Soc. Netw. Anal. Min. (2016) 6:22
DOI 10.1007/s13278-016-0332-2

ORIGINAL ARTICLE

Structure-preserving sparsification methods for social networks

Michael Hamann[1] · Gerd Lindner[1] · Henning Meyerhenke[1] · Christian L. Staudt[1] · Dorothea Wagner[1]

Received: 18 December 2015 / Revised: 4 March 2016 / Accepted: 9 April 2016 / Published online: 29 April 2016

Abstract Sparsification reduces the size of networks while preserving structural and statistical properties of interest. Various sparsifying algorithms have been proposed in different contexts. We contribute the first systematic conceptual and experimental comparison of *edge sparsification* methods on a diverse set of network properties. It is shown that they can be understood as methods for rating edges by importance and then filtering globally or locally by these scores. We show that applying a local filtering technique improves the preservation of all kinds of properties. In addition, we propose a new sparsification method (*Local Degree*) which preserves edges leading to local hub nodes. All methods are evaluated on a set of social networks from Facebook, Google+, Twitter and LiveJournal with respect to network properties including diameter, connected components, community structure, multiple node centrality measures and the behavior of epidemic simulations. To assess the preservation of the community structure, we also include experiments on synthetically generated networks with ground truth communities. Experiments with our implementations of the sparsification methods (included in the open-source network analysis tool suite NetworKit) show that many network properties can be preserved down to about 20 % of the original set of edges for sparse graphs with a reasonable density. The experimental results allow us to differentiate the behavior of different methods and show which method is suitable with respect to which property. While our Local Degree method is best for preserving connectivity and short distances, other newly introduced local variants are best for preserving the community structure.

Keywords Complex networks · Sparsification · Backbones · Network reduction · Edge sampling

1 Introduction

1.1 Context

Complex networks have nontrivial structures and statistical properties and are often represented by graphs. Such data models have been employed in countless domains based on the observation that the structure of relationships yields insights into the composition and behavior of complex systems (Costa et al. 2011). Many concepts were pioneered in the study of social networks, in which edges represent social ties between social actors. Most real-world complex networks, including social networks, are already sparse in the sense that for n nodes the edge count m is asymptotically in $O(n)$. Nonetheless, typical densities lead to a computationally challenging number of edges. Here, we pursue the goal of further sparsifying such networks by retaining just a fraction of edges (sometimes called a "backbone" of the network), while showing experimentally that important properties of networks such as the ranking of the nodes in terms of various measures of centrality, the average clustering coefficient and the diameter can be preserved in the process.

Potential applications of network sparsification are numerous. One of them is information visualization: Even

Parts of this paper have been published in preliminary form in the study by Lindner et al. (2015).

✉ Michael Hamann
michael.hamann@kit.edu

[1] Karlsruhe Institute of Technology (KIT), Institute of Theoretical Informatics, Am Fasanengarten 5, 76131 Karlsruhe, Germany

moderately sized networks turn into "hairballs" when drawn with standard techniques, as the amount of edges is visually overwhelming. In contrast, showing only a fraction of edges can reveal network structures to the human eye if these edges are selected appropriately. Sparsification can also be applied as an acceleration technique; by disregarding a large fraction of edges that are unimportant for the task, running times of graph and network analysis algorithms can be reduced. Many other possible applications arise if we think of sparsification as lossy compression. Large networks can be strongly reduced in size if we are only interested in certain structural aspects that are preserved by the sparsification method. From a network science perspective, sparsification can yield valuable insights into the importance of relationships and the participating nodes: given that a sparsification method tends to preserve a certain property, the method can be used to rank or classify edges, discriminating between essential and redundant edges.

The core idea of the research presented here is that not all edges are equally important with respect to properties of a network: for example, a relatively small fraction of long-range edges typically act as shortcuts and are responsible for the small-world phenomenon in complex networks. The importance of edges can be quantified, leading to *edge scores*, often also referred to as *edge centrality values*. In general, we subsume under these terms any measure that quantifies the importance of an edge depending on its position within the network structure. Sparsification can then be broken down into the stages of (i) edge scoring and (ii) filtering the edges using a global score threshold.

Despite the similar terminology, our work is only weakly related to a line of research in theoretical computer science where *graph sparsification* is understood as the reduction of a dense graph ($\Theta(n^2)$ edges) to a sparse ($O(n)$ edges) or nearly sparse graph while provably preserving properties such as spectral properties (e.g., Batson et al. 2013). The networks of our interest are already sparse in this sense. With the goal of reducing network data size while keeping important properties, our research is related to a body of work that considers sampling from networks (on which Ahmed et al. 2014 provides an extensive overview). Sampling is concerned with the design of algorithms that select edges and/or nodes from a network. Here, node and edge sampling methods must be distinguished: for node sampling, nodes *and* edges from the original network are discarded, while edge sampling preserves all nodes and reduces the number of edges only. The literature on node sampling is extensive, while pure edge sampling and filtering techniques have not been considered as often. A seminal paper (Leskovec and Faloutsos 2006) concludes that node sampling techniques are preferable, but considers few edge sampling techniques. The study presented in Ebbes et al. (2008) looks at how well a sample of 5–20 % of the original network preserves certain properties, and is mainly focused on node sampling through graph exploration. It concludes that random walk-based node sampling works best on complex networks, but does so on the basis of experiments on synthetic graphs only and compares only with very simple edge sampling methods.

Only edge sampling techniques are directly comparable to our edge scoring and filtering methods. In this work, we restrict ourselves to reducing the edge set, while keeping all nodes of the original graph. Preserving the nodes allows us to infer properties of each node of the original graph. This is important because in network analysis, the unit of analysis is often the individual node, e.g., when a score for each user in an online social network scenario shall be computed. With respect to the goal of accelerating the analysis, many relevant graph algorithms scale with m, so reducing m is more relevant.

Another related approach is the *Multiscale Backbone* (Serrano et al. 2009), which is applicable on weighted graphs only and is, therefore, not included in our study. Instead of applying a global edge weight cutoff for edge filtering, which hides important structures at different scales, this approach aims at preserving them at all scales.

A very recent study with a design similar to ours (John and Safro 2016) focuses in detail on algebraic distance for edge rating (see Sec. 3.6).

1.2 Contribution

We contribute the first systematic conceptual and experimental comparison of existing and novel edge scoring and filtering methods on a diverse set of network properties. Descriptions and literature references for the related methods which we reimplemented are given in Sect. 2, for some of them we include descriptions of how we parallelized them. In Sect. 2, we also introduce our Local Degree sparsification method and Edge Forest Fire, an adaption of the existing node sparsification technique to edges. Furthermore, we propose a local filtering step that has been introduced by Satuluri et al. (2011) for one specific sparsification technique as a generally applicable and beneficial post-processing step for preserving the connectivity of the network and most properties we consider.

Our results illuminate which methods are suitable with respect to which properties of a network. Additionally, we take a look at emergent properties by simulating epidemic spreading on sparsified networks in comparison with the original network. We show that our Local Degree method is best for preserving connectivity and short distances

which results in a good preservation of the diameter of the network, some centrality measures and the behavior of epidemic spreading. Depending on the network, our Local Degree method can also preserve clustering coefficients. Considering the preservation of the community structure, we show that some of the newly introduced variants with local filtering are best for preserving the community structure while the variants without local filtering do not preserve the community structure in our experiments.

Furthermore, we have published efficient parallelized implementations and a framework for such methods as part of the *NetworKit* open-source tool suite (Staudt et al. 2014). While our study covers various approaches from the literature, it is by no means exhaustive due to the vast amount of potential sparsification techniques. With future methods in mind, we hope to contribute a framework for their implementation and evaluation.

2 Edge sparsification

All edge sparsification methods we consider can be split up into two stages: (i) the calculation of a score for each of the edges in the input graph (where the score is high if the edge is important) and (ii) subsequent filtering according to such an edge score. In this section, we will first define this framework of scoring and filtering edges more formally. Using this framework we introduce Local Filtering as an optionally applicable step. We conclude this section by addressing some limits of edge sparsification.

Formally, we define an edge (centrality) score as follows:

Definition 1 (*Edge centrality scores*) Given a simple, undirected graph $G = (V, E)$, an edge (centrality) score is a function $c_G : E \to \mathcal{A}$ which assigns to each edge $e = \{u, v\} \in E$ an attribute value $x \in \mathcal{A}$ of (at least) ordinal scale of measurement. The assigned value depends on the position of the edge within the network G as defined by a set of edges $E' \subseteq E$. In the following, we call such a value an edge score, write $c(e)$ where the graph is implied from the context, and assume that $\mathcal{A} = \mathbb{R}^+$.

2.1 Global and local filtering

The simplest way to filter by such an edge score is to apply a global threshold and keep all edges whose score is equal to or above the threshold. For the comparison of the different sparsification methods we need to be able to filter the network such that all methods keep an equal percentage of the edges of the network. As the methods produce scores with different ranges of values and different distributions of these values we cannot simply define a threshold that is the same for all methods. Also using a different threshold per method does not solve the problem as it is possible that many edges have the same score and there is, therefore, no threshold that leads to the desired ratio of kept edges. Therefore, we define global filtering not to apply a given threshold but to filter the edges such that only a given ratio of remains:

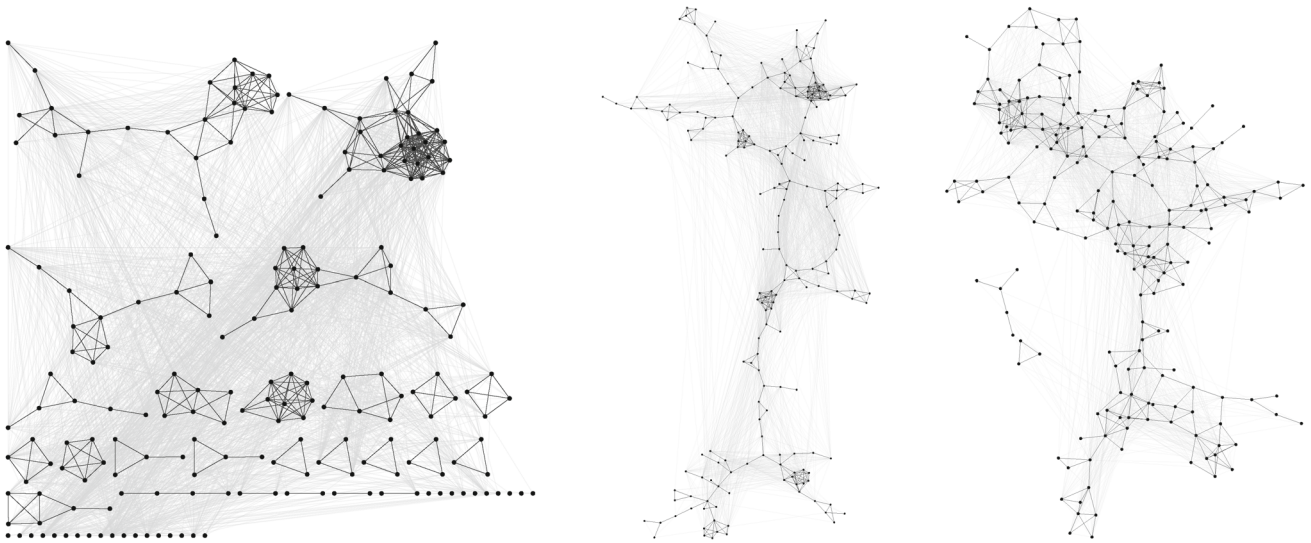
Definition 2 (*Global filtering*) Given a graph $G = (V, E)$ and an edge score $c : E \to \mathbb{R}^+$, a global filtering step by ratio $r \in [0, 1]$ reduces the edges in the sparsified graph $G' = (V, E')$ to $\lfloor r \cdot |E| \rfloor$ edges with the highest values of $c(e)$, i.e., $E' \subseteq E$, $|E'| = \lfloor r \cdot |E| \rfloor$ and $c(e') \geq c(e)$ $\forall\, e' \in E'\ \forall\, e \in E \backslash E'$.

For an actual implementation of global filtering, it is enough to sort all edges by the edge score in descending order and keep the first $\lfloor r \cdot |E| \rfloor$ edges. To make sure that in the case of edges with equal score a given order in the graph does not influence our results, we sort edges that have an equal score in a random order using a random edge score as tie breaker in comparisons.

A problem with global filtering as described above is that methods that are based on local measures like the number of quadrangles an edge is part of tend to assign different scores in different parts of the network as, e.g., some parts of the network are much denser than other parts. Sparsification techniques like (Quadrilateral) Simmelian Backbones (Nocaj et al. 2014) use different kinds of normalizations of quadrangles (see below for details) to compensate for such differences. Unfortunately, these normalizations still do not fully compensate for these differences. In Fig. 1a, we visualize the Jazz network (Gleiser and Danon 2003) with 15 % kept edges as an example. As one can see in the figure, many nodes are isolated or split into small components, the original structure of the network (shown with gray edges) is not preserved.

Simmelian Backbones have been introduced for visualizing networks that are otherwise hard to layout. For layouts it is important to keep the connectivity of the network as otherwise nodes cannot be positioned relative to their neighbors. To preserve the connectivity, Nocaj et al. (2014) keep the union of all maximum spanning trees (UMST) in addition to the original edges. In Fig. 1b, we show the result when we keep the UMST. While the network is obviously connected, much of the local structure is lost in the areas between the dense parts—which is not surprising as we only added the union of some trees.

Satuluri et al. (2011) face a similar problem as with their sparsification technique based on Jaccard Similarity (see below for details) they want to preserve the community structure. They propose a different solution: each



(a) Quadrilateral Simmelian Backbone **(b)** Quadrilateral Simmelian Backbone with UMST **(c)** Quadrilateral Simmelian Backbone with local filtering

Fig. 1 Drawing of the Jazz musicians collaboration network according to a variant of the Quadrilateral Simmelian Backbone with 15 % of the edges (*in black*)

node u keeps the top $\lfloor d(u)^\alpha \rfloor$ edges incident to u, ranked according to their similarity where $d(u)$ is the degree of u and $\alpha \in [0,1]$. An edge is kept when it is ranked high enough by one of the nodes that are incident to it. This procedure ensures that at least one incident edge of each node is retained and thus prevents completely isolated nodes.

To define this filtering step formally, we first introduce a formal definition of the rank of node v in the neighborhood of node u:

Definition 3 (*Neighborhood rank*) Given a graph $G = (V,E)$ and an edge score $c : E \to \mathbb{R}^+$, the neighborhood rank $r_c(u,v)$ is the position of v in the sorted list of neighbors of u, i.e., $r_c(u,v) := |\{x \in N(u) : c(\{u,x\}) < c(\{u,v\})\}| + 1$.

Note that when two edges that are incident to u have the same edge score, they also have the same rank. Again, we want to be able to keep an exact ratio of edges. To achieve this, we transform this local rank into a score that can be used for global filtering.

Definition 4 (*Local filtering score*) Given a graph $G = (V,E)$ and an edge score $c : E \to \mathbb{R}^+$, the directed local filtering score $l_c : V \times V \to [0,1]$

$$l_c(u,v) := \begin{cases} 1, & \text{if } d(u) = 1 \\ 1 - \log(r_c(u,v))/\log(d(u)), & \text{otherwise} \end{cases}$$

Then $l_c(\{u,v\}) := \max(l(u,v), l(v,u))$ is the local filtering score that belongs to c.

Lemma 1 *Filtering locally according to an edge score c with parameter α is equivalent to keeping all edges with $l_c(\{u,v\}) \geq 1 - \alpha$.*

Proof An edge $\{u,v\}$ is kept by local filtering iff $r_c(u,v) \leq d(u)^\alpha$ or $r_c(v,u) \leq d(v)^\alpha$, as the top $d(u)^\alpha$ (or $d(v)^\alpha$) edges are kept. If $d(u) = 1$, the only edge that is incident to u is always kept as $1^\alpha = 1$. For $d(u) > 1$, it holds that

$$\begin{aligned} & r_c(u,v) \leq d(u)^\alpha \\ \Leftrightarrow\ & \log(r_c(u,v)) \leq \log(d(u)) \cdot \alpha \\ \Leftrightarrow\ & \frac{\log(r_c(u,v))}{\log(d(u))} \leq \alpha \\ \Leftrightarrow\ & \underbrace{1 - \frac{\log(r_c(u,v))}{\log(d(u))}}_{=l_c(u,v)} \geq 1 - \alpha \end{aligned}$$

which shows the claim, as $l_c(\{u,v\}$ is exactly defined as the maximum of $l_c(u,v)$ and $l_c(v,u)$, which means that global filtering by $l_c(\{u,v\})$ will keep the edge iff one of the directions would have been kept by local filtering. □

In Algorithm 1, we show the parallel algorithm that we use to transform edge scores for local filtering. We iterate over all nodes in parallel, sort the neighbors, determine the rank for each neighbor and assign its score. As the two scores of an edge are possibly calculated at the same time, we use an atomic maximum for assigning the final score. The total time complexity is $O(m \cdot \log(d_{\max}))$, where $d_{\max}$ denotes the maximum degree in G, as every list of neighbors needs to be sorted.

Input: Graph $G = (V, E)$, edge score $s : E \to \mathbb{R}$
Output: Edge score $l : E \to [0, 1]$

```
l(u, v) ← 0 ∀{u, v} ∈ E;
foreach u ∈ V do in parallel
    r ← 0; // rank
    s ← 1; // number of equal scores in a row
    o ← −∞; // old score
    foreach v ∈ N(u) sorted by s(u, v) in descending order do
        if s(u, v) ≠ o then
            r ← r + s;
            s ← 1;
        else
            s ← s + 1;
        if d(u) > 1 then
            e ← 1 − log(r)/ log(d(u));
        else
            e ← 1;
        l(u, v) ← atomic max of l(u, v) and e;
```

Algorithm 1: Transformation of global edge scores into edge scores that are equivalent to local filtering

In Fig. 1c, we show the Jazz network, sparsified again to 15 % of the edges with the Quadrilateral Simmelian Backbone method using local filtering. With the local filtering step, the network is almost fully connected and local structures are maintained, too. Additionally, as already Satuluri et al. observed when they applied local filtering to their Jaccard Similarity, the edges are much more distributed among the different parts of the network. This means that we can still see the local structure of the network in many parts of the network and do not only maintain very dense but disconnected parts. An explanation for this is that a node u has the minimum degree $d(u)^{\alpha}$ which means high-degree nodes loose more incident edges than low-degree nodes but high-degree nodes still keep more neighbors than low-degree nodes. Therefore, some properties are kept but still the differences are smoothened to ensure that we retain structure in every part of the network. In our evaluation, we confirm that many properties of the considered networks are indeed better preserved when the local filtering step is added. Furthermore, we show that the local filtering step leads to an almost perfect preservation of the connected components on all considered networks even though this is not inherent in the method. This suggests that local filtering is superior to preserving a UMST as not only connectivity but also local structures are preserved.

As one of our contributions we, therefore, propose to apply this local filtering step to all sparsification methods and not only to Jaccard Similarity, where the local filtering step has been introduced. In our evaluation, we do not further consider the alternative of preserving a UMST as preliminary experiments have shown that adding a UMST has no significant advantage over the local filtering step in terms of the preservation of network properties. With local filtering, our sparsification pipeline consists of the following stages: (i) calculation of an edge score, (ii) conversion of the edge score into a local edge score and (iii) global filtering. In the evaluation, we prefix the abbreviations of the local variants with "L".

2.2 Limits of edge sparsification

While we show in our work that edge sparsification can be successfully applied to many social networks, we also want to mention the limits of these sparsification approaches.

Let us consider a network that has two times as many edges as nodes, i.e., the average degree is four. If we remove 50 % of the edges and the network is still connected we have a tree with one additional edge. This means that almost all triangles, which are characteristic for a social network, are destroyed. Also a possibly existing community structure that is defined by being particularly dense compared to the rest of the network cannot be maintained in a tree as every community is either a tree or disconnected and it is, therefore, not possible to discriminate between different connected communities based on density. Therefore, one cannot expect to maintain the properties of the network when the number of edges is equal to (or even less than) the number of nodes.

Some of the sparsification methods we present rely on particular structures that are present in many real-world social networks. For our novel method, local degree, we exploit the presence of meaningful hubs that have a relatively high degree. Other methods assume the presence of triangles or quadrangles that describe the community structure of the network. While it is possible that networks do not have these structures, these structures are fundamental characteristics of social networks. Therefore, in the context of social networks relying on these structures is no limitation.

In terms of the memory usage some of the sparsification methods we present need additional memory, but at maximum a separate copy of the graph including edge ids and edge scores. Some of them also use additional memory per thread in the parallel computation, but then only one bit per node, the neighborhood of one node or a queue of in the worst case all nodes. Therefore, memory usage can be a limitation if the considered graph is almost as large as the available memory—but then also the memory needed for storing the edge scores itself might not be available. It is possible to adapt some of these algorithms to use only a limited amount of internal memory and external memory like an HDD or an SSD instead. For example, for triangle counting, a building block of some of the sparsification methods we consider, efficient external memory algorithms exist (Hu et al. 2014). Developing sparsification algorithms

for external memory or also distributed settings is, therefore, an interesting topic for future work to further push the limits of sparsification in terms of computational resources.

3 Sparsification methods

In this section, we describe the sparsification methods that we use in this work. We describe the existing sparsification methods: random edge filtering, Jaccard Similarity, Simmelian Backbones and Algebraic Distances. Further, we introduce Edge Forest Fire and Local Degree as novel sparsification methods. For all methods where this had not been proposed before, we propose to use local filtering as post-processing step. We also present parallel implementations for all methods, though some of them are only partial parallelizations. As most parallelizations are straightforward, we cannot exclude that they have already been used in other implementations. The details are described in the following sections while Table 1 gives an overview of all considered methods.

3.1 Random edge (RE)

When studying different sparsification algorithms, the performance of random edge selection is an important baseline. As we shall see, it also performs surprisingly well. The method selects edges uniformly at random from the original set such that the desired sparsification ratio is obtained. This is equivalent to scoring edges with values chosen uniformly at random. Naturally this needs time linear in the number of edges and calculating the edge scores can be trivially parallelized.

3.2 Triangles

Especially in social networks, triangles play an important role because the presence of a triangle indicates a certain quality of the relationship between the three involved nodes. The sociological theory of Simmel Simmel and Wolff (1950) states that "triads (sets of three actors) are fundamentally different from dyads (sets of two actors) by way of introducing mediating effects." In a friendship network, it is likely for two actors with a high number of common friends to be friends as well. Filtering globally by triangle counts tends to destroy local structures, but several of the following sparsification methods are based on the triangles edge score $T(u, v)$ that denotes for an edge $\{u, v\}$ the number of triangles it belongs to. The time needed for counting the number of all triangles is $O(m \cdot a)$ (Ortmann and Brandes 2014), where a is the graph's arboricity (Chiba and Nishizeki 1985).

Parallelization We use a novel parallelized variant of the algorithm introduced by Ortmann and Brandes (2014). This variant is different from the parallel variant introduced in Shun and Tangwongsan (2015) as they need additional overhead in the form of sorting operations or atomic operations for storing local counters which we avoid. Algorithm 2 contains the pseudo-code for our algorithm. The algorithm needs a node ordering. While a smallest first ordering that is obtained by iteratively removing nodes of minimum degree can guarantee the theoretical running time, simply ordering the nodes by degree is actually faster in practice as noticed by Ortmann and Brandes (2014). Therefore, we use such a simple degree ordering. While $N(u)$ denotes all neighbors of u, $N^+(u)$ denotes the neighbors of the node that are higher in the ordering. Note that when using a smallest first ordering $|N^+(u)|$ is bounded by a. In contrast to Ortmann and Brandes (2014), we count each triangle three times which does not increase the asymptotic running time. In each iteration step of the outer loop we encounter each triangle u and the edges incident to u are part of exactly once. Therefore, it is enough to count the triangle for the edges that are incident to u and where u has the higher id. This avoids multiple accesses to the same edge by several threads, we, therefore, do not need any locks or atomic operations. In the same way we could also update triangle counters per node, e.g., for computing clustering coefficients without additional work and without using locks or atomic operations. Note that node markers are thread local.

Table 1 Summary of the methods and the novel local variants we introduce

Method	Novel	Novel local variant
Random edge	○	•
Jaccard similarity	○	○
Simmelian backbones	○	•
Edge forest fire	•	•
Algebraic distance	○	•
Local degree	•	○

```
1 foreach u ∈ V do in parallel
2     Mark all v ∈ N(u);
3     foreach v ∈ N(u) do
4         foreach w ∈ N+(u) do
5             if w is marked then
6                 Count triangle u, v, w;
7     Un-mark all v ∈ N(u);
```

Algorithm 2: Parallel triangle counting

3.3 (Local) jaccard similarity (JS, LJS)

One line of research attempts to sparsify graphs with the goal of speeding up data mining algorithms. Satuluri et al. (2011) propose a local graph sparsification method with the intention of speedup and quality improvement of community detection. They suggest reducing the edge set to 10–20 % of the original graph and use the Jaccard measure to quantify the overlap between node neighborhoods $N(u)$, $N(v)$ (i.e., the sets of nodes adjacent to u or v) and thereby the (Jaccard) similarity of two given nodes:

$$\mathrm{JS}(u,v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} = \frac{T(u,v)}{d(u) + d(v) - T(u,v)},$$

The time needed for calculating the Jaccard Similarity is the time for counting all triangles. The authors also propose a fast approximation which runs in time $O(m)$.

For this Jaccard Similarity, Satuluri et al. (2011) propose the local filtering technique that we have already explained above and that we denote by LJS. The time needed for calculating this local edge score is the time for calculating the Jaccard Similarity and for sorting the neighbors of all nodes, which can be done in $O(m \log(d_{\max}))$.

This sparsification technique has also been adapted for accelerating *collective classification*, i.e., the task of inferring the labels of all nodes in a graph given a subset of labeled nodes (Saha et al. 2013).

Parallelization With our parallel triangle counting variant and pre-calculated node degrees, the edge scores for Jaccard Similarity can be calculated in parallel. We are not aware of any prior work that calculated the Jaccard Similarity in parallel. As shown in Algorithm 1 also the local filtering can be parallelized. As we will show in our experimental evaluation the achieved running times with our parallel implementation are very good and not far from random edge filtering.

3.4 Simmelian backbones (TS, QLS)

The *Simmelian backbones* introduced by Nick et al. (2013) aim at discriminating between edges that are placed within dense subgraphs and those between them. The original goal of these methods was to produce readable layouts of networks. To achieve a "local assessment of the level of actor neighborhoods" (Nick et al. 2013), the authors propose the following approach, which we adapt to our concept of edge scores. Given an edge scoring method S and a node u, they introduce the notion of a rank-ordered neighborhood as the list of adjacent neighbors sorted by $S(u,\cdot)$ in descending order. The original *(Triadic) Simmelian Backbone* uses triangle counts T for S. The newer *Quadrilateral Simmelian Backbone* by Nocaj et al. (2014) uses *quadrilateral edge embeddedness*, which they define as

$$Q(u,v) = \frac{q(u,v)}{\sqrt{q(u) \cdot q(v)}}$$

with $q(u, v)$ being the number of quadrangles containing edge $\{u, v\}$ and $q(u)$ being the sum of $q(u, v)$ over all neighbors v of u. They argue that this modified version performs even better at discriminating edges within and between dense subgraphs.

On top of the rank-ordered neighborhood graph that is induced by the ranked neighborhoods of all nodes, Nick et al. introduce two filtering techniques, a parametric one and a non-parametric one. Like Nocaj et al. we use only the non-parametric variant. By *TS*, we denote the Triadic Simmelian Backbone and by *QLS* the Quadrilateral Simmelian Backbone. The non-parametric variant uses the Jaccard measure similar to Local Similarity but instead of considering the whole neighborhood, they use the maximum of the Jaccard measure of the top-k neighborhoods for all possible values of k. While the time needed for quadrangle counting is equal to the time for triangle counting (Chiba and Nishizeki 1985), the overlap and Jaccard measure calculation of prefixes needs time $O(m \cdot d_{\max})$ as it needs to be separately calculated for all edges.

Parallelization Our implementation of triangular Simmelian Backbones is fully parallelized, we use our parallel triangle counting implementation, then we sort all neighborhoods in parallel. Using this information we can compute the triangular Simmelian Backbone scores in parallel for all edges. For the quadrilateral Simmelian Backbones, the quadrangle counting step is sequential as we are not aware of an efficient, parallelized quadrangle counting algorithm on edge level but the remaining part of the algorithm is parallelized as in the case of triangular Simmelian Backbones.

3.5 Edge forest fire (EFF)

The original Forest Fire node sampling algorithm (Leskovec and Faloutsos 2006) is based on the idea that nodes are "burned" during a fire that starts at a random node and may spread to the neighbors of a burning node. Note that contrary to random walks the fire can spread to more than one neighbor but already burned neighbors cannot be burned again. The basic intuition is that nodes and edges that get visited more frequently than others during these walks are more important. To filter edges instead of nodes, we introduce a variant of the algorithm in which we use the frequency of visits of each edge as a proxy for its relevance.

Algorithm 3 shows the details of the algorithm we use to compute the edge score. The fire starts at a random node

which is added to a queue. The fire always continues at the next extracted node v from the queue and spreads to neighboring unburned nodes until either all neighbors have been burned or a random probability we draw is above a given burning probability threshold p. The number of burned neighbors thus follows a geometric distribution with mean $p/(1-p)$.

```
Input: targetBurnRatio ∈ ℝ, p ∈ [0, 1)
edgesBurnt ← 0;
while edgesBurnt < m· targetBurnRatio do
    Add random node to queue;
    while queue not empty do
        v ← node from queue;
        while true do
            q ← random element from[0, 1);
            if q > p or v has no un-burnt neighbors
             then
                break;
            x ← random un-burnt neighbor of v;
            Mark x as burnt;
            Add x to queue;
            Increase edgesBurnt;
            Increase burn counter of {v, x};
```

Algorithm 3: Edge Forest Fire

As the total length of all walks is hard to estimate in advance, we cannot give a tight bound for the running time.

Parallelization We use a very simple parallelization for our Edge Forest Fire algorithm. We burn several fires in parallel with separate burn markers per thread and atomic updates of the burn frequency. To avoid too frequent updates we update the global counter for the number of edges burned only after a fire has stopped burning before we start the next fire.

3.6 Algebraic distance (AD)

Algebraic distance (Chen and Safro 2011) (α) is a method for quantifying the structural distance of two nodes u and v in graphs. Its essential property is that $\alpha(u,v)$ decreases with the number of paths connecting u and v as well as with decreasing lengths of those paths. Algebraic distance, therefore, measures the distance of nodes by taking into account more possible paths than, e.g., shortest path distance and with wider in scope than, e.g., the Jaccard coefficient of two nodes' immediate neighborhood. Nodes that are connected by many short paths have a low algebraic distance. It follows that nodes within the same dense subgraph of the network are close in terms of α. Algebraic distance can be described in terms of random walks on graphs and, roughly speaking, $\alpha(u,v)$ is low if a random walk starting at u has a high probability of reaching v after few steps. In a straightforward way, algebraic distance can be used to quantify the "range" of edges, with short-range edges (low $\alpha(u,v)$ for an edge $\{u,v\}$) connecting nodes within the same dense subgraph, and long-range edges (high $\alpha(u,v)$ for an edge $\{u,v\}$) forming bridges between separate regions of the graph. Hence, α restricted to the set of connected node pairs is an edge score in our terms, and can be used to filter out long- or short-range edges. We use $1-\alpha(u,v)$ as edge score to treat short-range edges as important.

α is computed by performing iterative local updates on d-dimensional "coordinates" of a node. The coordinates are initialized with random values. Then, in each iteration, the coordinates are set to some weighted average of the old coordinates and the average of the old coordinates of all neighbors. The algebraic distance is then any distance between the two coordinate vectors, we choose the ℓ_2-norm. As described in Chen and Safro (2011), d can be set to a small constant (e.g., 10) and the distances stabilize after tens of iterations of $O(m)$ running time each. We choose 20 systems and 20 iterations.

Parallelization Updates of node coordinates can be easily executed in parallel for all nodes as the new values only depend on the values of the previous round.

3.7 Local degree (LD)

Inspired by the notion of hub nodes, i.e., nodes with locally relatively high degree, and that of local sparsification, we propose the following new sparsification method: for each node $v \in V$, we include the edges to the top $\lfloor d(v)^{\alpha} \rfloor$ neighbors, sorted by degree in descending order. Similar to the local filtering step we explained above we use again $1-\alpha$ for the minimum parameter α such that an edge is still contained in the sparsified graph as edge score. The goal of this approach is to keep those edges in the sparsified graph that lead to nodes with high degree, i.e., the hubs that are crucial for a complex network's topology. The edges left after filtering form what can be considered a "hub backbone" of the network. In Fig. 2, we visualize the Jazz network (Gleiser and Danon 2003) as an example.

As only the neighbors of each node need to be sorted, this can be done in $O(m \log(d_{\max}))$. Using linear-time sorting it is even possible in $O(m)$ time. We have decided against the linear-time variant and instead parallelized the calculation:

Parallelization The parallelization of the Local Degree score calculation is very similar to the parallelization for local filtering. We can handle all nodes in parallel. For each node we can independently sort the neighbors and assign the appropriate scores. Again we need to use atomic maximum calculation to make sure that parallel updates of edge scores are handled correctly.

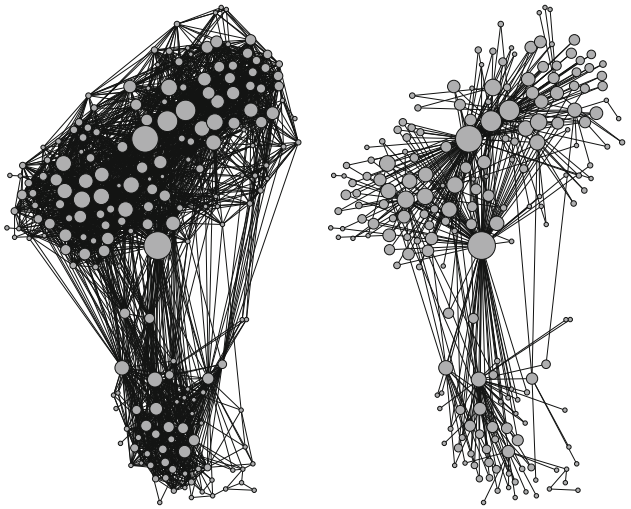


Fig. 2 Drawing of the Jazz musicians collaboration network and the *Local Degree* sparsified version containing 15 % of edges. Node size proportional to degree

4 Implementation

For this study, we have created efficient C++ implementations of all considered sparsification methods, and have accelerated them using OpenMP parallelization. All methods [with exception of the inherently sequential quadrangle counting algorithms (Chiba and Nishizeki 1985)] have been parallelized. We have implemented the algorithms in *NetworKit* (Staudt et al. 2014), an interactive tool suite for scalable network analysis. It provides a large set of graph algorithm implementations we have used for our experiments. NetworKit combines kernels written in C++ with an interactive Python shell to achieve both high performance and interactivity, a concept we use for our implementations as well.

Gephi (Bastian et al. 2009) is a graph visualization tool which we use not only for visualization purposes but also for interactive exploration of sparsified graphs. To achieve said interactivity, we implemented a client for the *Gephi Streaming Plugin* in NetworKit. It is designed to stream graph objects from and to Gephi utilizing the JSON format. Using our implementation in NetworKit, a few lines of Python code suffice to sparsify a graph, calculate various network properties, and export it to Gephi for drawing. The approach of separating sparsification into edge score calculation and filtering allows for a high level of interactivity by exporting edge scores from NetworKit to Gephi and dynamic filtering within Gephi.

For the drawings of the Simmelian Backbones we use *visone*[1].

[1] http://visone.info.

5 Experimental study

Our experimental study consists of two parts. In the first part (Sec. 5.2) we compare correlations between the calculated edge scores on a set of networks. In the second part (Sec. 5.3) we compare how similar the sparsified networks are to the original network by comparing certain properties of the networks.

5.1 Setup

Our experiments have been performed on a multicore compute server with 4 physical Intel Core i7 cores at 3.4 GHz, 8 threads, and 32 GB of memory. For this explorative study, we use a collection of 100 social networks representing early snapshots of Facebook, each of which is an online friendship network for a US university or college (Traud et al. 2012), most members are students. Sizes of the Facebook networks are between 10 k and 1.6 million edges, the number of nodes and edges is shown in Fig. 3.

Unless otherwise noted, we aggregate experimental results over this set of networks. The common origin and the high structural similarity among the networks allow us to get meaningful aggregated values.

For the experiments on the preservation of properties we also use the Twitter and Google+ networks (Leskovec and Mcauley 2012) and the LiveJournal (com-lj) network from Yang and Leskovec (2012). All of them are friendship networks, the Twitter and Google+ networks consist of the combined ego networks of 973 and 132 users, respectively. In Table 2, we provide the number of nodes and edges as well as diameter and clustering coefficient averaged over all Facebook networks and the individual values for the three other networks. Furthermore, we provide the number

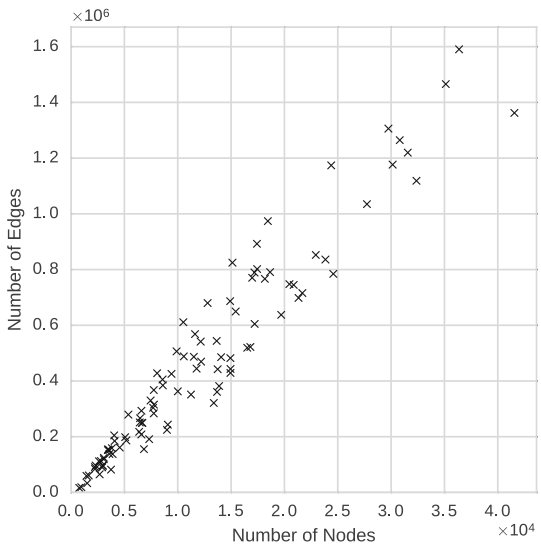


Fig. 3 Number of nodes and edges of the Facebook networks

Table 2 Number of nodes n, the number of edges m, m/n, the diameter D and the average local clustering coefficient Cc of the used social networks (average values for the Facebook networks)

Network	n	m	m/n	D	Cc
Facebook	12 083	469 845	38.4	7.8	0.25
com-lj	3 997 962	34 681 189	8.7	21	0.35
gplus	107 614	12 238 285	113.7	6	0.52
Twitter	81 306	1 342 296	16.5	7	0.60

of edges divided by the number of nodes, which indicates how much redundancy there is in the network. As we already mentioned in Sect. 2.2, it is not realistic to expect that we can preserve the structure of the network (apart from the connectivity) if too few edges remain. The networks we selected have a varying degree of redundancy, but all of them are dense enough such that even if we remove 75 % of the edges more than twice as many edges as nodes remain. The characteristics of the Facebook networks are relatively similar, we provide the individual values in the Appendix.

For the evaluation of the preservation of the community structure we also use networks with known ground truth communities. For this purpose, we use the synthetic LFR benchmark (Lancichinetti and Fortunato 2009a).

It remains an open question to what extent results can be translated to other types of complex networks, since according to experience the performance of network analysis algorithms depends strongly on the network structure.

5.2 Correlations between edge scores

Among our sparsification methods, some are more similar to others in the sense that they tend to preserve similar edges. Such similarities can be clarified by studying correlations between edge scores. We calculate edge score correlations for the set of 100 Facebook networks as follows: for each single network, edge scores are calculated with the various scoring methods and Spearman's rank correlation coefficient is applied. The coefficient is then averaged over all networks and plotted in the correlation matrix (Fig. 4). There is one column for each method, and the column Mod represents edge scores that are 1 for intra-community edges and 0 for inter-community edges after running a modularity-maximizing Louvain community detection algorithm. Positive correlations with these scores indicate that the respective rating method assigns high scores to edges within modularity-based communities. The column Tri simply represents the number of triangles an edge is part of. As some of the methods are normalizations

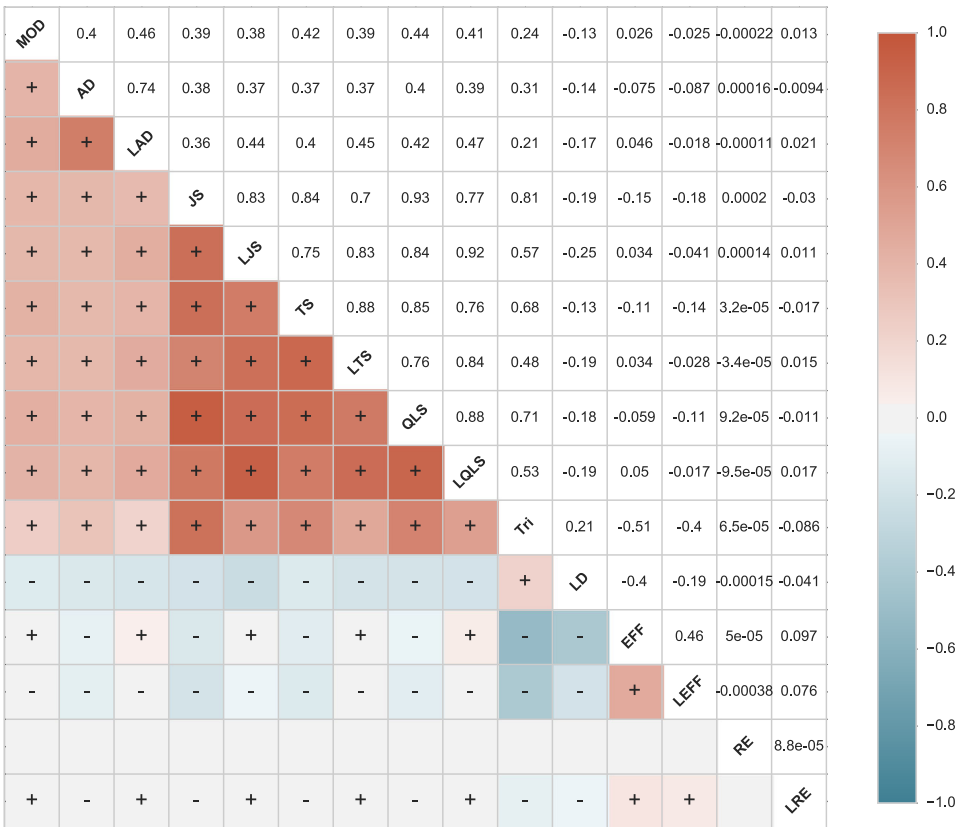


Fig. 4 Edge score correlations (Spearman's ρ, average over 100 Facebook networks)

of this score, this shows how similar the ranking still is to the original score.

Interpretation of the results is challenging: the correlations we observe reflect intrinsic, mathematical similarities of the rating algorithms on the one hand, but on the other hand they are also caused by the structure of this specific set of social networks (e.g., it may be a characteristic of a given network that edges leading to high-degree nodes are also embedded in many triangles). Nonetheless, we note the following observations: there are several groups of methods. Simmelian Backbones, Jaccard Similarity and Triangles are highly positively correlated which is not unexpected as they are all based on triangles or quadrangles and are intended to preserve dense subgraphs. Algebraic distance is still positively correlated with these methods but not as strongly even though they are also intended to prefer dense subgraphs. An explanation for this weaker correlation is that while both prefer dense regions, the order of the individual edges is different. All of the previously mentioned methods are also correlated with the Modularity value, algebraic distance with local filtering has the highest score among all of these methods. Our experiments on the preservation of community structure (Sec. 5.3) confirm this relationship. The correlation of the Modularity value and these methods are similar to the correlation between algebraic distance and the rest of the methods which shows again that the lower correlation values are probably due to different orderings of the individual edges.

Our new method Local Degree is slightly negatively correlated with all these methods but still positively correlated with the Triangles. It is also slightly negatively correlated with the Modularity value, this is due to the method's preference of inter-cluster edges which is also confirmed by our experiments below. The newly introduced Edge Forest Fire is also negatively correlated with Local Degree and even more negatively with Triangles. This strong negative correlation between Edge Forest Fire and triangle count can be explained by the fact that the Edge Forest Fire can never "burn" a triangle, as nodes cannot be visited twice. Random edge filtering is not correlated at all, which is definitely expected.

It is interesting to see that each method is also relatively strongly correlated with its local variant, apart from random edge filtering (we use different random values as basis of the local filtering process). Even the Edge Forest Fire method, which should also be relatively random, has a positive correlation with its local variant. This shows that it prefers a certain kind of edge and that this preference is kept when applying the local filtering.

Among the variants of Simmelian Backbones and Jaccard Similarity also the local variants are more correlated to other local variants than to other non-local variants and also not as strongly correlated to triangles. This shows that the local filtering indeed adds another level of normalization. Also Jaccard Similarity seems to be more correlated to Quadrilateral Simmelian Backbones than to the variant based on triangles even though Jaccard Similarity is based on triangles itself. This is also interesting to see, as Quadrilateral Simmelian Backbones are computationally more expensive than the Jaccard Similarity.

5.3 Similarity in network properties

Quantifying the similarity between a network and its sparsified version is an intricate problem. Ideally, a similarity measure should meet the following requirements:

1. *Ignoring trivial differences* consider, for example, the degree distribution: One cannot expect the distribution to remain identical after edges get removed during sparsification. It is clear, however, that the general shape of the distribution should remain "similar" and that high-degree nodes should remain high-degree nodes to consider the degrees as preserved.
2. *Intuitive and normalized* similarity values from a closed domain like [0, 1] allow for aggregation and comparability. A similarity value of 1 indicates that the property under consideration is fully preserved, whereas a value of 0 indicates that similarity is entirely lost. In some cases, we also used relative changes in the interval $[-1, 1]$ where 0 means unchanged as they provide a more detailed view at the changes.
3. *Revealing method behavior* a good similarity measure will clearly expose different behavior between sparsification methods.
4. *Efficiently computable*.

Following these requirements, we select measures that quantify relative changes for global properties like diameter, size of the largest connected component and quality of a community structure. Node degree, betweenness and PageRank can be treated as node centrality indices which represent a ranking of nodes by structural importance. Since absolute values of the centrality scores are less interesting than the resulting rank order, we compare the rankings before and after sparsification using Spearman's ρ rank correlation coefficient. (this focus on rank order is also the reason why we did not adopt the Kolmogorov–Smirnov statistic used in Leskovec and Faloutsos (2006), which compares distributions of absolute values.) Even though the local clustering coefficient can be interpreted as a centrality score as well, the comparison of ranks does not seem meaningful in this case due to the fact that it is a local score. Instead, we analyze the deviation of the average local clustering coefficient from the original value.

In the following plots, the measures are shown on the y-axis for a given ratio of kept edges (m'/m) on the x-axis (e.g., a ratio of 0.2 means that 20 % of edges are still present). For each value there are two rows of plots. The first contains averages over the 100 Facebook networks with error bars that indicate the standard deviation. The second row contains the values at 20, 50 and 80 % remaining edges of the three other networks. In each row, we show two plots: the left plot with the non-local methods and the right plot with the methods that use local filtering.

Connected components All nodes in a *connected component* are reachable from each other. A typical pattern in real-world complex networks is the emergence of a giant connected component which contains the bigger part of all nodes and is usually accompanied by a large number of very small components. As all of our networks have such a giant component that comprises most nodes, we track its change by dividing the size of the largest component in the sparsified network by the size of the largest component in the original network. As shown in Fig. 5, out of the non-local methods Edge Forest Fire best preserves the connected component. Random edge deletion leads to a slow decrease in the size of the largest component while Simmelian Backbones, Jaccard Similarity and algebraic distance lead to a separation very quickly. Below 20 % of retained edges, the size of the largest component on the Facebook networks drops very quickly, here the networks seem to be decomposed into multiple smaller parts. On the other networks, this drop occurs at different ratios of kept edges which reflects their different densities and probably also their different structures. Local Filtering is able to maintain the connectivity. On the Facebook networks, all methods keep the largest component almost fully connected up to 20 % of retained edges, only below that small differences are visible. The results on the

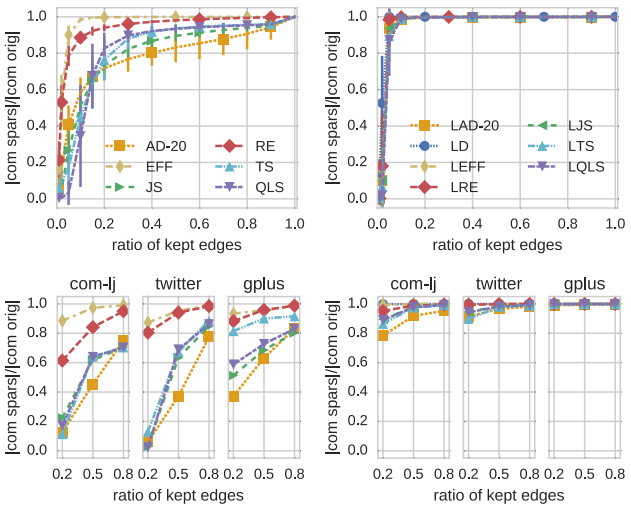


Fig. 5 The size of the largest component in the sparsified network divided by the size of the largest component in the original network

LiveJournal, Twitter and Google+ networks show that—as expected—with increasing density it is easier to preserve the connectivity of the network. Our Local Degree method best preserves the connected components of all networks, closely followed by the local variant of random edge deletion and Edge Forest Fire.

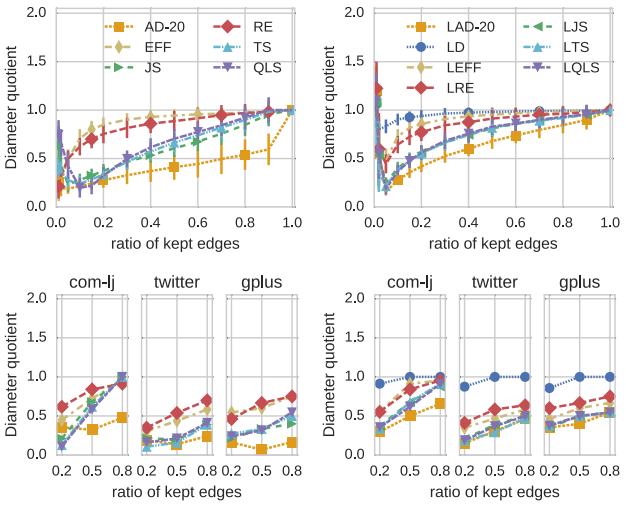
Diameter The *diameter* of a graph is the maximum length of a shortest path between any two nodes (Newman 2010). The diameter of social networks is often surprisingly small, this is also known as the *small-world phenomenon*. In case of disconnected graphs, we consider the largest diameter of all connected components. To observe how the network diameter changes through sparsification, we plot the quotient of the original network diameter and the resulting diameter, which yields legible results since in practice the diameter is mostly increased during sparsification. We compute the exact diameters using a variation of the ExactSumSweep algorithm (Borassi et al. 2015).

We motivate the Local Degree method with the idea that shortest paths commonly run through hub nodes in social networks. Therefore, preserving edges leading to high-degree nodes should preserve the small diameter. This is confirmed by our experiments (Fig. 6a). In contrast, methods that prefer edges within dense regions clearly do not preserve the diameter. With Simmelian Backbones the diameter drops when only few edges are left; this can be explained by the fact that Simmelian Backbones do not maintain the connectivity and that at the end the graph is decomposed into multiple connected components which have a smaller diameter. Algebraic distance is even more extreme in this aspect. Local filtering leads to a slightly better preservation of the diameter when applied to the other methods but algebraic distance remains the worst method in this regard. Note that the LiveJournal network has a higher diameter than the other networks (see Table 2); this might explain why the diameter is better preserved there.

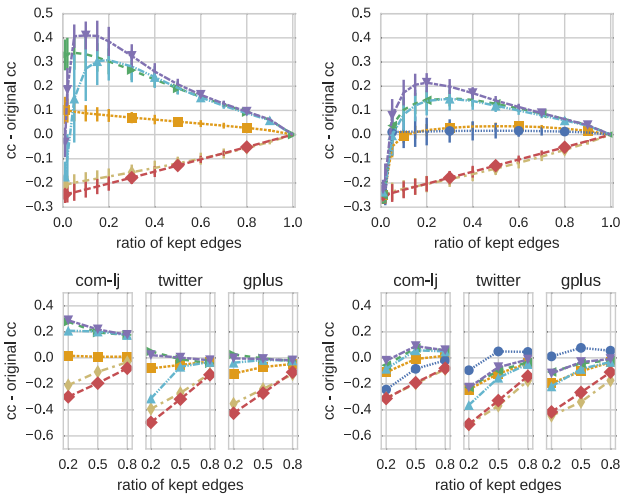
Clustering coefficient are key figures for the amount of transitivity in networks. The *local clustering coefficient* expresses how many of the possible connections between neighbors of a node exist (Newman 2010). Figure 6b shows the deviation of the average local clustering from the value of the original network. Both for local and non-local methods we observe three classes of methods on the Facebook networks: methods that clearly decrease the clustering coefficient, methods that preserve the clustering coefficient and methods that increase it.

For both Random Edge and Edge Forest Fire, which are based on randomness, the clustering coefficient drops almost linearly with decreasing sparsification ratio. This can also be observed on the other three networks. The additional local filtering step does not significantly change this.

Simmelian Backbones and Jaccard Similarity keep mostly edges within dense regions, which results in



(a) Original network diameter divided by network diameter



(b) Deviation from original clustering coefficient

Fig. 6 Preservation of global network properties

increasing clustering coefficients on all networks. Triadic Simmelian Backbones show the weakest increase, on the Twitter network even a decrease of the clustering coefficients. Note that with 0.52 and 0.6, the clustering coefficients are already relatively high on the Google+ and Twitter networks; therefore, the very small increase is not surprising. Local filtering slightly weakens this effect on the Facebook networks, on the other networks it is even reversed. Given the high clustering coefficients in the original networks, this is not very surprising as we would need to retain very dense areas while local filtering leads to a more balanced distribution of the edges.

From the previous results especially concerning the connected components one would expect that algebraic distance also increases the clustering coefficients. Interestingly though, filtering using algebraic distance leads to a slight increase of the clustering coefficient on the Facebook networks, constant clustering coefficients on the LiveJournal network and even slightly decreasing clustering coefficients on the Twitter and Google+ networks. With the additional local filtering step algebraic distance almost preserves the clustering coefficients on the Facebook networks while on the other networks it is slightly decreased. Algebraic distance leads to random noise on the individual edge weights; therefore, they probably lead to a more random selection of edges that also destroys more triangles than the selection of Simmelian Backbones and Jaccard Similarity. Our Local Degree method best preserves the clustering coefficient on the Facebook networks, though with some differences between the various networks in the dataset (note the error bars). On the LiveJournal network it leads to a decrease of the clustering coefficient while on the Twitter and Google+ networks it leads to a slight increase of the clustering coefficient. This is probably due to the special structure of ego networks.

Our experiments, therefore, do not reveal a general best method for preserving clustering coefficients. If high clustering coefficients shall be created or preserved, the Jaccard Similarity and Quadrilateral Simmelian Backbones seem to be a good choice. Algebraic distance is good at preserving clustering coefficients with slight deviations but our Local Degree method also works well on the considered social networks.

Node centrality measures quantify the relative importance of a node within the network structure. We consider any function which assigns to each node an attribute value of at least ordinal scale of measurement to be a node centrality measure. The distribution of *degrees*, the number of connections per node, can be seen as the simplest measure that falls under this definition. It plays an important role in characterizing a network: empirically observed complex networks tend to show a heavy tailed *degree distribution* which follows a power law with a characteristic exponent: $p(k) \sim k^{-\gamma}$. Such networks have been categorized as *scale-free* (Barabási and Albert 1999), referring to the fact that it is not possible to pick a node of typical degree. *Betweenness centrality* expresses the concept that a node is important if it lies on many shortest paths between nodes in the network. *PageRank* (Page et al. 1999) assigns relative importance to nodes according to their connections, incorporating the idea that edges leading to high-scoring nodes contribute more.

The exact calculation of betweenness centrality is in practice too expensive for the whole set of networks and sparsification methods we consider. Therefore, we use the approximation algorithm (Geisberger et al. 2008) with at least 16 samples, for smaller networks also with up to 512

samples. For the calculation of the PageRank centrality we use a damping factor of 0.85 and an error tolerance of 10^{-9}.

The similarity of curves in Fig. 7 catches the eye immediately: for these node centrality measures, the sparsification methods behave in a very similar way. This similarity could be explained by strong correlations between node degree, PageRank and betweenness, which have been observed before (e.g., Fortunato et al. 2008). Note that betweenness centrality is also not exactly preserved on the original network; this is due to the approximation, which adds additional noise.

Random edge deletion and Local Degree perform best on most networks. In accordance with our intuition that edges leading to high-degree neighbors are important and should be preserved, our experiments show that the Local Degree method preserves all three considered node centralities. Nevertheless, random edge filtering with the additional local filtering step outperforms it concerning the preservation of Betweenness Centrality. The differences are small though and similar to those that are due to the approximation error so they might actually be caused by the approximation method for Betweenness centrality that behaves differently depending on the structure of the network. On the Facebook networks, Edge Forest Fire fails early while on the other networks it is among the best methods. As the expected number of randomly selected incident edges via the "burning process" of Edge Forest Fire is relatively low even for high-degree nodes, it fails at preserving node degrees. Nevertheless, in the non-Facebook networks it seems to preserve enough important connections to preserve PageRank and betweenness centralities relatively well. Methods that are focused on keeping edges within dense regions are not as good at preserving these centralities. Adding the additional local filtering step again leads to a better preservation of the properties but does not change the general picture.

Community structure Communities are subsets of nodes that are internally dense and externally sparsely connected. For community detection, we use an efficient implementation of the Louvain method with refinement that is also part of NetworKit (Staudt and Meyerhenke 2016) as it is fast enough for the vast amount of networks that we get due to the different sparsification methods and ratios of kept edges while still detecting communities of a reasonable quality. To understand how the community structure of the networks is maintained, we consider for each network a fixed community structure that has been found by the Louvain method on the original network. We report some properties of this community structure for each level of sparsification. There are many ways to characterize a community structure. We pick two properties of communities that we consider to be crucial. Communities are

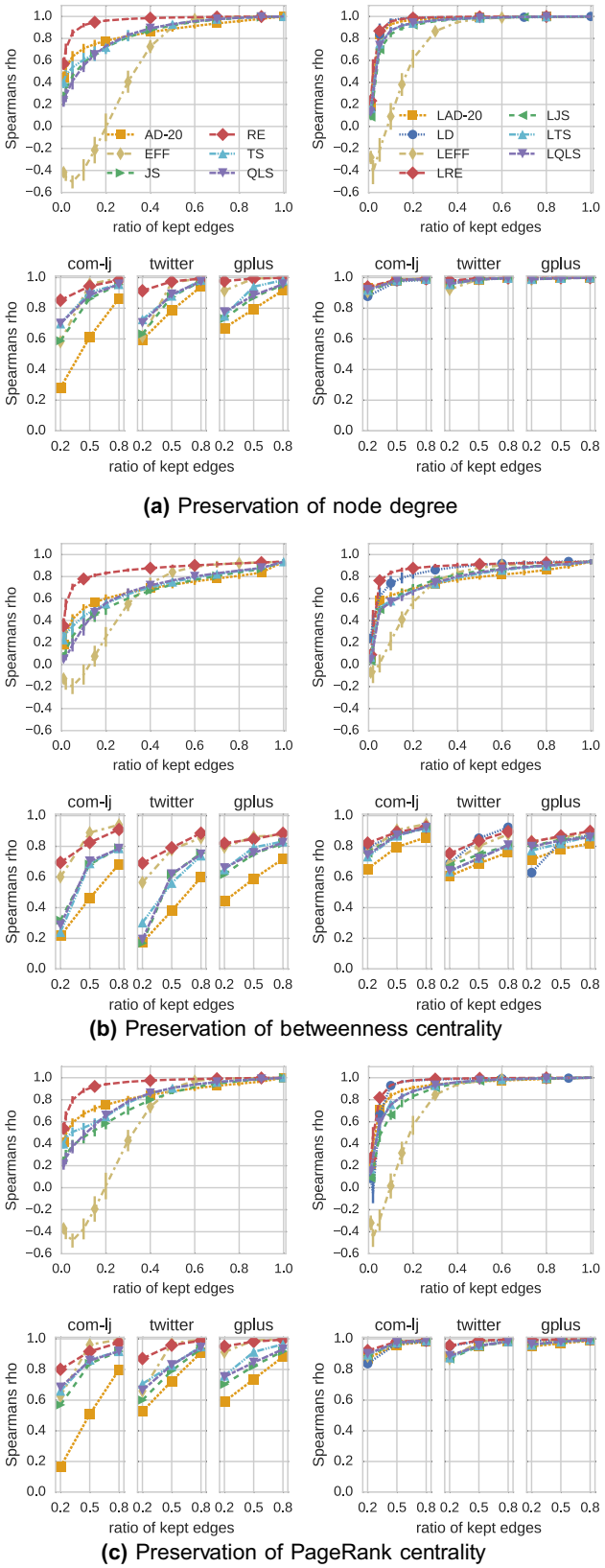
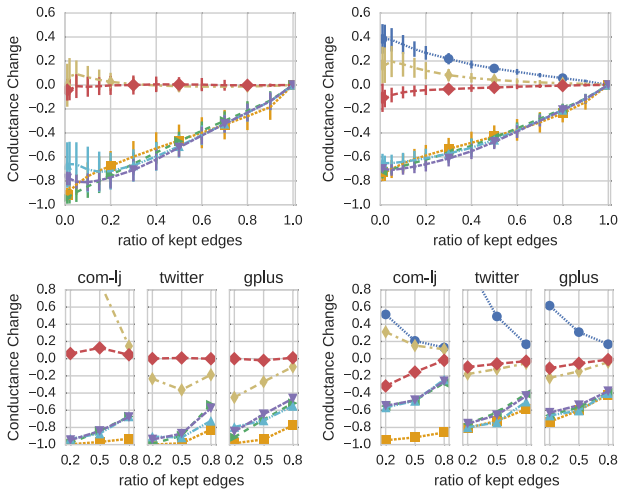


Fig. 7 Preservation of the ranking of node centrality measures (Spearman's ρ rank correlation coefficient)

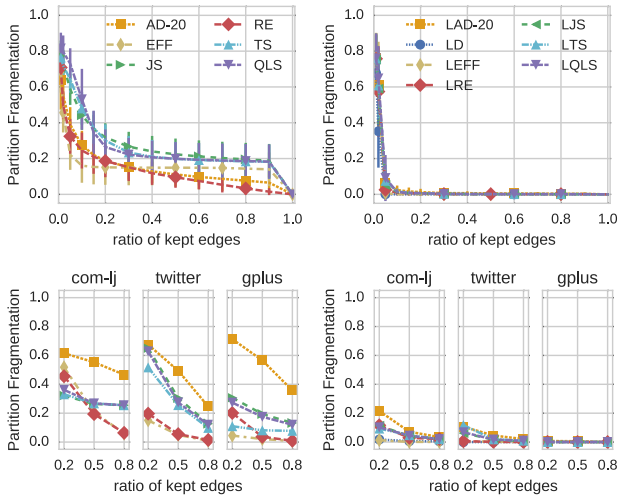
commonly described to be internally dense and externally sparse subgraphs. A natural measure is thus *conductance* which compares the size of the cut of a community to the volume of the community, i.e., the sum of all degrees (or the volume of the rest of the network if it should be larger). Low conductance values indicate clearly separable communities. We consider the average conductance of all communities. Furthermore, we expect that communities are connected. To measure this, we introduce the fraction of the nodes in a community that does not belong to the largest connected component of the community as partition fragmentation. We report the average fragmentation of all communities.

We plot the relative inter-cluster conductance change in Fig. 8a. A value of 0 means that the conductance stays the same, a value of -1 indicates that the conductance became 0 (i.e., a decrease by 100 %) and a value of 1 indicates that the conductance has been doubled (i.e., an increase of 100 %). We can again see that there are three categories of algorithms: the first group consisting of random edge sampling preserves the conductance values on most networks. The second group contains only Local Degree and increases the conductance. Edge Forest Fire has no clear behavior. On the LiveJournal network it increases the conductance, while on Twitter and Google+ it rather decreases it. On the Facebook networks, Edge Forest Fire without local filtering preserves the conductance values while with local filtering the conductance values are slightly increased. The third group consisting of Jaccard Similarity, Simmelian Backbones and algebraic distance strongly decreases the conductance. With the additional local filtering step the decrease in conductance is not as strong but still very significant. The keeping of inter-community edges of the Local Degree method, which also explains why it preserves the connectivity so well, can be explained as follows: consider a hub node x within a community with neighbors that are for the most part also connected to a hub node y with higher degree than x. Due to the way Local Degree scores edges, x will lose many of its connections within the community and may be pulled into the community of a neighboring high-degree node z that is not part of the original community of x. Jaccard Similarity, Simmelian Backbones and algebraic distance on the other hand focus—by design—on intra-community edges. Random edge sampling and Edge Forest Fire filter both types of edges almost equally distributed which is not surprising given their random nature. Depending on the network Edge Forest Fire shows different behavior, this indicates that these networks have a different structure.

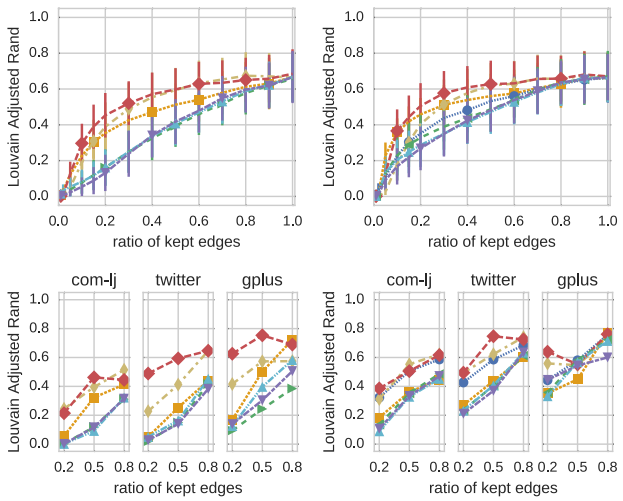
In Fig. 8b it becomes obvious that only local filtering allows methods to keep the intra-cluster connectivity up to very sparse graphs. On the Facebook networks, Simmelian Backbones and Jaccard Similarity without local filtering



(a) Relative conductance change of a fixed partition



(b) Average partition fragmentation



(c) Adjusted rand measure between partition into communities on the original network and on the sparsified network

Fig. 8 Preservation of community structure

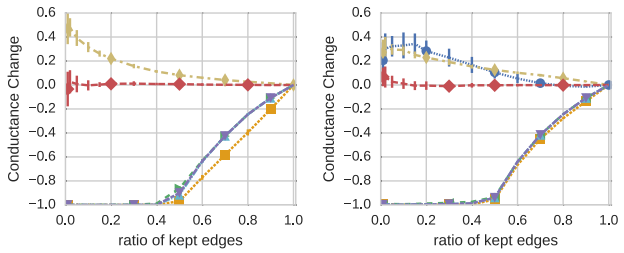
are actually the worst in this respect, they do not keep the connectivity even though they prefer intra-cluster edges as we have seen before. On the other networks, algebraic distance is even more extreme in this regard. Random edge sampling and Edge Forest Fire on the non-Facebook networks are the only non-local method where a slow increase of the fragmentation can be observed, all other methods lead to a steep increase of the fragmentation during the first 10 % of edges that are removed.

Given these observations, we expect that we should still be able to find a very similar community structure at least if we use Simmelian Backbones, Jaccard Similarity or algebraic distance with local filtering. In Fig. 8c we compare the community structure that is found by the Louvain method on the sparsified network to the one found on the original network. For this comparison we use the adjusted rand index (Hubert and Arabie 1985). Note that the frequently used normalized mutual information (NMI) measure reports higher similarity values for a larger number of found communities (see e.g., Vinh et al. 2009). This makes it unsuitable for comparing partitions on sparsified networks as we have to expect many small communities when a lot of edges are removed. As Vinh et al. (2009) also show in their experiments, the adjusted rand index does not have these properties as it has an expected value of 0 for random partitions.

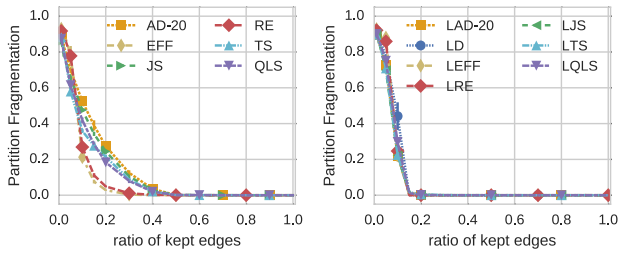
As a first observation we need to note that even when all edges are still in the network (at the right boundary of the plot for the Facebook networks), the community structure found is already different. The Louvain method is randomized; therefore, it is not unlikely that every found community structure is different. The amount of difference between the community structures even without filtering edges indicates that there is not a single, well-defined community structure in these graphs but many different ones. Preliminary tests with the (slightly slower) Infomap community detection algorithm (Rosvall et al. 2009) which has an excellent performance on synthetic benchmark graphs for community detection (Lancichinetti and Fortunato 2009b) show a very similar variance which indicates that this is not due to a weakness of the Louvain algorithm. Filtering the edges such that we can measure that the conductance of one of these many community structures is decreased most probably does not just make this structure clearer but does also lead the algorithm into finding different community structures. Therefore, most methods lead to significantly different community structures. It is possible that some of the sparsification methods, especially local variants of the Simmelian Backbones, Jaccard Similarity and algebraic distance, simply reveal a different community structure. On the contrary, if all edges are kept with the same probability, the almost same set of community structures can still be found up to a certain ratio of kept edges. Removing less than 40 % of the edges at random using random edge sampling or Edge Forest Fire does not seem to lead to more different structures than the already found ones. Note that on the three other networks these results are hard to interpret as they are from just a single run, but the general tendencies are similar. Local methods keep the connectivity and are thus slightly better at preserving the community structure.

To verify the hypothesis that some differences are due to different community structures being found, we use synthetic networks with ground truth communities. For this purpose, we use the popular LFR generator (Lancichinetti and Fortunato 2009a). As parameters we choose the configuration with 1000 nodes and small communities from the study by Lancichinetti and Fortunato (2009b). This means our synthetic networks have a power law degree distribution with exponent -2, average degree 20 and maximum degree 50. The communities have between 10 and 50 nodes, the community sizes also follow a power-law distribution but with exponent -1. As mixing parameter μ we choose 0.5. This means that each node has as many neighbors in its own community as in all other communities together. For smaller mixing parameters the differences between the different techniques are less obvious, for larger mixing parameters we reach the limits where community detection algorithms are no longer able to identify the ground truth communities. For the plots in Fig. 9, we use ten different random networks with the same configuration and report again the average and the standard deviation. As we have known ground truth communities for these networks, we use these ground truth communities instead of a community structure found by the Louvain algorithm for the following comparisons.

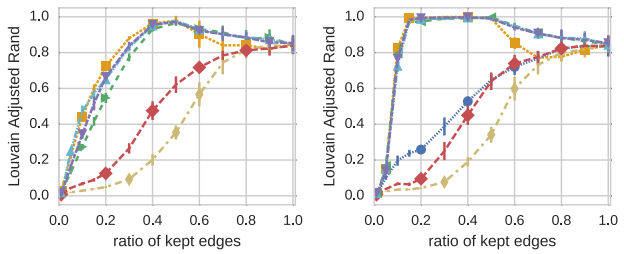
For the inter-cluster conductance in Fig. 9a, the results are similar to the results for the Facebook networks but much clearer. Random edge filtering almost perfectly preserves the inter-cluster conductance both in the variant without and the one with local filtering. Edge Forest Fire and Local Degree lead to a clear increase of the conductance, again independent of the local filtering step. Compared to the Facebook networks this increase for Edge Forest Fire is now much stronger and earlier in the sparsification process. Simmelian Backbones, Jaccard Similarity and algebraic distance lead to a strong decrease of the inter-cluster conductance. With 40 % remaining edges, the conductance reaches almost 0 for all of them. Note that if a measure was able to perfectly distinguish between intra- and inter-cluster edges, the inter-cluster conductance could reach 0 when the ratio of kept edges reaches 50 %. All methods are not far from that goal, but algebraic distance is the best method in this regard. With a local filtering post-processing step, algebraic distance is more similar to the other methods. For the other methods, only minor changes can be observed. It is visible, though, that some inter-cluster edges seem to remain.



(a) Relative conductance change of the ground truth communities



(b) Average partition fragmentation of the ground truth communities



(c) Adjusted rand measure between ground truth communities and found communities on the sparsified network

Fig. 9 Preservation of the community structure on the generated LFR graphs

On the LFR networks, the connectivity in the communities seems to be preserved much better than on the Facebook networks, see Fig. 9b. Up to 50 % of removed edges, none of the methods leads to any noticeable fragmentation. Only when more edges are removed, the Simmelian Backbones, Jaccard Similarity and algebraic distance seem to disconnect parts of the communities. With the additional local filtering step the connectivity inside communities is almost perfectly preserved up to 15 % remaining edges. As the networks have an average degree of 20 we also cannot expect that connectivity is preserved much further as with 10 % remaining edges only a tree could be preserved.

In Fig. 9c, we compare the ground truth communities to the community structure found by the Louvain algorithm (again with refinement). Without the sparsification step, the Louvain algorithm is unable to detect the ground truth communities, this is most likely due to the resolution limit (Fortunato and Barthelemy 2007). On the generated networks with clear ground truth communities, our intuition that random edge, local degree and edge forest fire should not be able to preserve the community structure is verified. While removing less than 20 % of the edges leads to similar detection rates, the differences increase as more and more edges are removed. Edge Forest Fire is worst at keeping the community structure on LFR networks. On the contrary, those methods that show positive results for the preservation of the community structure actually lead to sparsified networks where the Louvain algorithm is better able to find the community structure. As we could expect from the partition fragmentation, without local filtering the detection rate is decreased after 50 % of the edges have been removed. With local filtering, though, there is a range between 50 and 15 % of remaining edges where the Louvain algorithm can almost exactly recover the ground truth communities on the sparsified network. This shows that sparsification can even increase the quality of communities found by community detection algorithms. Algebraic distance with local filtering seems to work best in this regard. During the first 50 % of removed edges, algebraic distance shows a strange behavior, though—especially with local filtering—the detection rate first drops a bit. This is surprising as algebraic distance leads to the strongest decrease of the inter-cluster conductance right at the beginning. A possible explanation is that the Louvain algorithm merges especially small clusters. If other methods filter edges between these small clusters first, this most probably helps the Louvain algorithm most.

With all these experiments we have seen that measuring the preservation of the community structure is a challenging task especially when no ground truth communities are known. Our results suggests that the social networks either do not contain a clear community structure or that the Louvain algorithm is unable to identify this structure. Random edge deletion and edge forest fire seem to preserve this uncertainty in the sense that the Louvain algorithm still identifies relatively similar communities. Simmelian Backbones, Jaccard Similarity and algebraic distance on the other hand are designed to prefer intra-cluster edges which can also be seen in our experiments. On the sparsified networks this has the effect that the Louvain algorithm detects communities that are different from the communities it detects on the original network. On synthetic networks, local filtering with these methods preserves and even enforces the community structure. They are able to preserve the ground truth communities up to a ratio of kept edges of 0.15. This suggests that Simmelian Backbones, Jaccard Similarity and algebraic distance with local filtering indeed keep and enforce some community structure but that on networks without clearly detectable community structure this is not necessarily the same structure as the structure that is found by the Louvain algorithm.

5.4 Epidemic simulations

The previous experiments focused on static structural properties only. Now we briefly turn to dynamic, emergent properties that can be observed by simulating processes on networks. Epidemic models are simplified means of describing the transmission of communicable diseases through a population of individuals, which can intuitively be applied to networks. Studies have recognized the importance of social networks in disease transmission (Salathé et al. 2010). In the following, we apply the SEIR model (Keeling and Rohani 2008), which assigns one of four states to each node: initially one node has been exposed (E) to the infection and all other nodes are susceptible (S). An exposed node becomes infectious (I) after a number of time steps. At each time step, an infectious node contacts all of its neighbors, and with a certain transmission probability, a susceptible neighbor becomes exposed. Nodes stay infectious for a given number of steps, and are then removed (R), either by immunization or death. Counting the number of nodes of each state at each time step yields epidemic curves that describe the dynamics of the outbreak.

There is a nontrivial relationship between network structure and epidemic dynamics [e.g., they have been connected to spectral properties of the graph (Wang et al. 2003)]. We can ask whether sparsified versions of a network give rise to similar epidemiological dynamics, in terms of the size and timing of a disease outbreak, and add another level of analysis for the sparsification methods. We select `fb-Texas84` (ca. 1.6 million edges) as a representative social network and run the SEIR simulation 50 times with a latency period of 2 time steps, an infectious period of 9 time steps, and a transmission probability of 0.1. Figure 10 shows the aggregated epidemic curves (where the central line represents the median number of nodes and the shaded areas around it the standard deviation) for the original network. While epidemic dynamics can depend strongly on the specific network structure, the following observations were roughly consistent across the Facebook-type networks.

The Local Degree method most closely replicates the epidemic curves of the original down to an edge ratio of 0.2, producing only a minor delay in the outbreak and slightly lower peak number of infected nodes, but an identical converged state (Fig. 11). One reason for this is certainly that connectedness and short paths are preserved. It may also point to the importance of local hubs in epidemic propagation. Random edge sampling and Forest Fire sparsification also perform well and produce a similar high fidelity. Other methods deviate more from the original epidemic dynamics by delaying and dampening the outbreak, with some also strongly reducing the final number of infections. For Local Jaccard Similarity, thinning the network slows the outbreak slightly, leading to a less sharp and high peak of infected nodes, but reproduces essentially the same epidemic curve shapes and final state. At the other end of the spectrum, sparsification by algebraic distance (Fig. 12) and the Simmelian methods selectively remove bottleneck edges between dense regions of the network. These edges are likely to be critically important for the propagation of a disease, and hence epidemic dynamics are significantly altered with deleting those edges.

5.5 Running time

Measured running times are shown in Fig. 13. Random Edge sparsification is clearly the fastest method, closely followed by Local Degree. Jaccard Similarity is also not much slower and scales also very well. Therefore, these methods are well suited for large-scale networks in the range of millions to billions of edges. The efficiency of the Jaccard Similarity method shows that our parallel triangle counting implementation is indeed very scalable. The authors also proposed inexact Jaccard coefficient calculation for a further speedup though given our numbers it can be doubted if—given an efficient triangle counter—this is necessary or helpful at all. Algebraic distance is a bit slower but scales very well nevertheless. Using less systems or iterations could further speed up algebraic distance if speed is an issue. Both Simmelian methods are significantly slower than the other methods, but still efficient enough for the network sizes we consider. The visible difference between quadrilateral and triangular Simmelian Backbones can be explained by the difference between

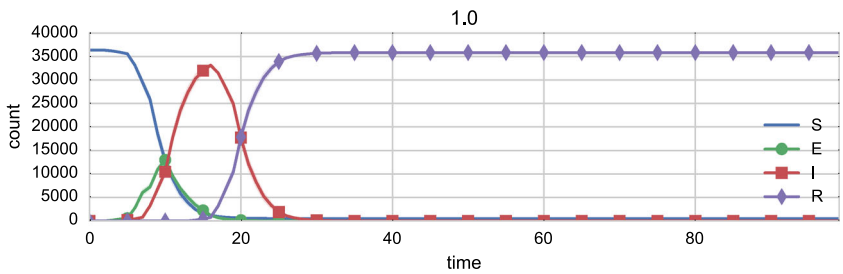


Fig. 10 Epidemic curves of SEIR simulation on a Facebook social network

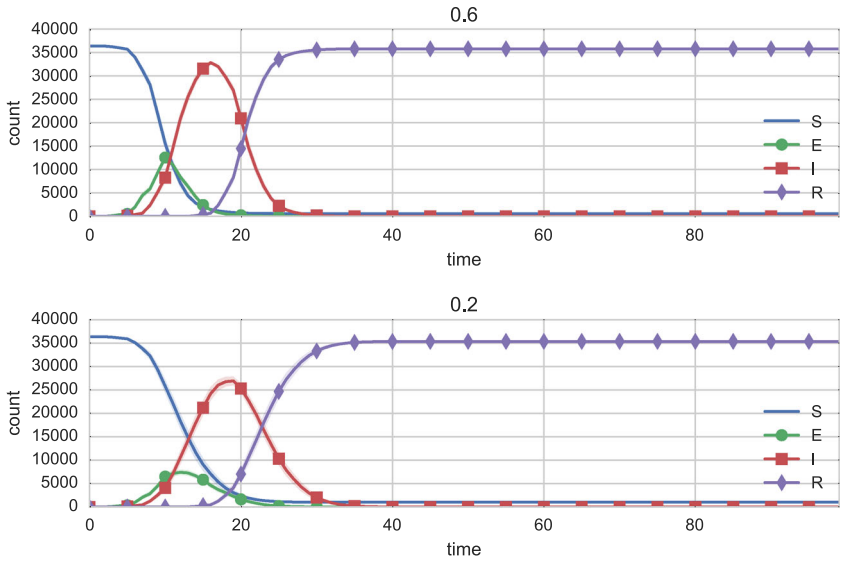


Fig. 11 Epidemic curves after Local Degree sparsification

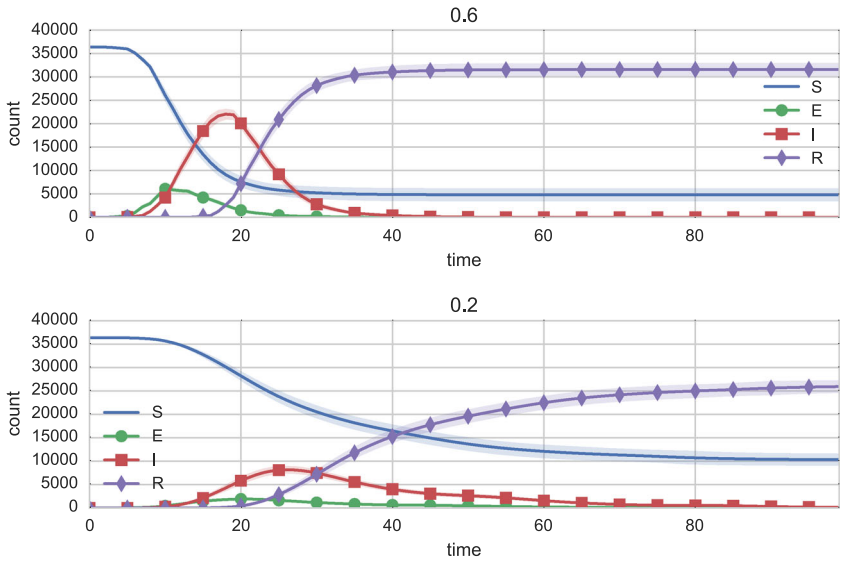


Fig. 12 Epidemic curves after algebraic distance sparsification

triangle and quadrangle counting, additionally we did not parallelize the latter. While the time complexity in O-notation of Edge Forest Fire is difficult to assess, it seems to be slightly faster than Simmelian Backbones.

6 Conclusion

Our experimental study on networks from Facebook, Twitter and Google+ as well as synthetically generated networks shows that several sparsification methods are capable of preserving a set of relevant properties of social networks when up to 80 % of edges have been removed.

Random edge deletion performs surprisingly well and retains a wide range of properties, but more targeted methods can perform even better. We propose local filtering as a generally applicable and computationally cheap post-processing step for edge sparsification methods that improves the preservation of almost all properties as it leads to a more equal rate of filtering across the network. Simmelian Backbones, Jaccard Similarity and algebraic distance prefer intra-cluster edges and thus do not keep global structures but with the added local filtering step they are able to enforce and retain a community structure as it was already shown for Jaccard Similarity. However, the preserved community structure is not necessarily the same as the one the Louvain algorithm finds. Our novel method, Local Degree, which is based on the notion that connections to hubs are highly important for the network's structure, in contrast preserves shortest paths and the overall connectivity of the network. This can be seen at the almost perfectly preserved diameter and the well-preserved

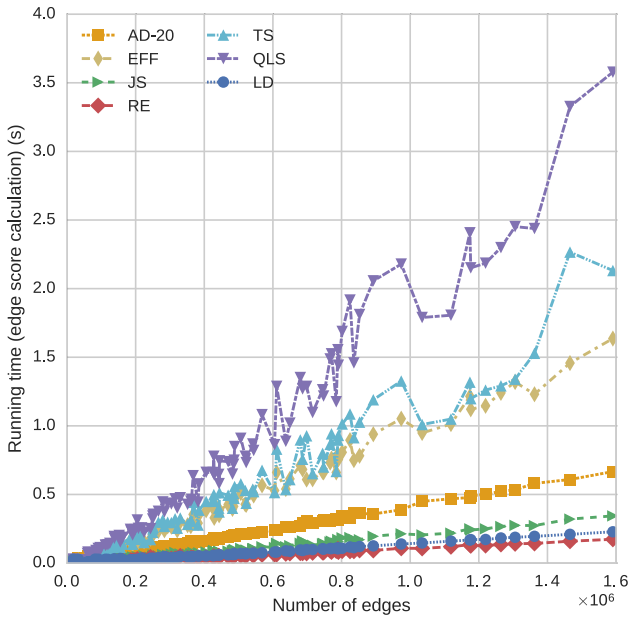


Fig. 13 Running times of various edge scoring methods on the Facebook networks

behavior of the network in epidemic simulations. Depending on the network, the Local Degree method is also able to preserve clustering coefficients and centralities. Our adaption of the Forest Fire sampling algorithm to edge scoring depends strongly on the specific network's structure. It is good at preserving connectivity, on some networks it also preserves centralities and the diameter.

We hope that the conceptual framework of edge scoring and filtering as well as our evaluation methods are steps towards a more unified perspective on a variety of related methods that have been proposed in different contexts. Future developments can be easily carried out within this framework and based on our implementations, which are available as part of the open-source network analysis package NetworKit.[2]

Many real-world social networks are not static, but evolve over time. We are not aware of any work that has adapted any of the presented methods for dynamic networks. An interesting direction for future work would thus be to adapt and compare sparsification methods for dynamic networks.

In terms of scalability and networks that do not fit into the main memory anymore, variants that work with external memory or in distributed settings could be developed. While basic building blocks such as triangle counting exist (Hu et al. 2014), the actual sparsification algorithms would need to be adapted.

[2] https://networkit.iti.kit.edu/.

Acknowledgments This work is partially supported by the German Research Foundation (DFG) under Grants ME 3619/3-1 and WA 654/22-1 within the Priority Programme 1736 *Algorithms for Big Data*.

Appendix 1: Properties of the used networks

Below you can find the number of nodes n, the number of edges m, the diameter D and the average local clustering coefficient Cc of all used social networks.

Network	n	m	m/n	D	Cc
American75	6386	217662	34.1	9	0.25
Amherst41	2235	90954	40.7	7	0.32
Auburn71	18448	973918	52.8	7	0.23
BC17	11509	486967	42.3	9	0.21
BU10	19700	637528	32.4	9	0.20
Baylor93	12803	679817	53.1	7	0.21
Berkeley13	22937	852444	37.2	7	0.21
Bingham82	10004	362894	36.3	8	0.23
Bowdoin47	2252	84387	37.5	6	0.29
Brandeis99	3898	137567	35.3	7	0.27
Brown11	8600	384526	44.7	9	0.22
Bucknell39	3826	158864	41.5	6	0.28
Cal65	11247	351358	31.2	8	0.23
Caltech36	769	16656	21.7	6	0.43
Carnegie49	6637	249967	37.7	8	0.29
Colgate88	3482	155043	44.5	6	0.27
Columbia2	11770	444333	37.8	9	0.24
Cornell5	18660	790777	42.4	8	0.23
Dartmouth6	7694	304076	39.5	8	0.25
Duke14	9895	506442	51.2	8	0.25
Emory27	7460	330014	44.2	8	0.26
FSU53	27737	1034802	37.3	8	0.22
GWU54	12193	469528	38.5	9	0.22
Georgetown15	9414	425638	45.2	11	0.23
Hamilton46	2314	96394	41.7	6	0.30
Harvard1	15126	824617	54.5	11	0.22
Haverford76	1446	59589	41.2	6	0.33
Howard90	4047	204850	50.6	7	0.23
Indiana69	29747	1305765	43.9	8	0.21
JMU79	14070	485564	34.5	7	0.20
Johns Hopkins55	5180	186586	36.0	8	0.28
Lehigh96	5075	198347	39.1	6	0.27
MIT8	6440	251252	39.0	8	0.28
MSU24	32375	1118774	34.6	8	0.21
MU78	15436	649449	42.1	7	0.22
Maine59	9069	243247	26.8	7	0.25
Maryland58	20871	744862	35.7	7	0.21
Mich67	3748	81903	21.9	7	0.29

Network	n	m	m/n	D	Cc
Michigan23	30147	1176516	39.0	9	0.22
Middlebury45	3075	124610	40.5	7	0.29
Mississippi66	10521	610911	58.1	7	0.25
NYU9	21679	715715	33.0	8	0.20
Northeastern19	13882	381934	27.5	9	0.21
Northwestern25	10567	488337	46.2	9	0.24
Notre Dame57	12155	541339	44.5	8	0.21
Oberlin44	2920	89912	30.8	7	0.27
Oklahoma97	17425	892528	51.2	9	0.23
Penn94	41554	1362229	32.8	8	0.22
Pepperdine86	3445	152007	44.1	9	0.29
Princeton12	6596	293320	44.5	9	0.24
Reed98	962	18812	19.6	6	0.33
Rice31	4087	184828	45.2	6	0.30
Rochester38	4563	161404	35.4	7	0.30
Rutgers89	24580	784602	31.9	8	0.23
Santa74	3578	151747	42.4	8	0.27
Simmons81	1518	32988	21.7	7	0.33
Smith60	2970	97133	32.7	7	0.29
Stanford3	11621	568330	48.9	9	0.25
Swarthmore42	1659	61050	36.8	6	0.30
Syracuse56	13653	543982	39.8	7	0.24
Temple83	13686	360795	26.4	8	0.22
Tennessee95	16979	770659	45.4	7	0.24
Texas80	31560	1219650	38.6	8	0.22
Texas84	36371	1590655	43.7	7	0.20
Trinity100	2613	111996	42.9	6	0.29
Tufts18	6682	249728	37.4	10	0.24
Tulane29	7752	283918	36.6	8	0.26
UC33	16808	522147	31.1	8	0.24
UC61	13746	442174	32.2	8	0.27
UC64	6833	155332	22.7	8	0.28
UCF52	14940	428989	28.7	8	0.24
UCLA26	20467	747613	36.5	8	0.22
UCSB37	14935	482224	32.3	8	0.23
UCSC68	8991	224584	25.0	8	0.24
UCSD34	14948	443221	29.7	9	0.23
UChicago30	6591	208103	31.6	10	0.26
UConn91	17212	604870	35.1	8	0.21
UF21	35123	1465660	41.7	8	0.22
UGA50	24389	1174057	48.1	8	0.21
UIllinois20	30809	1264428	41.0	8	0.22
UMass92	16516	519385	31.4	8	0.21
UNC28	18163	766800	42.2	7	0.21
UPenn7	14916	686501	46.0	9	0.22
USC35	17444	801853	46.0	9	0.22
USF51	13377	321214	24.0	8	0.24
USFCA72	2682	65252	24.3	7	0.28
UVA16	17196	789321	45.9	8	0.22
Vanderbilt48	8069	427832	53.0	7	0.26
Vassar85	3068	119161	38.8	8	0.25
Vermont70	7324	191221	26.1	7	0.24
Villanova62	7772	314989	40.5	7	0.24
Virginia63	21325	698178	32.7	9	0.22
Wake73	5372	279191	52.0	9	0.28
WashU32	7755	367541	47.4	8	0.27
Wellesley22	2970	94899	32.0	8	0.27
Wesleyan43	3593	138035	38.4	7	0.27
William77	6472	266378	41.2	8	0.26
Williams40	2790	112986	40.5	6	0.30
Wisconsin87	23842	835952	35.1	9	0.21
Yale4	8578	405450	47.3	9	0.24
com-lj	3997962	34681189	8.7	21	0.35
gplus	107614	12238285	113.7	6	0.52
Twitter	81306	1342296	16.5	7	0.60

References

Ahmed NK, Neville J, Kompella R (2014) Network sampling: from static to streaming graphs. ACM Trans Knowl Discov Data (TKDD) 8(2):7

Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286:509–512

Bastian M, Heymann S, Jacomy M (2009) Gephi: An open source software for exploring and manipulating networks. In: Adar E, Hurst M, Finin T, Glance NS, Nicolov N, Tseng BL (eds) ICWSM, The AAAI Press. http://dblp.uni-trier.de/db/conf/icwsm/icwsm2009.html#BastianHJ09

Batson J, Spielman DA, Srivastava N, Teng SH (2013) Spectral sparsification of graphs: theory and algorithms. Commun ACM 56(8):87–94

Borassi M, Crescenzi P, Habib M, Kosters WA, Marino A, Takes FW (2015) Fast diameter and radius bfs-based computation in (weakly connected) real-world graphs: with an application to the six degrees of separation games. Theor Comput Sci 586:59–80. doi:10.1016/j.tcs.2015.02.033, http://www.sciencedirect.com/science/article/pii/S0304397515001644 (fun with Algorithms)

Chen J, Safro I (2011) Algebraic distance on graphs. SIAM J Sci Comput 33(6):3468–3490

Chiba N, Nishizeki T (1985) Arboricity and subgraph listing algorithms. SIAM J Comput 14(1):210–223. doi:10.1137/0214017

Costa LdF, Oliveira ON Jr, Travieso G, Rodrigues FA, Villas Boas PR, Antiqueira L, Viana MP, Correa Rocha LE (2011)

Analyzing and modeling real-world phenomena with complex networks: a survey of applications. Adv Phys 60(3):329–412

Ebbes P, Huang Z, Rangaswamy A, Thadakamalla HP, Unit ORGB (2008) Sampling large-scale social networks: insights from simulated networks. In: 18th annual workshop on information technologies and systems, Paris, France, Citeseer

Fortunato S, Barthelemy M (2007) Resolution limit in community detection. Proc Natl Acad Sci 104(1):36–41

Fortunato S, Boguñá M, Flammini A, Menczer F (2008) Approximating pagerank from in-degree. In: Aiello W, Broder A, Janssen J, Milios E (eds) Algorithms and models for the web-graph, Springer, Berlin, pp 59–71

Geisberger R, Sanders P, Schultes D (2008) Better approximation of betweenness centrality. In: ALENEX, SIAM, pp 90–100

Gleiser PM, Danon L (2003) Community structure in jazz. Adv Complex Syst 6(4):565–574. http://dblp.uni-trier.de/db/journals/advcs/advcs6.html#GleiserD03

Hu X, Tao Y, Chung CW (2014) I/o-efficient algorithms on triangle listing and counting. ACM Trans Database Syst 39(4):1–27. doi:10.1145/2691190.2691193

Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193–218. doi:10.1007/BF01908075

John E, Safro I (2016) Single-and multi-level network sparsification by algebraic distance. arXiv:160105527 (**arXiv preprint**)

Keeling M, Rohani P (2008) Modeling infectious diseases in humans and animals. Princeton University Press, Princeton

Lancichinetti A, Fortunato S (2009a) Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Phys Rev E 80(1):016118. doi:10.1103/PhysRevE.80.016118

Lancichinetti A, Fortunato S (2009b) Community detection algorithms: a comparative analysis. Phys Rev E 80:056117. doi:10.1103/PhysRevE.80.056117

Leskovec J, Faloutsos C (2006) Sampling from large graphs. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, USA, KDD '06, pp 631–636. doi:10.1145/1150402.1150479

Leskovec J, Mcauley JJ (2012) Learning to discover social circles in ego networks. In: Pereira F, Burges C, Bottou L, Weinberger K (eds) Advances in neural information processing systems 25. Curran Associates Inc, New York, pp 539–547

Lindner G, Staudt CL, Hamann M, Meyerhenke H, Wagner D (2015) Structure-preserving sparsification of social networks. In: Pei J, Silvestri F, Tang J (eds) Proceedings of the 2015 IEEE/ACM international conference on advances in social networks analysis and mining, ASONAM 2015, Paris, France, August 25–28, 2015, ACM, pp 448–454. doi:10.1145/2808797.2809313

Newman M (2010) Networks: an introduction. Oxford University Press, Oxford

Nick B, Lee C, Cunningham P, Brandes U (2013) Simmelian backbones: Amplifying hidden homophily in facebook networks. In: Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining, ACM, New York, NY, USA, ASONAM '13, pp 525–532. doi:10.1145/2492517.2492569

Nocaj A, Ortmann M, Brandes U (2014) Untangling hairballs—from 3 to 14 degrees of separation. In: Duncan CA, Symvonis A (eds) Graph Drawing—22nd international symposium, GD 2014, Würzburg, Germany, September 24-26, 2014 Revised Selected Papers, Lecture Notes in Computer Science, vol 8871. Springer, Berlin, pp 101–112. doi:10.1007/978-3-662-45803-7_9

Ortmann M, Brandes U (2014) Triangle listing algorithms: Back from the diversion. In: McGeoch CC, Meyer U (eds) 2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2014, Portland, Oregon, USA, January 5, 2014, SIAM, pp 1–8. doi:10.1137/1.9781611973198.1

Page L, Brin S, Motwani R, Winograd T (1999) The pagerank citation ranking: bringing order to the web. Technical Report 1999-66, Stanford InfoLab. http://ilpubs.stanford.edu:8090/422/, previous number = SIDL-WP-1999-0120

Rosvall M, Axelsson D, Bergstrom CT (2009) The map equation. Euro Phys J Spec Topics 178(1):13–23. doi:10.1140/epjst/e2010-01179-1

Saha T, Rangwala H, Domeniconi C (2013) Sparsification and sampling of networks for collective classification. In: Greenberg AM, Kennedy WG, Bos ND (eds) Social Computing, Behavioral-cultural modeling and prediction. Springer, Berlin, pp 293–302

Salathé M, Kazandjieva M, Lee JW, Levis P, Feldman MW, Jones JH (2010) A high-resolution human contact network for infectious disease transmission. Proc Natl Acad Sci 107(51):22020–22025

Satuluri V, Parthasarathy S, Ruan Y (2011) Local graph sparsification for scalable clustering. In: Proceedings of the 2011 ACM SIGMOD international conference on management of data, ACM, New York, NY, USA, SIGMOD '11, pp 721–732. doi:10.1145/1989323.1989399

Serrano MÁ, Boguñá M, Vespignani A (2009) Extracting the multiscale backbone of complex weighted networks. Proc Natl Acad Sci 106(16):6483–6488. doi:10.1073/pnas.0808904106, http://www.pnas.org/content/106/16/6483.abstract

Shun J, Tangwongsan K (2015) Multicore triangle computations without tuning. In: Proceedings of the IEEE international conference on data engineering (ICDE). http://dblp.unitrier.de/rec/bibtex/conf/icde/ShunT15

Simmel G, Wolff K (1950) The sociology of Georg Simmel. Free Press paperback, Free Press. http://books.google.de/books?id=Ha2aBqS415YC

Staudt C, Meyerhenke H (2016) Engineering parallel algorithms for community detection in massive networks. IEEE Trans Parallel Distrib Syst 27(1):171–184. doi:10.1109/TPDS.2015.2390633

Staudt C, Sazonovs A, Meyerhenke H (2014) Networkit: a tool suite for large-scale complex network analysis. CoRR http://arxiv.org/abs/1403.3005, arXiv: abs/1403.3005

Traud AL, Mucha PJ, Porter MA (2012) Social structure of facebook networks. Phys A Stat Mech Appl 391(16):4165–4180

Vinh NX, Epps J, Bailey J (2009) Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: Proceedings of the 26th annual international conference on machine learning, ACM, New York, NY, USA, ICML '09, pp 1073–1080. doi:10.1145/1553374.1553511

Wang Y, Chakrabarti D, Wang C, Faloutsos C (2003) Epidemic spreading in real networks: an eigenvalue viewpoint. In: Proceedings of the 22nd international symposium on reliable distributed systems, 2003, IEEE, pp 25–34

Yang J, Leskovec J (2012) Defining and evaluating network communities based on ground-truth. In: Proceedings of the ACM SIGKDD workshop on mining data semantics, ACM, p 3