User-guided Graph Exploration: A Framework for Algorithmic Complexity Reduction in Large Data Sets

Tim Grube*, Florian Volk†, Max Mühlhäuser*, Suhas Bhairav‡, Vinay Sachidananda‡, Yuval Elovici‡
*Telecooperation Group, TU Darmstadt, Germany
†House of IT, Germany
‡iTrust, SUTD, Singapore
Email: {grube, max}@tk.tu-darmstadt.de, volk@house-of-it.eu, {suhas_setikere, sachidananda, yuval_elovici}@sutd.edu.sg

***Abstract*—Human exploration of large data sets becomes increasingly difficult with growing amounts of data. For this purpose, such data sets are often visualized as large graphs, depicting information and interrelations as interconnected vertices. A visual representation of such large graphs (for example, social networks, collaboration analyses or biological data sets) has to find a trade-off between showing details in a magnified—or zoomed-in—view and the overall graph structure. Showing these two aspects at the same time results in a visual overload that is largely inaccessible to human users. In this article, we augment previous work and present a new approach to address this overload by combining and extending graph-theoretic properties with community detection algorithms. Our non-destructive approach to reducing visual complexity while retaining core properties of the given graph is user-guided and semi-automated. The results yielded by applying our approach to large real-world network data sets reveal a massive reduction of displayed vertices and connections while keeping essential graph structures intact.**

***Keywords*—Complexity reduction; graph visualisation; big data exploration; graph metrics; community detection.**

I. Introduction

Processing data sets is becoming increasingly easier. With the rise of *big data* and powerful computing devices, the collection and processing of large data sets has become a common thing in both research and industry. This article extends beyond our previous work on reducing visual complexity in graphs [1] with the introducing of our framework and an extended evaluation.

Many fields profit from the capability to reveal new insights and connections that can only be detected by analyzing large amounts of collected data.

However, Moore's Law [2] does not apply to the ability of human users to understand and explore such big data sets. Making large data sets accessible to human users is difficult and becomes increasingly more difficult with ever-growing data sets. Human-centric data analysis techniques usually employ visualization of the data sets. Visualizing real-world data sets as graphs quickly results in an inaccessible chaos of large amounts of interconnected vertices due to the large amount of information to be shown. See Figure 1 for a visualization of a part of the Facebook social network [3]. In order to comprehend relations within such a graph, a user needs to magnify the visualization a lot. This magnification implies a loss of overview, so that a user might be able to understand specific relations within a part of the graph, but loses track of the overall structure.

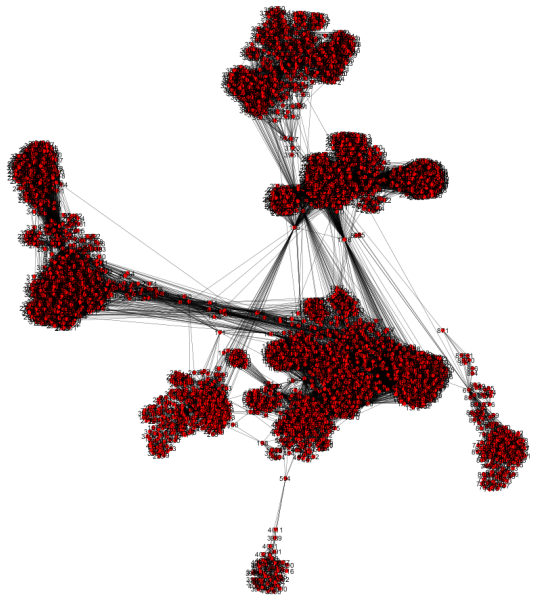

Fig. 1: User relations in a social network (data taken from Facebook).

This issue of simultaneous visualization of both details and overall structure complicates the process of exploring data sets when the intended outcome is unknown. This is a common scenario for exploratory data analysis. As such explorative scenarios often call for human creativity as the aim is to find new insights, connections and cross-references that are not yet algorithmically formalized. Pre-defined and well-specified analyses can be processed automatically without human involvement, and thus, without the need for visual representation. However, exploratory data analysis usually does not allow for automated pre-filtering of data sets to obtain a human-centric, *decluttered* view on the data and the *information* to be discovered.

The framework and methodology we present in this paper combine and extend graph-theoretic properties with community detection. The assumption is that concealing less important vertices (and their connections) from a graph leads to a compressed and easier-to-understand visual representation

that still contains the information intended to be discovered. The framework is designed to assist a user-guided iterative process of data reduction. For every iteration step, the user first selects a vertex as template, thereby defining a *selection criteria value* (SCV). All other vertices are filtered depending on their SCV in comparison with the selected SCV. Vertices of lower importance are concealed while ones with equal or higher importance are retained. This step can be combined with community detection, which reduces clusters of similar vertices to a single vertex, retaining the structure of the graph, but not the data density. This semi-automated, non-destructive approach to reduce visual complexity is assumed to identify core insights to be yielded from the data while side-information is hidden.

The proposed framework and methodology aim at improving human-lead analysis and understanding of huge data sets; efficiency and scalability aspects in automated processing of such data sets are not in the focus of this research.

Our results indicate a massive reduction of displayed vertices and links between them in every iteration of our proposed approach. Therefore, only one to three iterations effectively reduce graphs as the one shown in Figure 1 to a representation that is easy to comprehend for human users (see Figure 3 for the results of just three iterations).

The article is structured as follows: in Section II, we discuss relevant related work on complexity reduction in graphs; in Section III, we discuss fundamental methodological background information to our contribution by introducing the graph-theoretic metrics and community detection mechanisms incorporated in our framework. For every method, we discuss why and how we assume it is beneficial to our goal of removing unnecessary data. Section IV, the general approach to apply community detection and SCVs to iteratively filter vertices is presented. Our framework itself is presented in Section V, discussing both the internal structure and the visual user interface. Our assumptions and framework are experimentally evaluated in Section VI, in which we describe the simulation setup and discuss the obtained results in detail for various data sets. We conclude with a summary and future research directions in Section VII.

II. State of the Art

In this section, we discuss related work with respect to the field of complexity reduction in graphs. Most of these contributions are related to the field of human computer interaction (HCI) and aim on making complex datasets accessible to humans.

Kimelman et al. [4] proposed techniques like ghosting, hiding and grouping of edges. The vertices and edges of the graph were removed based on various techniques like weights of edges, labels of vertices, etc., but these techniques were concerned with dynamic graphs. Differently, Holten et al. [5] reduced only the visual cluttering of edges by bundling them.

Fisheye techniques [6] [7] tend to concentrate only on the interesting regions of the user. The zooming feature in such techniques is only responsible for making a very small region of a graph appear larger. They do not remove any vertices or edges. So, the overall graph content remains the same. Fisheye views that retain structure are introduced by Furnas [8]. Abello et al. [9] introduced hierarchical clustering and depiction of a tree-map in addition to a compound fisheye view technique but never concentrated on reducing the overall size of a network.

Various approaches towards creating communities in large graphs are presented in [10] [11] [12]. These techniques provide a significant level of understanding of the kind of vertices and their properties in large networks but never used the same to reduce the overall content in a large network and provide a simpler view.

Sundararajan et al. [13] introduced rectangular partitioning and Voronoi partitioning techniques. The former involved partitioning the area of display into four quadrants while the latter involved the partition area being closer to the concerned vertex. This only reduces the distortion in the graphs.

Batagelj et al. [14] took a mathematical approach through the usage of matrix. Large graphs were reduced to *k-cores*. Later, the graphs are represented as an adjacency matrix or a contextual matrix based on their size. But when the graphs grows really large, managing the matrix becomes a enormous task.

All before mentioned techniques reduce complexity based on a global perspective, disregarding the current interests of a user. Thus, these techniques may maintain and highlight information that is not relevant the user's current situation while rejecting important information from the user's slant.

III. Methods

In this section we introduce our terminology and the various graph-theoretic metrics we apply to measure the importance of vertices within a graph. The reason hereby is to distinguish important vertices that are relevant for the human observer from those vertices that can be removed from the visualization. Less important vertices are considered to be non-essential for the visualization and, as such, can be removed from the visualization without altering the general information the graph explorer is looking for.

A. Terminology: Graph-based Data Representation

Most data sets can be represented by a Graph $G = (V, E)$ that is formed by a set of *vertices* $v_i \in V$, which represents the pieces of data. Relations and *connections* between these pieces of data are represented by *edges* $E \subseteq V \times V$, i.e., connections are represented by pairs of vertices (v_x, v_y). Considering our exemplary application scenario of a social network, each vertex represents a user and edges represent the connections between users. Another example is the model of (research) citations as citation graph where publications are represented as vertices, and a citation is represented by a directed edge.

A graph G can be *directed* or *undirected*. In a directed graph, an edge e_k can exist between a vertex v_i and v_j while the other direction (e_l: v_j to v_i) may or may not exist, independent of the existence of e_k. In an undirected graph, the existence of before-mentioned edge e_k implies the existence of e_l. A social network like Facebook applies undirected connections, thus, if user Alice is connected to user Bob, Bob is also connected to Alice (Alice and Bob are "friends"); in contrast to that, Twitter applies directed connections. Thus, Alice may be connected to Bob (Alice "follows" Bob), but Bob may not be connected to Alice.

The number of connections of a vertex v_i is noted as the *degree* d_{v_i} of v_i. In case of a directed graph, the degree of a vertex has to be specified for outgoing connections: the *out-degree* and for incoming connections: the *in-degree*.

A connection between any two vertices is called a path p. A path between adjacent vertices has the length 1; yet, a path may include intermediate vertices to connect them. A path p is represented by an sequence of edges $p = (e_0, e_1, ..., e_n)$. The *shortest path* between two vertices is the path with least edges.

B. Selection Criteria Values

Vertex selection is one of the fundamental steps in reducing the complexity of a graph, i.e., it is the selection of an appropriate subgraph. Selecting a set of vertices based on specific parameters helps in creating a subgraph that retains the inherent properties of the graph and reflects the user's interests. The vertices are identified based on *selection criteria values*. An SCV acts as the basis of vertex selection in our framework. For a vertex to be selected as part of a graph, its SCV must be greater than the SCV of the user's interest. Here, we express the user's interest by selecting a (start) vertex whose SCV is compared with every other vertex in the graph.

The SCVs include graph-theoretic properties and centrality measures but are not limited to just these presented measures. We focus this paper on the usage of importance, connectivity, and distance measures as these appeared most promising in our literature research. The importance measures are PageRank and Betweenness Centrality; the connectivity measures are Clustering Coefficient and Degree Centrality; the distance measure is Closeness Centrality. In the following sections, we discuss every SCV to understand their significance to our idea of reducing visual complexity.

1) Closeness Centrality: Closeness centrality determines the closeness of vertices in a network, i.e., it determines the distance of vertices in a graph. The vertices that have high closeness centrality values are considered to be closer to other vertices in the graph. For example, in a network where information flows from one vertex to another, transmission of information takes place quickly due to their high closeness centrality values.

The rationale for using closeness centrality is as follows: a set of vertices that are close to each other may form a center of information within a graph. Close neighbors are thereby thought to attribute to the general information that a human user is looking for. Vertices that are far away from this information center only contain auxiliary information and can be omitted during graph exploration. Closeness is generally attributed to the quickness in flow of information in the network. The quicker the information arrives at or departs from a vertex, the closer is the destination or the source vertex respectively [15].

According to Freeman [16], closeness centrality of a vertex is defined as:

"The sum of graph-theoretic distances from all other vertices, where the distance from a vertex to another is defined as the length of the shortest path from one to the other."

For example, in a social network scenario, this could mean when an information is shared between two individuals, the chances of such information propagating in the network is higher with those who are immediate friends or neighbors. For a given vertex p_i, closeness centrality [17] is calculated using (1), where $|V|$ denotes the total number of vertices in the graph, i and k are integers where $i \neq k$, $d(p_i, p_K)$ denotes the number of edges in the shortest path between p_i and p_k.

$$C_C(p_i) = \frac{|V| - 1}{\sum_{k=1}^{V} d(p_i, p_k)} \tag{1}$$

2) Betweenness Centrality: Betweenness centrality depicts influence or powerfulness of a vertex in a network. The vertices that have high betweenness centrality values are *highly influential* as they act as bridges or shortcuts for interactions between several pairs of vertices or even information centers. Vertices with low betweenness centrality are considered to be less influential than others and can be removed from a graph without changing the structure of its information flow.

According to Freeman [16], the betweenness centrality of a vertex is defined as:

"The share of times that a vertex i needs a vertex k (whose centrality is being measured) in order to reach a vertex j via the shortest path."

For example, in a social network scenario, where individuals form friendships with individuals who in turn have a large group of friends tend to have an influential clout due to their connections with such individuals.

Betweenness centrality [17] can be calculated using (2), where g_{jk} denotes the total number of shortest paths between p_i and p_k, and $g_{jk}(p_i)$ denotes the number of shortest paths containing p_i.

$$C_B(p_i) = \sum_{i \neq j \neq k \in V} \frac{g_{jk}(p_i)}{g_{jk}} \tag{2}$$

3) PageRank: PageRank [18] provides a rank to every vertex in a graph. It extends the idea of computing citations to a web page by ensuring that all rank values are normalized based on the total number of links on a page instead of treating all web pages equally.

A transition matrix is created based on the transfer of importance from one to another. Initially, we apply a uniform distribution based on the initial grade structure. Then, depending upon the incoming connections we re-calculate the PageRank value [19].

A high number of incoming links is thought to indicate high importance of the respective vertex. Transitive importance is given by PageRank as those who have incoming links from vertices with a high PageRank inherit this importance. As such, vertices with low PageRank values are not closely connected to those vertices, which contain the core information in a graph. By only keeping the important vertices within a graph, this core information is retained even if all other vertices are hidden from the human user.

The most prominent example for PageRank application is the world wide web network: a web page that is referenced in many other web pages will have a high PageRank value. In a social network scenario, individuals with higher PageRank are generally highly powerful due to a lot of connections whose betweenness centrality value is also high. The concept of PageRank is closely related to the betweenness centrality as it measures the importance by the number of vertices that are (transitively) pointing towards another one; the transitive propagation of importance is, however, unique to the PageRank.

4) Degree Centrality: Degree centrality portrays the level of connectivity of a vertex in a network. The vertices that have high degree centrality values are more influential or important. In difference to PageRank, degree centrality is not inherited from connected vertices, but only measures a degree of connectedness. Anyhow, similar to PageRank, highly connected vertices are considered to be of higher importance to retaining the information in a graph than vertices with only few connections. The latter ones are hidden from the user by our framework.

The degree centrality is computed as presented in (3) and given by the number of adjacent neighbors.

$$C_D(p_i) = deg(p_i) \tag{3}$$

For example, in a social network, a vertex with a high number of connections is powerful and highly visible when compared to others [11].

5) Clustering Coefficient: As stated before for the centrality metrics, we assume the existence of information centers within graphs. Our framework is tailored towards identifying and retaining these information centers while removing as many less important vertices as possible from the graph.

The clustering coefficient gives an indication of clusters in a network. The higher the clustering coefficient, the closer a vertex is to a cluster, i.e., vertices with a high clustering coefficient are parts of clusters. Vertices with a low clustering coefficient are considered outliers with respect to information centers and, therefore, can be removed from the graph without a change to the information a user is looking for.

According to Zafarani et al. [20], the clustering coefficient can be defined as shown in (4).

$$CC = \frac{3 \times (Number\ of\ triangles)}{Number\ of\ Connected\ Triples\ of\ Vertices} \tag{4}$$

For example, in a social network, vertices with high clustering coefficients have a desire to form close bonds or friendships with their neighbors [21].

C. Community Detection Algorithms

Community detection allows to group vertices that share similar properties. The properties vary depending on the type of the used *community detection algorithm* (CDA). For example, Louvain algorithm optimizes communities with respect to maximizing their modularity while direct K-means community detection optimizes on the spread of communities.

We use the found communities to reduce complexity in larger steps, being able to remove larger portions of the graph in the first few complexity reduction cycles. CDAs form one of the core evaluation criteria in our implementation. The four algorithms used in our framework are Hierarchical Clustering, Original Louvain, Louvain with Multilevel Refinement, and Direct Clustering. In the following sections, we will discuss the CDAs that help us in understanding the way in which various clusters are formed.

1) Hierarchical Clustering: The hierarchical clustering method by Jain and Dubes [22] involves obtaining a series of partitions that are nested by transforming a proximity matrix. The proximity matrix represents the distance of the individual vertices.

Based on this matrix, vertices are merged in an iterative process. First, the two closest, trivial clusters of one vertex are merged; after that, the next two closest clusters are merged. As the algorithm is applied, a hierarchy of clusters is established. Those clusters are usually visualized using a dendrogram, as shown in Figure 2.

We utilize the complete-link clustering method in an agglomerative (bottom up) setting in our implementation. One

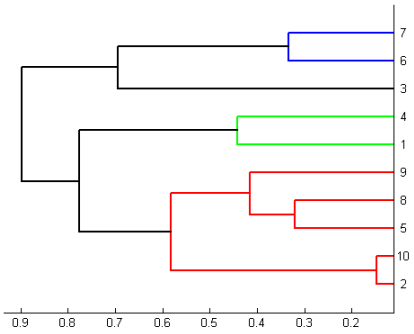

Fig. 2: Hierarchical Agglomerative Clustering visualized.

of the main advantages of this method is the avoidance of chaining-effects. Also, the clusters are balanced and smaller when applying this method.

2) Original Louvain: The original Louvain algorithm by Blondel et al. [23] detects communities in two steps that are repeated:

1) establishing communities by locally grouping vertices such that the graph *modularity* [24] is maximized. For the modularity maximization, vertices are moved between communities [25].
2) replacing the communities with single vertices to reduce the complexity of the graph.

These steps are repeated until no improvement of modularity is achievable and a hierarchy of communities is established.

3) Louvain with Multilevel Refinement: Louvain with Multilevel Refinement algorithm was formulated and conceived by Rotta et al. [26]. The algorithm is similar to the original Louvain community detection, but the adds a *refinement* phase at the end to the community detection algorithm. In this refinement phase, the achieved clustering hierarchy of the original Louvain method is visited in reverse order, i.e., from the coarsest communities to the trivial communities.

This multilevel approach provides communities exhibiting better modularity values at the cost of higher computation effort.

4) Direct K-means: The direct K-means algorithm by Alsabti et al. [27] establishes k communities. For that, the following steps are repeated until the algorithm converges to its solution:

1) k vertices are selected at random and form the centers of the k communities.
2) the remaining vertices are assigned to their closest community.
3) for each community, it is evaluated, which vertex is the actual center of the community by computing the mean position.
4) the vertices that are closest to the mean position of their respective community are selected as the center of their community.
5) repeat from step 2)

IV. Complexity Reduction Approach

In this section, we present our iterative and human-centric approach to reduce the data complexity in graphs using SCVs and CDAs.

A. Subjectivity of Visual Complexity

Visual complexity is a subjective measure. Every user has a different understanding of visual complexity of a data set, depending on previous experience, expert knowledge, and familiarity with the respective subject. Hence, the challenge of visual complexity reduction cannot be solved with a "one for all" approach.

To overcome this subjectivity challenge, we provide a user-interactive round-based complexity reduction method that allows to reduce the visual complexity in succeeding iterations until the user is able to understand the data set well enough to collect the desired information.

B. Start Vertex Selection

Selecting a start vertex is highly dependent on the analysis goal of the user. Thus, prior knowledge beyond the scope of this paper is required. However, this knowledge is available to the user. That is why the proposed approach employs an interactive start vertex selection, which (i) suits the analysis goal as well as (ii) solves the issue of aquiring appropriate prior knowledge. The interactive start vertex selection may be implemented using a graphical repesentation of the overall graph, in which a human user simply points and clicks on the desired start vertex. Otherwise, explicitely naming or describing the vertex by a set of properties or by a name is possible as well.

The evaluation in section VI is non-interactive. Therefore, a random start vertex is selected in every iteration. As the evaluation is to analyse the effectiveness of visual complexity reduction, selecting a "random" analysis goal seems reasonable.

C. Visual Complexity Reduction

Our method to reduce the visual complexity is a two-step process, which also allows to parallelize the computation to reduce computation time. We give an algorithmic description in Algorithm 1. In the first step, we apply a community detection algorithm to group similar information. According to lines 3-6 of Algorithm 1, we compute the disjoint communities and proceed with each of them individually. At this point, we

```
1: function ReduceVisualComplexity(g, r, s, id, type_g, c_alg)
   g: Input graph
   r: Number of iterations
   s: Selection criteria
   id: Start vertex (picked by user or at random)
   type_g: Graph Type (directed or undirected)
   c_alg: Community detection algorithm
   g_reduced: Output graph with reduced complexity
2:     g_reduced ← new Graph();
3:     for i ← 1 to r do
                                   ▷ loc: List of Communities
4:         loc ← getCommunities(id, g, algo, g_type)
5:         for j ← 1 to sizeOf(loc) do
6:             community ← loc.get(j)
7:             for k ← 1 to sizeOf(community) do
8:                 v_k ← community[k]
9:                 if getSCV(v_k, s) > getSCV(id, s) then
                                       ▷ compute shortest path
10:                    sp ← getSP(v_k, id)
11:                    addToReducedGraph(sp, g_reduced)
                   ▷ break if applied w/ community detection
12:                if loc > 0 then
13:                    break loop
14:                end if
15:            end if
16:        end for
17:    end for
18:    i++
19:    end for
20:    return g_reduced
21: end function
```

Algorithm 1: Iterative Reduction of Visual Complexity.

can easily parallelize the computation, having the communities being handled concurrently.

In the second step, we have to differentiate whether community detection was applied in the first place or not. If community detection is not applied, we calculate the SCV of each vertex and compare it with the SCV of the vertex representing the user's interest. When the SCV of the inspected vertex is higher than a threshold, which is in our case the SCV of the user's interest, the vertex is retained in the reduced graph. To preserve the relation and connection of the user's interest and inspected vertex, we retain not only the vertex but also the vertices on the shortest path. In our social network scenario, these vertices represent the chain of friends between two subjects and, thus, may be relevant to the user.

If community detection was applied in the first place, we only retain representatives of each community and hide the remaining vertices of a community. This way, we can easily achieve a massive complexity reduction in only a few iterations. Yet, the user can select the representative of a community as next interest, restarting the process from this new perspective and gain new insights into the data set.

These two steps are then repeated until the user is able to sufficiently comprehend the graph. Our algorithmic description uses a predefined number of iterations, yet an application would delegate this decision to the user, who might request further iterations.

In Figure 3, we visualize the iterative complexity reduction process. On the left-hand side, the original graph, which fed into the visual complexity reduction mechanism, is visualized. In three steps, visualized towards the right-hand side, we reduce the visual complexity of the data set considering a randomly chosen vertex to reflect an otherwise user-selected interest. The user interest is marked by the green vertex with the ID 12 in Figure 3. The original data set, labeled "Start", is the Facebook data set already presented in the introduction in Figure 1.

D. Optional Application of CDAs

The application of CDAs is optional in our framework and approach. If the user assumes the presence of "distinct" communities or clusters, the application of CDAs helps to preserve this structure.

Without application of CDAs, the SCVs focus on absolute values that they compute and, therefore, they select the absolut top vertices according to their graph metric. In combination with CDAs, the representative vertices that are retained in each iteration are selected from the different communities. Thus, a higher diversity is achieved.

However, without inherent community structure in the graph, the application of CDAs may be a) misleading and b) a waste of computing resources.

V. Framework

In this section, we describe design decisions and components that are considered while developing the complexity reduction framework. The framework is released to the research community on a public GitHub repository [28].

A. Internal Structure

Our framework for complexity reduction is based on the graph analysis framework *GraphStream* [29], a framework by researchers of the University of Le Havre and members of the RI2C research team. GraphStream is extended by the CDAs and the SCVs and a graphical user interface to enable human users to explore complex data sets visually.

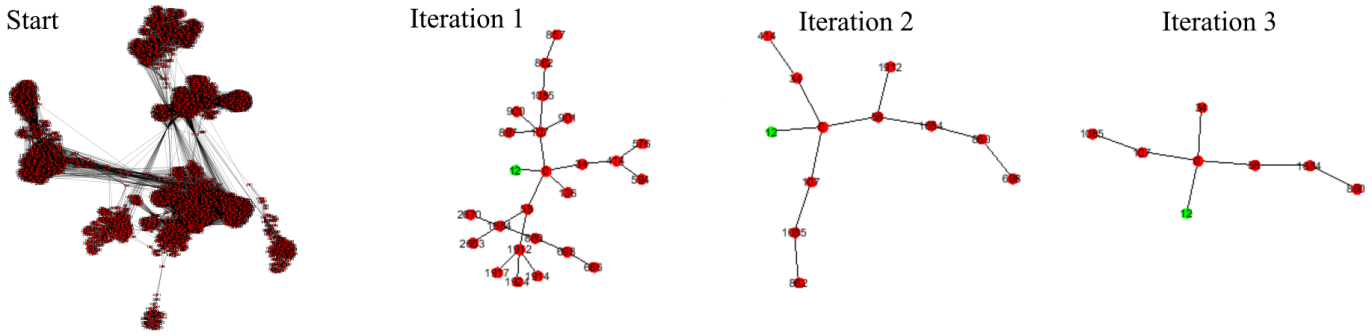

Fig. 3: Complexity Reduction of Graph in Fig. 1 by applying Algorithm 1 (first three iterations shown).

1) GraphStream: GraphStream is a Java-based library for modeling and analyzing dynamic graphs. For that, GraphStream provides a graph model that enables vertices and connections between them to appear and to disappear over time. Static data sets and graphs can be modeled by ignoring the dynamic capabilities of GraphStream.

GraphStream provides capabilities to visualize—and therefore, to layout—graph-based data sets and to calculate graph-theoretic measures to analyze and understand graph-based data sets; moreover, by using the dynamic capabilities of GraphStream, we are able to alter the data sets to reduce their visual complexity.

a) Data Handling: Users can load their specific data set into the application. Today, the dataset should be structured according to the `dgs`-format of GraphStream; in this format, a graph and its dynamic changes are described according to time steps (or clock ticks) and a number of changes (events) within in these intervals—a static graph is described by skipping the time step declaration such that every change (event) is performed at the beginning. The format is as follows:

```
DGS004 .........................◁ file format & version
<name> <#time steps> <#events>
st 0 ...............................◁ time step w/ id 0
an n1 ..................◁ add vertex n1 w/o parameter
an n2 x=1, y=3  ◁ add vertex n2 w/ parameters x, y
cn n2 x=2, y=2 ......◁ change parameters x, y of n2
dn n2 ...............................◁ delete vertex n2
ae n1 n2 ..............◁ add connection between n1 n2
ae n1 > n2 ...........◁ add connection from n1 to n2
ae n1 < n2 ...........◁ add connection from n2 to n1
ce e1 w=10 ....◁ change parameter of connection e1
de e1 ............................◁ delete connection e1
```

Providing a continuous flow of these events, for example, using a network interface, dynamic changes of the data base can visualized.

To enable the subsequent evaluation in this article, a reader for simple adjacency lists (i.e., a list where each entry is a tuple "n1 n2" representing a connection between vertices n1 and n2) is implemented.

More readers for different file formats can be added easily, which is a necessity when considering the usage of our framework in vastly different application scenarios. For that, the user has to provide a file parser that generates the vertices and connections between vertices according to their very own specifications and semantic.

b) Layout and Visualization: The standard layout algorithm in GraphStream is force-based, i.e., the layout algorithm in GraphStream applies repelling forces to each vertex and contracting forces to each connection. These forces ensure that vertices are placed with distance to each other and, yet, connected vertices are grouped.

Force-based visualization already supports human users in understanding large and complex data sets by arranging the graph according to its connection-structure and its clusters, for example, the initial graph in Figure 3 reveals seven groups of vertices. By this grouping, the distance between groups and the placement and concentration of connections, a human user is able to get first insights by taking his application knowledge into account.

2) Control Flow: The internal control flow in the framework is depicted in Figure 4. First, the user loads a data set into the framework. Using the graphical user interface, the user can then configure the complexity reduction by choosing a selection criteria value and an (optional) community detection algorithm. With the selection of their interest, the user passes the control to the complexity reduction module, which is composed of the community detection (using the CDAs) and computation of the SCVs.

In the complexity reduction module, the optional communities are established and the representatives for each community are computed; after that, the vertices are removed accordingly. This step is then repeated until either of the following termination conditions is fulfilled:

- *Iterations done:* after n iterations, the complexity reduction step is completed and may be restarted by the user.
- *No change:* if two subsequent graph representations are equal, i.e., computing communities and selection criteria values does not enable the removal of additional vertices, the complexity reduction is completed. By selecting anot-

her vertex as interest, the user may restart the complexity reduction.

After completing the complexity reduction, the control is passed back to the user interface where the reduced data set is visualized to the user. From here, the user can refine their interest and select another start vertex for another complexity reduction.

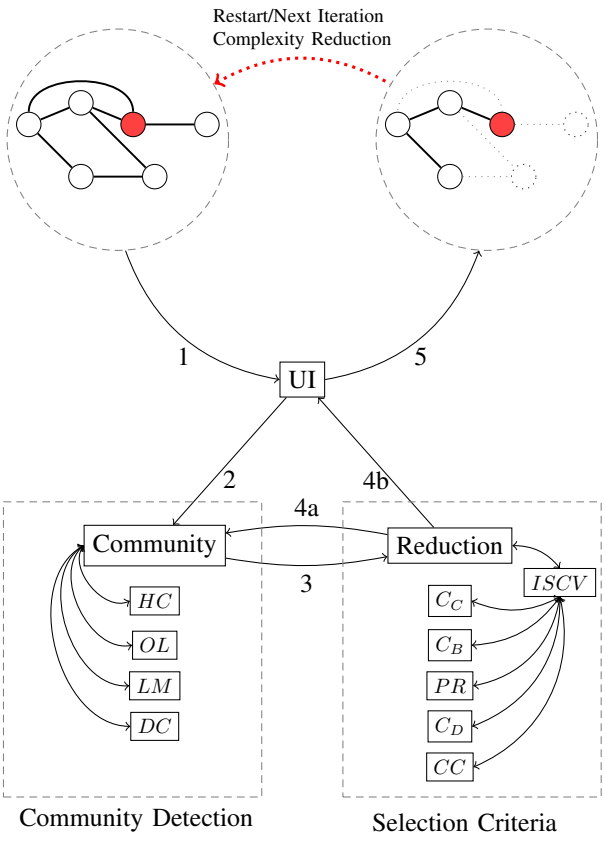

Fig. 4: Control Flow.

3) CDAs and SCVs as Modules: The CDAs and SCVs are the core components of our framework. By their ranking of vertices, CDAs and SCVs do also define, which properties of vertices are distinguishing the vertices to preserve from the ones to remove.

The actual properties defining the importance of vertices are highly depending on the application scenario and use case, i.e., the type of information the user is looking for. For example:

> *"Reducing complexity in a social network when searching 'influential people' will look for dense communities and the highly connected people in these communities."*

To account for this variability of required properties, the implementation of CDAs and SCVs follows a modular structure that allows easy extension by researchers or users of the framework.

The CDAs are encapsulated in their own class, the actual implementation of the community detection algorithms is based on the `S-Space` [30] repository of Jurgens and Stevens from UCLA.

The SCVs are hidden behind the interface `ISelectionCriteria` to provide their functionality without enforcing a stronger coupling of different modules with the core framework.

B. User Interface

The focus of the user interface is the visualization of the graph. From this perspective, the user is able to explore their data according to their application scenario. Figure 5 visualizes the framework from a user's perspective.

From the initial view, the user has the ability to load a data set provided in the aforementioned `dgs`-format—or to generate a synthetic network, for example a *Barabàsi-Albert* network.

After visualizing the initial "root" graph, the user can select the SCV fitting their requirements as well as the number of iterations, and whether a community detection algorithm is to be applied or not.

The reduced graphs of all iterations are visualized in a "Graph Grid" tab such that the user is made aware of the different stages of complexity reduction and the respective implications.

VI. Evaluation

In a simulation study, we evaluated the performance and effectiveness of our proposed technique to reduce the visual complexity of data sets. In the next section, we provide insights into our experimental setup and details of our simulation study. After that, we present and discuss our results for different network classes.

A. Experimental Setup

In this section, we describe and discuss our selection of data sets, followed by the description of our simulation setup.

a) Networks: To provide useful insights, we decided to perform our simulation based on samples of real-world networks. We imagine the usage of our technique in applications based on social networks (with respect to to social ties), in (scientific) libraries (with respect to to co-authorship), biological data (with respect to protein interactions), and computer networks (with respect to to the logical interconnections) as described in Table I. We restrict our simulation to the largest connected component of the data set, as accounting for importance and relationship of non-related information is out of scope of this paper.

As the representative for a social network, we decided to use a sample of Facebook's social graph [3]. This data set (FB)

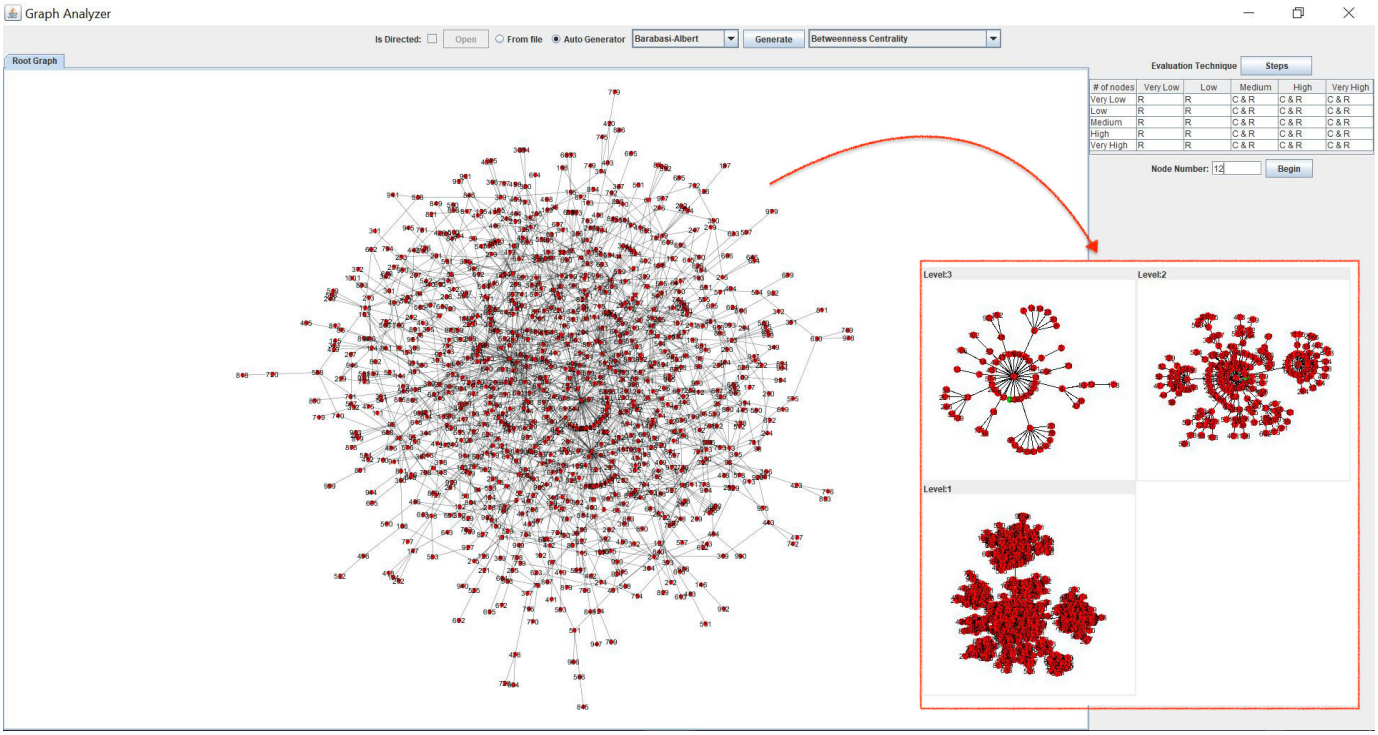

Fig. 5: Framework after loading the initial Dataset. Complexity reduction mechanism can be configured visually.

TABLE I: Large Network data sets

Class	data set	Vertices	Connections
Social	Facebook (FB)	4,029	88,234
Library	Arxiv gr-qc (CA)	4,158	13,428
Biological	Vidal (VI)	3,133	6,726
	Moreno (MO)	1,870	2,277
Computer	p2p-Gnutella08 (P8)	6,301	20,777
	p2p-Gnutella09 (P9)	8,114	26,013

consists of 4,029 vertices and 88,234 connections, representing users of Facebook and their respective interconnections.

As the representatives for library-based and collaboration-based data sets, we selected a co-authorship data set [3] that reflects the research collaborations and co-authorships in the Arxiv gr-qc (General Relativity and Quantum Cosmology) area. This data set (CA) consists of 4,158 vertices and 13,428 connections in their largest connected component.

As the representatives of biological networks, we selected the protein interaction networks vidal [31] and moreno [32]. These data sets reflect interactions of proteins on a molecular level. The vidal data set (VI) consists of 3,133 vertices and 6,726 connections; the moreno data set (MO) consists of 1,870 vertices and 2,277 connections.

As the representative of computer networks, we selected a sample of the Gnutella network [3] that reflects the interconnections of users in the Gnutella network. The sample (P8) is sampled on August 8th, 2002 (p2p-Gnutella08) and consists of 6,301 vertices and 20,777 connections. The second sample (P9) is sampled one day later on August 09th, 2002 (p2p-Gnutella09) and consists of 8,114 vertices and 26,013 connections.

b) Simulation Setup: We perform a reduction of visual complexity on each of the aforementioned data sets and calculate graph-theoretic properties that are also used as SCVs, namely closeness centrality (C_C), betweenness centrality (C_B), PageRank (PR), degree centrality (C_D), and clustering coefficient (CC).

We perform the complexity reduction on each of the data sets with each pairwise combination of selection critera values (SCV) and community detection algorithms (CDAs) been described in Sections III-B and III-C and compare and interpret the results represented by the statistical mean values and number of removed vertices.

We need a user interest expressed as one "preselected" piece of information from the data set to apply our technique to reduce visual complexity. To account for this, we selected a random vertex as user interest and repeat this whole process 15-times.

In Table II, we summarize the details of our simulation study and used abbreviations.

B. Results

In this section, we present the results of our simulative study. The results are structured according to the class of networks as aforementioned. Detailed measurements are shown in Table III.

TABLE II: Simulation Details

Property	Value(s)
SCV	Closeness Centrality (C_C) Betweenness Centrality (C_B) PageRank (PR) Degree Centrality (C_D) Clustering Coefficient (CC)
CDA	Hierarchical Clustering (HC) Direct Clustering (DC) Original Louvain (OL) Louvain w/ multilevel refinement (LM)
Repetitions	15
Iterations	3
User Interest	V.getRandomVertex()

a) Social Networks: Social networks, here the FB data set, exhibit a strongly connected core, the so-called *rich club*. These vertices are connectors between various clusters. This structural behavior leads to comparable short paths, i.e., the closeness centrality is comparably high, and core vertices exhibit a high betweenness and degree centrality. Moreover, due to the "clustering" of vertices, the clustering coefficient is also high (in the FB data set: 0.6055).

Our complexity reduction technique reduces up to 4,035 (99.9%) vertices in the three iterations. All combinations but one of the SCV and CDA combinations preserve the core of the data set, which is indicated by the high C_B, C_C, and C_D values. These indicate that the retained information is representative for their clusters. The drop of PR and CC indicate that the preserved structure roughly follows a star-topology, which is also shown in Figure 3.

Using CC as only SCV leads to a different result where different information is retained, yet, this is expected. The core vertices have by definition a lower CC value and are, thus, not likely to be above the threshold. These vertices are connected to a multitude of different other vertices that are forming their own, smaller clusters.

b) Library/Collaboration Data Set: A collaboration, or library, data set yields a similar structure to a social network. Collaborators establish clusters and an inner core of highly influential people and publications. The gr-qc data set reveals a high CC (0.5296) and a strongly connected core. While the diameter of the data set suggests otherwise, the core of the data set still shows a high connectivity and tendency towards short paths. That is evident when comparing the diameter (17) and the diameter when only 90% of vertices have to be connected (0.9-percentile effective diameter: 7.6).

Our complexity reduction technique removes up to $4,196$ vertices in the three performed iterations. We can see that most combinations of SCV and CDA perform similarly, producing star-topology-like results retaining relevant information of the core of the data set as indicated by higher C_B and C_D. The drop of the CC supports the star-topology again.

However, using CC or C_D as SCV produces different results. Using one of these properties as SCV retains information revealing the significantly higher CC, C_D, and PR and lower C_D values. Thus, retaining representatives of smaller but better and tighter connected clusters.

c) Biological Data Set: Protein interaction networks reveal a different structure, the spread between average and maximum degree is between factor 23 and 30; this factor is by a magnitude smaller than in the previously discussed social networks. This drop in degree goes hand in hand with increased diameter (13–19) and a massive decline in the CC (0.035–0.055 compared to 0.6055 on the Facebook data set).

Our complexity reduction technique removes up to 3,125 (99.74%) vertices on the vidal protein interaction network and up to 1,860 (99.47%) vertices on the moreno protein interaction network. Our technique performs similarly on both data sets, and produces star topologies regardless of the used combination of SCV and CDA. Thus, the resulting C_B is comparably high, and the C_C is very low < 0.0001. The CC also drops to 0.01–0.0004 and endorses the star topology as the remaining vertices are hardly forming local clusters. The PR is similarly low. Yet, the CO-DC combination retains local clusters, which is indicated by higher C_C and CC values.

d) Computer Networks: Computer networks, in this case the samples from the Gnutella network, are resembling either AS-networks with social-like structures on a large scale or, if consisting of only "a few" computers (compared to the whole population), random networks. As the Gnutella network were comprised of only a small subset of all Internet users, the Gnutella sample is akin to a random network. Yet, the degree distribution reveals on the base data set exponentially distributed degrees. Nonetheless, the PageRanks reveal the lack of a highly connected inner core; this core is essential to a "social" network.

Our complexity reduction techniques removed up to 6,299 (99.97%) vertices on the P8 data set and up to 8,132 (99.85%) vertices on the P9 data set. The consistently high C_B shows that some well-connected vertices are retained, but the retained vertices are distant from each other, thus, resulting in very low C_C values; only PR used as SCV preserves closer vertices. Using C_C, CC, or PR as SCV results in retaining local clusters in the results while the other SCVs do not keep clusters in their reduced data sets – this is visible by the high CC and PR values for these SCVs.

C. Destructive vs. Non-Destructive Complexity Reduction

The complexity reduction can be performed in two variations: a destructive and a non-destructive one.

First, the execution can be *destructive* by removing vertices that are not matching the user's interest, i.e., removing vertices when their SCV is lower than the threshold and if they are not selected to be a representative. During execution, the data set will shrink and, thus, free memory and reduce the computational complexity of the calculation of SCVs with each performed complexity reduction iteration. However, the user cannot revert the reduction process to earlier stages of complexity reduction.

TABLE III: Properties of Complexity-reduced Data Sets of Facebook, Collaboration, Gnutella, and the Protein Interaction Networks.

	Technique	Mean Values				
		C_B	C_C	CC	C_D	PR
FB	C_C & DC	2211000	0.3457	0.00038	298.1	0.00215
	C_C & HC	2356000	0.3478	0.00033	311.9	0.00230
	C_C & OL	1925000	0.3425	0.00037	260.3	0.00191
	C_C & LM	1925000	0.3425	0.00037	260.3	0.00191
	C_B & DC	1453000	0.3327	0.00148	205.9	0.00163
	C_B & HC	1685000	0.3182	0.00604	219.5	0.00209
	C_B & OL	2112000	0.3397	0.00291	251	0.00197
	C_B & LM	2253000	0.3389	0.00356	262.5	0.0021
	PR & DC	1520000	0.3308	0.00268	203.9	0.0017
	PR & HC	1754000	0.3356	0.00208	223.5	0.00178
	PR & OL	1588000	0.3332	0.00284	206.2	0.00163
	PR & LM	1809000	0.3339	0.00336	232.6	0.00185
	C_D & DC	1539000	0.3304	0.0024	219.7	0.00175
	C_D & HC	1810000	0.3291	0.00492	236.4	0.002
	C_D & OL	1943000	0.3311	0.0063	236.2	0.00194
	C_D & LM	2100000	0.3356	0.00451	262.3	0.00211
	CC & DC	70010	0.2638	0.07401	14.13	0.00013
	CC & HC	79330	0.2639	0.08139	15.11	0.00014
	CC & OL	30171	0.27314	2.42765	36.62	2.0282
	CC & LM	83490	0.2641	0.08111	15.71	0.00015
CA	C_C & DC	73470	0.00005	0.02978	16.36	0.00041
	C_C & HC	71590	0.00005	0.04217	8.94	0.00031
	C_C & OL	74020	0.00005	0.02445	17.66	0.00044
	C_C & LM	73470	0.00005	0.02978	16.36	0.00041
	C_B & DC	233900	0.00005	0.00592	16.95	0.00058
	C_B & HC	224500	0.00005	0.00579	16.47	0.00057
	C_B & OL	158000	0.00005	0.01006	14.12	0.00049
	C_B & LM	151600	0.00005	0.00884	13.93	0.00047
	PR & DC	222000	0.00005	0.0015	20.85	0.00067
	PR & HC	98330	0.00005	0.00811	14.92	0.00045
	PR & OL	215300	0.00005	0.00445	19.91	0.00063
	PR & LM	226500	0.00005	0.00266	20.64	0.00066
	C_D & DC	28418	4.44	1.57434	8.38	0.88412
	C_D & HC	28261	4.44	1.57331	8.37	0.88485
	C_D & OL	28370	4.44	1.57288	8.37	0.88744
	C_D & LM	97370	0.00004	0.00991	12.61	0.00039
	CC & DC	22449	4.17	0.84129	4.83	2.43779
	CC & HC	22449	4.17	0.84129	4.83	2.43779
	CC & OL	21060	4.14	0.80191	4.63	2.58936
	CC & LM	43790	0.00003	0.1903	7	0.0003
P8	C_C & DC	38492	0.00023	1.42208	9.55	3.59775
	C_C & HC	39680	0.00023	0.00003	9.53	0
	C_C & OL	38492	0.00023	1.42208	9.55	3.59775
	C_C & LM	38666	0.00023	1.43795	9.54	3.59132
	C_B & DC	82970	0.00023	0	9.92	0
	C_B & HC	88810	0.00023	0.00003	9.72	0
	C_B & OL	162900	0.00023	0	9.84	0
	C_B & LM	137000	0.00023	0	9.83	0
	PR & DC	20920	0.61077	0.58976	3.77	4.02269
	PR & HC	20920	0.61077	0.58976	3.77	4.02269
	PR & OL	18941	0.61688	0.57661	3.69	4.09170
	PR & LM	21934	0.60621	0.59507	3.81	4.14545
	C_D & DC	119200	0.00023	0	12.68	0
	C_D & HC	122900	0.00023	0	12.55	0
	C_D & OL	47420	0.00023	0.00002	10.18	0
	C_D & LM	47420	0.00023	0.00002	10.18	0
	CC & DC	46280	0.00393	2.44391	9.69	3.76677
	CC & HC	46304	0.00393	2.44341	9.69	3.76735
	CC & OL	46280	0.00393	2.44391	9.69	3.76677
	CC & LM	46576	0.00388	2.43269	9.69	3.75996

	Technique	Mean Values				
		C_B	C_C	CC	C_D	PR
VI	C_C & DC	57420	0.00009	0.00196	15.28	0.00107
	C_C & HC	57420	0.00009	0.00196	15.28	0.00107
	C_C & OL	58990	0.00009	0.00154	15.73	0.0011
	C_C & LM	54320	0.00009	0.00232	14.61	0.00102
	C_B & DC	61750	0.00009	0.00045	15.6	0.0011
	C_B & HC	58410	0.00009	0.00044	15.06	0.00106
	C_B & OL	57790	0.00009	0.00046	14.98	0.00106
	C_B & LM	56350	0.00009	0.00046	14.78	0.00104
	PR & DC	62850	0.00009	0.00600	15.86	0.00112
	PR & HC	138900	0.00009	0.00133	29.01	0.00203
	PR & OL	49980	0.00009	0.00686	13.65	0.00097
	PR & LM	85260	0.00006	0.01049	18.23	0.00136
	C_D & DC	31410	0.00009	0.02806	9.053	0.00064
	C_D & HC	156200	0.00009	0.00314	31.63	0.00221
	C_D & OL	83280	0.00007	0.04043	18.82	0.00136
	C_D & LM	111300	0.00007	0.04704	24.04	0.00172
	CC & DC	23410	8.61693	1.67212	7.98	0.00318
	CC & HC	30480	0.00008	0.1194	8.65	0.00063
	CC & OL	23590	0.00009	0.07014	7.94	0.00057
	CC & LM	24450	0.00009	0.08001	8.03	0.00058
MO	C_C & DC	33615	0.00005	0.00681	6.31	0.00115
	C_C & HC	31886	0.00005	0.00596	6.16	0.00114
	C_C & OL	33467	0.00005	0.00668	6.33	0.00116
	C_C & LM	42355	0.00005	0.0124	7.32	0.00132
	C_B & DC	22275	0.00004	0.00152	4.99	0.00096
	C_B & HC	21245	0.00004	0.00153	4.91	0.00094
	C_B & OL	62989	0.00004	0.00306	9.87	0.00178
	C_B & LM	61556	0.00004	0.00312	9.7	0.00175
	PR & DC	33375	0.00004	0.00595	6.67	0.00125
	PR & HC	30472	0.00004	0.00604	6.34	0.0012
	PR & OL	32441	0.00004	0.0062	6.51	0.00123
	PR & LM	24884	0.00004	0.00618	5.70	0.00109
	C_D & DC	25683	0.00004	0.00277	5.82	0.00109
	C_D & HC	23793	0.00004	0.0028	5.64	0.00106
	C_D & OL	24978	0.00004	0.00284	5.80	0.00108
	C_D & LM	61071	0.00004	0.00376	10.53	0.00189
	CC & DC	20061	0.00004	0.0867	5.28	0.00099
	CC & HC	20061	0.00004	0.0867	5.28	0.00099
	CC & OL	20503	0.00004	0.0857	5.33	0.001
	CC & LM	21185	0.00004	0.0845	5.39	0.00101
P9	C_C & DC	79094	0.00034	0.00133	10.14	0
	C_C & HC	79094	0.00034	0.00133	10.14	0
	C_C & OL	79094	0.00034	0.00133	10.14	0
	C_C & LM	79094	0.00034	0.00133	10.14	0
	C_B & DC	78569	0.00034	0.00001	9.73	0
	C_B & HC	115689	0.00034	0	9.93	0
	C_B & OL	249905	0.00034	0.00002	10.92	0
	C_B & LM	267735	0.00034	0.00003	11.08	0
	PR & DC	28406	0.00014	0.0001	3.69	0
	PR & HC	25826	0.00013	0.0001	3.60	0
	PR & OL	50534	0.00017	0	4.68	0
	PR & LM	46586	0.00016	0	4.60	0
	C_D & DC	145848	0.00034	0	13.17	0
	C_D & HC	145848	0.00034	0	13.17	0
	C_D & OL	163656	0.00034	0.00001	13.18	0
	C_D & LM	134312	0.00034	0	13.16	0
	CC & DC	68933	0.00034	0.00188	9.96	0
	CC & HC	68933	0.00034	0.00188	9.96	0
	CC & OL	69376	0.00034	0.00179	9.94	0
	CC & LM	71280	0.00034	0.00209	9.98	0

Second, the execution can be *non-destructive* by, e.g., concealing "removed" vertices only without deleting them from the actual graph. This variation allows moving back and forth between different stages of complexity reduction. Moreover, this also allows a user to adapt her interest, i.e., she can use gained knowledge and re-run the visual complexity reduction with a different focus/interest.

The main functionality of our proposed approach to visual complexity reduction remains unaltered by this design decision as it only affects the ability to move through the different stages of complexity reduction and the visualization for the human user.

Our proposed technique to reduce the visual complexity is capable of removing the vast majority of information in just a few iterations as we have shown before. In order to overcome unwated loss of information during the process of data reduction, our approach always allows the user to track back and select different vertices as user interest. This, however, is

infeasible for dynamic data sets, which can be adressed, for example, by caching results from earlier iterations.

VII. Conclusion and Future Work

The collection and processing of large data sets got common with the rise of *big data* and powerful computing devices. Human users are hardly able to keep up with the increasing pace of collecting and accruing data. Accessing and—more important—exploring as well as understanding these data sets becomes difficult.

In this article, we present our framework that combines an interactive, user-guided approach to the exploration of large data sets with automated complexity reduction based on graph-theoretic properties and community detection. This approach reduces visual complexity in order to render visualizations of data sets more usable for human users. The reduction of visual complexity is achieved by removing parts of the data sets in a semi-automated fashion: based on a user-selected information, i.e., a selected vertex becomes a template for a high *selection criteria value* (SCV), other vertexes with a lower SCV as well as their connections are hidden from the data set. Thereby, the complexity is reduced, which we assume to render the visualization easier to comprehend. The computation of SCVs is based on graph-theoretic properties like closeness and the detection of communities within the data set.

The presented framework combines this automated computation and removal with interactive selection of a new template vertex before each complexity reduction iteration.

Our simulation study has shown that our combination of graph-theoretic properties, measuring the importance of data in the data set, and community detection, grouping similar data in the data set, is able to reduce the visual cluttering of information efficiently. If performed in a non-destructive setting, i.e., if discarded data is only concealed and not removed from the data set, human users can shift their focus when inspecting a data set to account for new insights gained by the visual inspection of data sets. The complexity reduction can then be performed again with a focus on new, shifted interests.

Our proposed technique opens an additional direction of research to support user-centric systems in rising exposure to big data and accruing amounts of data. Future research may include user studies to select the—per application field—appropriate graph-theoretic properties in order to achieve an appropriate visual complexity reduction.

Acknowledgements

This work has been co-funded by the DFG as part of project B.2 within the RTG 2050 Privacy and Trust for Mobile Users and by the German Federal Ministry for Education and Research (BMBF, Software Campus project KomBi, grant no. 01IS12054).

References

[1] T. Grube, F. Volk, M. Mühlhäuser, S. Bhairav, V. Sachidananda, and Y. Elovici, "Complexity Reduction in Graphs: A user Centric Approach to Graph Exploration," in *International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services (CENTRIC)*. IARIA, 2017.

[2] G. E. Moore, "Readings in computer architecture," M. D. Hill, N. P. Jouppi, and G. S. Sohi, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, ch. Cramming More Components Onto Integrated Circuits, pp. 56–59.

[3] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[4] D. Kimelman, B. Leban, T. Roth, and D. Zernik, "Reduction of visual complexity in dynamic graphs," in *International Symposium on Graph Drawing*. Springer, 1994, pp. 218–225.

[5] D. Holten and J. J. Van Wijk, "Force-directed edge bundling for graph visualization," in *Computer graphics forum*, vol. 28, no. 3. Wiley Online Library, 2009, pp. 983–990.

[6] M. Sarkar and M. H. Brown, "Graphical fisheye views of graphs," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1992, pp. 83–91.

[7] E. R. Gansner, Y. Koren, and S. C. North, "Topological fisheye views for visualizing large graphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 4, pp. 457–468, 2005.

[8] G. W. Furnas, "Generalized fisheye views," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '86. New York, NY, USA: ACM, 1986, pp. 16–23.

[9] J. Abello, S. G. Kobourov, and R. Yusufov, "Visualizing large graphs with compound-fisheye views and treemaps," in *International Symposium on Graph Drawing*. Springer, 2004, pp. 431–441.

[10] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, p. 066111, Dec 2004.

[11] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca, "Network analysis in the social sciences," *science*, vol. 323, no. 5916, pp. 892–895, 2009.

[12] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[13] P. Sundararajan, O. J. Mengshoel, and T. Selker, "Multi-fisheye for interactive visualization of large graphs," in *Proceedings of the 17th AAAI Conference on Scalable Integration of Analytics and Visualization*, ser. AAAIWS'11-17. AAAI Press, 2011, pp. 46–50.

[14] V. Batagelj, A. Mrvar, and M. Zaveršnik, "Partitioning approach to visualization of large graphs," in *International Symposium on Graph Drawing*. Springer, 1999, pp. 90–97.

[15] S. P. Borgatti, "Centrality and network flow," *Social networks*, vol. 27, no. 1, pp. 55–71, 2005.

[16] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1978.

[17] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2007, pp. 32–40.

[18] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual web search engine," *Computer networks*, vol. 56, no. 18, pp. 3825–3833, 2012.

[19] "Pagerank implementation description," http://www.ccs.northeastern.edu/home/daikeshi/notes/PageRank.pdf, accessed: 2018-05-28.

[20] R. Zafarani, M. A. Abbasi, and H. Liu, *Social media mining: an introduction*. Cambridge University Press, 2014.

[21] M. Brautbar and M. Kearns, "A clustering coefficient network formation game," in *Intl. Symp. on Algorithmic Game Theory*. Springer, 2011, pp. 224–235.

[22] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[23] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[24] M. E. J. Newman, "Modularity and community structure in networks." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 23, pp. 8577–8582, 2006.

[25] L. Waltman and N. J. van Eck, "A smart local moving algorithm for large-scale modularity-based community detection," *The European Physical Journal B*, vol. 86, no. 11, pp. 1–14, 2013.

[26] R. Rotta and A. Noack, "Multilevel local search algorithms for modularity clustering," *Jrnl. of Exper. Algorithmics*, vol. 16, pp. 2–3, 2011.

[27] K. Alsabti, S. Ranka, and V. Singh, "An efficient k-means clustering algorithm," 1997.

[28] T. Grube and S. Bhairav, "Graph exploration and complexity reduction framework," https://github.com/timgrube/GraphExplorationComplexityReduction, accessed: 2018-05-28.

[29] "Graphstream project," http://graphstream-project.org/, accessed: 2018-05-28.

[30] D. Jurgens and K. Stevens, "S-space," https://github.com/fozziethebeat/S-Space, accessed: 2018-05-28.

[31] "Human protein (vidal) network dataset – KONECT," Oct. 2016, accessed: 2018-05-28. [Online]. Available: http://konect.uni-koblenz.de/networks/maayan-vidal

[32] "Protein network dataset – KONECT," Oct. 2016, accessed: 2018-05-28. [Online]. Available: http://konect.uni-koblenz.de/networks/moreno_propro