quality of suggested exemplars depends on the vectorized representations: GraphWave outperforms other representations in the selection mode and Graphlet Kernel performs better than others in the sketching mode. We believe that the quality of suggestion can be further improved by designing vectorized representations that are more suitable for structure-based exploration. Moreover, we provide two mechanisms to improve the quality of suggested exemplars. First,users are enabled to filter out low-quality exemplars by manually adjusting parameters in the control panel. Second, high-quality suggestions are shown to users by sorting suggestions with their similarities to the specified exemplar. Experimental results also show that the ways of specifying exemplars affect the quality of suggestion because the proper value of parameters varies in different modes.

Advantages. Our approach offers three advantages over traditional graph query methods that are designed for graph database [32, 79]. First, our method can be applied more broadly than graph queries. Database-based graph query methods are usually designed for graphs with attributes. Our approach only requires topological information of a graph, and works well on unlabeled graphs. Second, our method can tolerate minor differences in structures and provide inclusive query results. Database-based query methods typically require an explicit expression of structures and may produce redundant query results. Third, our approach can be used by more diverse user groups. To use database-based query methods, users need to be skilled at formulating explicit query expressions of the target and relationship between nodes. In contrast, our approach allows users to specify or choose an exemplar, even without a clear goal in exploration or domain knowledge on network analysis.

Scalability. Our approach has a reasonable scalability. As discussed in Section 5.7, our algorithm performs excellently in terms of time complexity. The average and max query times increase linearly with the size of networks and exemplars. This suggests a direction to improve graph query performances.

Limitations. The major limitation of our approach is the potential loss of important contextual information in the query algorithm. The vectorized representation of a node contains the information of its neighbors in the vectorization methods. Therefore, the suggestion results tend to have similar context with the exemplar. In most cases, this is less a concern in exploration, but could lead to unexpected results if an exemplar is sketched without well-defined context, in particular when networks are very complex.

Future works. In the future, we first plan to improve the interactions in specifying exemplars, such as enabling users to specify exemplars with abstracted concepts. We plan to extend our method to other network types, such as networks with contextual information or dynamic networks. Currently, the vectorized representations in our method are designed for simple networks. We believe that our approach can be extended to support other more complex networks. The vectorized representations can be modified to support the translation of the context of nodes into other types to vectors. In addition to the user interface can be improved to support other network types (e.g., for dynamic networks, adding a timeline to indicate the evolution of networks and support the sketching of time-varying exemplars).

9 Conclusion

In this paper, we propose a structure-based suggestive exploration approach for large networks. Leveraging vectorized representations of nodes in networks, we develope an exemplar-based structure query algorithm to support graph query based on user-specified exemplar. Our query engine suggests similar structures in a network and can greatly help the exploration of large networks. We also build a visual interactive system to support the suggestive exploration, and the results from our experiments and usability study indicate that our system is easy to use and capable to support efficient exploration of large networks. The results of our research suggest that our approach could be effective for graph query in general networks, as long as their topological structures are clearly defined.

Acknowledgments

This research has been sponsored supported by National Key Research and Development Program (2018YFB0904503), National Natural Science Foundation of China (61772456, 61761136020, U1736109).

REFERENCES

[1] Blockchain. https://blockchain.info/.

[2] J. Abello, S. Hadlak, H. Schumann, and H.-J. Schulz. A modular degree-of-interest specification for the visual analysis of large dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):337–350, 2014.

[3] J. Abello, F. Van Ham, and N. Krishnan. Ask-graphview: A large scale graph visualization system. *IEEE transactions on visualization and computer graphics*, 12(5):669–676, 2006.

[4] B. Bach, N. H. Riche, C. Hurter, K. Marriott, and T. Dwyer. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE transactions on visualization and computer graphics*, 23(1):541–550, 2017.

[5] M. Behrisch, J. Davey, F. Fischer, O. Thonnard, T. Schreck, D. Keim, and J. Kohlhammer. Visual analysis of sets of heterogeneous matrices using projection-based distance functions and semantic zoom. In *Computer Graphics Forum*, vol. 33, pp. 411–420. Wiley Online Library, 2014.

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

[7] S. S. Bhowmick, B. Choi, and S. Zhou. Vogue: Towards a visual interaction-aware graph query processing framework. In *CIDR*, 2013.

[8] K. Börner, C. Chen, and K. W. Boyack. Visualizing knowledge domains. *Annual review of information science and technology*, 37(1):179–255, 2003.

[9] N. Cao, Y.-R. Lin, L. Li, and H. Tong. g-miner: Interactive visual group mining on multivariate graphs. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 279–288. ACM, 2015.

[10] S. K. Card and D. Nation. Degree-of-interest trees: A component of an attention-reactive user interface. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pp. 231–245. ACM, 2002.

[11] G. J. Chang. Algorithmic aspects of domination in graphs. In *Handbook of combinatorial optimization*, pp. 1811–1877. Springer, 1998.

[12] D. H. Chau, L. Akoglu, J. Vreeken, H. Tong, and C. Faloutsos. Tourviz: interactive visualization of connection pathways in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1516–1519, 2012.

[13] 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 *IEEE International Conference on Data Mining Workshops*, pp. 963–966, 2008.

[14] D. H. Chau, A. Kittur, J. I. Hong, and C. Faloutsos. Apolo: making sense of large network data by combining rich user interaction and machine learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 167–176. ACM, 2011.

[15] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, pp. 149–159, 2001.

[16] T. Crnovrsanin, I. Liao, Y. Wu, and K.-L. Ma. Visual recommendations for network navigation. In *Computer Graphics Forum*, vol. 30, pp. 1081–1090. Wiley Online Library, 2011.

[17] X. Ding, J. Jia, J. Li, J. Liu, and H. Jin. Top-k similarity matching in large graphs with attributes. In *International Conference on Database Systems for Advanced Applications*, pp. 156–170. Springer, 2014.

[18] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec. Learning structural node embeddings via diffusion wavelets. *arXiv preprint arXiv:1710.10321*, 2017.

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

[20] N. Elmqvist, T.-N. Do, H. Goodell, N. Henry, and J.-D. Fekete. Zame: Interactive large-scale graph visualization. In *PacificVIS*, pp. 215–222. IEEE, 2008.

[21] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, vol. 96, pp. 226–231, 1996.

[22] S. Ghani, N. H. Riche, and N. Elmqvist. Dynamic insets for context-aware graph navigation. In *Computer Graphics Forum*, vol. 30, pp. 861–870. Wiley Online Library, 2011.

[23] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *INFOVIS*, pp. 17–24. Ieee, 2004.

[24] H. Gibson, J. Faith, and P. Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information visualization*, 12(3-4):324–357, 2013.

[25] S. Gladisch, H. Schumann, and C. Tominski. Navigation recommendations for exploring hierarchical graphs. In *International Symposium on Visual Computing*, pp. 36–47. Springer, 2013.

[26] L. Gou, F. You, J. Guo, L. Wu, and X. L. Zhang. Sfviz: interest-based friends exploration and recommendation in social networks. In *Proceedings of the 2011 Visual Information Communication-International Symposium*, p. 15. ACM.

[27] P. Goyal and E. Ferrara. Graph embedding techniques, applications, and performance: A survey. *arXiv preprint arXiv:1705.02801*, 2017.

[28] J. A. Grochow and M. Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Annual International Conference on Research in Computational Molecular Biology*, pp. 92–106. Springer, 2007.

[29] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864. ACM, 2016.

[30] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *International Symposium on Graph Drawing*, pp. 285–295. Springer, 2004.

[31] S. Hachul and M. Jünger. Large-graph layout algorithms at work: An experimental study. *J. Graph Algorithms Appl.*, 11(2):345–369, 2007.

[32] H. He and A. K. Singh. Graphs-at-a-time: query language and access methods for graph databases. In *SIGMOD 2008*. ACM.

[33] J. Heer and D. Boyd. Vizster: Visualizing online social networks. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pp. 32–39. IEEE, 2005.

[34] N. Henry, J.-D. Fekete, and M. J. McGuffin. Nodetrix: a hybrid visualization of social networks. *IEEE transactions on visualization and computer graphics*, 13(6):1302–1309, 2007.

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

[36] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian. Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PloS one*, 9(6):e98679, 2014.

[37] Y. Jia, J. Hoberock, M. Garland, and J. Hart. On the visualization of social and other scale-free networks. *IEEE transactions on visualization and computer graphics*, 14(6):1285–1292, 2008.

[38] S. Kairam, N. H. Riche, S. Drucker, R. Fernandez, and J. Heer. Refinery: Visual exploration of large, heterogeneous networks through associative browsing. In *Computer Graphics Forum*, vol. 34, pp. 301–310. Wiley Online Library, 2015.

[39] E. Kerzner, A. Lex, C. L. Sigulinsky, T. Urness, B. W. Jones, R. E. Marc, and M. Meyer. Graffinity: Visualizing connectivity in large graphs. In *Computer Graphics Forum*, vol. 36, pp. 251–260. Wiley Online Library, 2017.

[40] O.-H. Kwon, T. Crnovrsanin, and K.-L. Ma. What would a graph look like in this layout? a machine learning approach to large graph visualization. *IEEE transactions on visualization and computer graphics*, 24(1):478–488, 2018.

[41] B. Lee, C. S. Parr, C. Plaisant, B. B. Bederson, V. D. Veksler, W. D. Gray, and C. Kotfila. Treeplus: Interactive exploration of networks with enhanced tree layouts. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1414–1426, 2006.

[42] O. Lenz, F. Keul, S. Bremm, K. Hamacher, and T. von Landesberger. Visual analysis of patterns in multiple amino acid mutation graphs. In *IEEE Conference on Visual Analytics Science and Technology*, pp. 93–102. IEEE, 2014.

[43] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 631–636, 2006.

[44] J. Leskovec and J. J. Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pp. 539–547, 2012.

[45] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu. Analyzing the training processes of deep generative models. *IEEE transactions on visualization and computer graphics*, 24(1):77–87, 2018.

[46] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization*

and computer graphics, 23(1):91–100, 2017.

[47] S. Liu, W. Cui, Y. Wu, and M. Liu. A survey on information visualization: recent advances and challenges. *The Visual Computer*, 30(12):1373–1393, 2014.

[48] K.-L. Ma and C. W. Muelder. Large-scale graph visualization and analytics. *Computer*, 46(7):39–46, 2013.

[49] D. Marcus and Y. Shavitt. Rage–a rapid graphlet enumerator for large networks. *Computer Networks*, 56(2):810–819, 2012.

[50] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga, and J.-D. Fekete. Topology-aware navigation in large networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2319–2328. ACM, 2009.

[51] C. Mueller, B. Martin, and A. Lumsdaine. A comparison of vertex ordering algorithms for large graph visualization. In *Visualization, 2007. APVIS'07. 2007 6th International Asia-Pacific Symposium on*, pp. 141–148. IEEE, 2007.

[52] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

[53] C. Partl, S. Gratzl, M. Streit, A. M. Wassermann, H. Pfister, D. Schmalstieg, and A. Lex. Pathfinder: Visual analysis of paths in graphs. In *Computer Graphics Forum*, vol. 35, pp. 71–80. Wiley Online Library, 2016.

[54] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM, 2014.

[55] R. Pienta, J. Abello, M. Kahng, and D. H. Chau. Scalable graph exploration and visualization: Sensemaking challenges and opportunities. In *International Conference on Big Data and Smart Computing*, pp. 271–278. IEEE, 2015.

[56] R. Pienta, F. Hohman, A. Endert, A. Tamersoy, K. Roundy, C. Gates, S. Navathe, and D. H. Chau. Vigor: Interactive visual exploration of graph query results. *IEEE transactions on visualization and computer graphics*, 24(1):215–225, 2018.

[57] R. Pienta, M. Kahng, Z. Lin, J. Vreeken, P. Talukdar, J. Abello, G. Parameswaran, and D. H. Chau. Facets: Adaptive local exploration of large graphs. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 597–605. SIAM, 2017.

[58] 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*, pp. 272–279. ACM, 2016.

[59] N. Pržulj, D. G. Corneil, and I. Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.

[60] H. C. Purchase, E. Hoggan, and C. Görg. How important is the mental map?–an empirical investigation of a dynamic graph layout algorithm. In *International Symposium on Graph Drawing*, pp. 184–195. Springer, 2006.

[61] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 385–394. ACM, 2017.

[62] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

[63] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pp. 488–495, 2009.

[64] A. Srinivasan and J. Stasko. Orko: Facilitating multimodal interaction for visual exploration and analysis of networks. *IEEE transactions on visualization and computer graphics*, 24(1):511–521, 2018.

[65] T. Tang, S. Rubab, J. Lai, W. Cui, l. Yu, and Y. Wu. istoryline: Effective convergence to hand-drawn storylines, To appear.

[66] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 567–580. ACM, 2008.

[67] C. Tominski. *Event based visualization for user centered visual analysis.* PhD thesis, University of Rostock, 2006.

[68] C. Tominski, J. Abello, F. Van Ham, and H. Schumann. Fisheye tree views and lenses for graph visualization. In *International Conference on Information Visualization*, pp. 17–24. IEEE, 2006.

[69] 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, 2016.

[70] F. Van Ham and A. Perer. search, show context, expand on demand: supporting large graph exploration with degree-of-interest. *IEEE Transactions on Visualization and Computer Graphics*, 15(6), 2009.

[71] T. von Landesberger, S. Bremm, J. Bernard, and T. Schreck. Smart query definition for content-based search in large sets of graphs. *Proceedings of EuroVAST*, pp. 7–12, 2010.

[72] T. von Landesberger, M. Görner, R. Rehner, and T. Schreck. A system for interactive visual analysis of large graphs using motifs in graph editing and aggregation. In *VMV*, vol. 9, pp. 331–340, 2009.

[73] T. von Landesberger, M. Gorner, and T. Schreck. Visual analysis of graphs with multiple connected components. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, pp. 155–162. IEEE, 2009.

[74] T. Von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner. Visual analysis of large graphs: state-of-the-art and future research challenges. In *Computer graphics forum*, vol. 30, pp. 1719–1749. Wiley Online Library, 2011.

[75] Y. Wu, N. Cao, D. Gotz, Y.-P. Tan, and D. A. Keim. A survey on visual analytics of social media data. *IEEE Transactions on Multimedia*, 18(11):2135–2148, 2016.

[76] Y. Wu, Z. Chen, G. Sun, X. Xie, N. Cao, S. Liu, and W. Cui. Streamexplorer: A multi-stage system for visually exploring events in social streams. *IEEE Transactions on Visualization and Computer Graphics*, (1):1–1, 2017.

[77] C. Xie, W. Zhong, W. Xu, and K. Mueller. Visual analytics of heterogeneous data using hypergraph learning. In *ACM Transactions on Intelligent Systems and Technology*. ACM, 2018.

[78] J. Zhao, C. Collins, F. Chevalier, and R. Balakrishnan. Interactive exploration of implicit and explicit relations in faceted datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2080–2089, 2013.

[79] P. Zhao and J. Han. On graph query optimization in large networks. *Proceedings of the VLDB Endowment*, 3(1-2):340–351, 2010.

[80] H. Zhou, P. Xu, X. Yuan, and H. Qu. Edge bundling in information visualization. *Tsinghua Science and Technology*, 18(2):145–156, 2013.