Motif Cut Sparsifiers

Michael Kapralov∗, Mikhail Makarov∗, Sandeep Silwal†, Christian Sohler‡ and Jakab Tardos∗

∗EPFL, Lausanne, Switzerland
†MIT, Cambridge, USA
‡University of Cologne, Cologne, Germany

Abstract—A motif is a frequently occurring subgraph of a
given directed or undirected graph $G$ (Milo et al.). Motifs
capture higher order organizational structure of $G$ beyond edge
relationships, and, therefore, have found wide applications such
as in graph clustering, community detection, and analysis of
biological and physical networks to name a few (Benson at al.,
Tsourakakis et al.). In these applications, the cut structure of
motifs plays a crucial role as vertices are partitioned into clusters
by cuts whose conductance is based on the number of instances
of a particular motif, as opposed to just the number of edges,
crossing the cuts.

In this paper, we introduce the concept of a motif cut sparsifier.
We show that one can compute in polynomial time a sparse
weighted subgraph $G'$ with only $O(n/\epsilon^2)$ edges such that for
every cut, the weighted number of copies of $M$ crossing the cut
in $G'$ is within a $1 + \epsilon$ factor of the number of copies of $M$
crossing the cut in $G$, for every constant size motif $M$.

Our work carefully combines the viewpoints of both graph
sparsification and hypergraph sparsification. We sample edges
which requires us to extend and strengthen the concept of
cut sparsifiers introduced in the seminal works of Karger and
Benczúr et al. to the motif setting. The task of adapting the
importance sampling framework common to efficient graph
sparsification algorithms to the motif setting turns out to be
nontrivial due to the fact that cut sizes in a random subgraph
of $G$ depend non-linearly on the sampled edges. To overcome
this, we adopt the viewpoint of hypergraph sparsification to
define edge sampling probabilities which are derived from the
strong connectivity values of a hypergraph whose hyperedges
represent motif instances. Finally, an iterative sparsification
primitive inspired by both viewpoints is used to reduce the
number of edges in $G$ to nearly linear.

In addition, we present a strong lower bound ruling out a sim-
ilar result for sparsification with respect to induced occurrences
of motifs.

I. INTRODUCTION

A motif is a (connected) subgraph of a given directed or
undirected graph $G = (V, E)$ that occurs more frequently than
one would typically assume in a random graph; it has been
observed empirically that motifs exist in many networks [1],
[2], [3], [4]. These higher order graph structures are crucial to
the organization of complex networks as they capture richer
structural information about the graph data and therefore carry
important information that can be exploited in network data
analysis. Indeed, in many application domains, such as in
clustering and social network analysis [5], [3], [6], [4], [7],
[8], [9], community detection [5], [3], [7], [4], [10], [11], [12],
and analysis of biological or physical networks [13], [14], [15],
[3], understanding higher order graph structures has become
increasingly important. See Section I-A for further details on
motif-based applications.

Graph clustering in particular is a prominent example where
clustering algorithms have been developed to exploit the motifs
structure of graphs [3], [4]. These algorithms first compute a
motif weighted graph where every edge is weighted by the
number of copies of a given motif it is contained in, and
then apply spectral clustering on this motif weighted graph
(see Section I-A for more details). Such an approach may be
viewed as partitioning the vertex set of a graph into subsets
(called clusters) with high internal motif connectivity and low
motif connectivity between the clusters.

Graph sparsification is an algorithmic technique for speed-
ning up cut based graph algorithms that was introduced in the
seminal work of [16] and [17], with powerful generalization
to spectral sparsifiers obtained in [18]. The main idea behind
graph cut sparsification is to design a sparse weighted graph
that approximates the cuts in the original graph to within a $1 \pm \epsilon$
factor for small $\epsilon \in (0, 1)$. Cut sparsifiers with $O(n/\epsilon^2)$ edges
that approximate all cuts in $G$ have been constructed, with
some constructions achieving an $O(n/\epsilon^1)$ upper bound on the
number of edges in nearly linear time [19]. The related concept
of hypergraph sparsification has received a lot of attention
in the literature recently, with nearly optimal size sparsifiers
obtained in [20]. In this paper we ask whether it is possible
to sparsify a graph while preserving the motif cut structure:

Given an arbitrary input graph $G$, is it possible to compute a
sparse weighted graph $G'$ (a motif cut sparsifier) that
approximates the motif cut structure of $G$?

Before we discuss how motif sparsification compares to
graph and hypergraph sparsification, we first informally state
our definition of a motif cut sparsifier. The main idea is very
intuitive: a motif cut sparsifier approximates the number of
motifs that cross a cut for every cut in the graph. In order to
utilize sparse graphs, edges need to be weighted and we must
define the weighted number of motifs crossing a cut. Here we
follow the standard interpretation of integer edge weights as
dge multiplicities, and therefore, define the motif weight as
the product of its edge weights (which under the previous
The interpretation is simply the number of distinct unweighted motifs crossing the cut. The definition generalizes to non-integral edge weights in a straightforward manner.

**Definition 1** (Motif cut sparsifier; informal). For a connected motif \( M \) and \( \epsilon \in (0,1) \) we say that a (possibly directed) weighted graph \( G' = (V,E) \) is an \( \epsilon \)-motif-sparsifier of \( G \) with respect to \( M \) if for every \( \emptyset \neq S \subset V \) the weighted number of copies of \( M \) in \( G \) crossing the cut \( (S, V \setminus S) \) is \((1 \pm \epsilon)\)-close to the number of copies of \( M \) crossing the same cut in \( G' \).

There is no consensus in the literature on whether these "copies" should be induced subgraphs of \( G \) or arbitrary subgraphs – both seem to be useful concepts in applications. We consider both cases, and it turns out there is a fundamental difference between them: In the case of non-induced motifs powerful and small motif-cut sparsifiers can be constructed for any graph \( G \) (as we’ll see below) while in the case of induced motifs this is not possible. Hence, below we focus on the non-induced case, and we state our result for the induced case at the end of the section.

**a) Motif sparsifiers vs hypergraph sparsifiers:** It may seem at first sight that one can easily compute a motif cut sparsifier by first computing a motif hypergraph that contains an edge for every motif, and then by sparsifying this hypergraph. The idea with this approach is that although there exists a corresponding motif hypergraph for every graph and every motif (at least when we allow parallel hyperedges), the converse is not true. Thus, while we can compute a motif hypergraph sparsifier, we do not know how to transform it back into a graph while maintaining the fact that the number of motifs crossing every cut is preserved. Similar issues arise if we first sparsify a motif weighted graph. This is illustrated in Figure 2 and detailed in Section II.

Indeed, motif sparsifiers are quite different from graph and hypergraph cut sparsifiers. For example, graph and hypergraph cut sparsifiers have the property that when \( G' = (V,E') \) is a sparsifier of \( G(V,E) \) and \( H' = (V,F') \) is a sparsifier for \( H(V,F) \) then \( E' \setminus F' \) is a sparsifier for \( (V,E \cup F) \). This property can, for example, be used to obtain a semi-streaming algorithm for many cut problems using \( O(n \cdot \text{poly}(\log n)) \) space [21], [22], [23], [24], [25], [26].

Unfortunately, motif sparsifiers in general do not have this property. Furthermore, even for a small motif like a triangle, it is not possible to compute a motif sparsifier in the semi-streaming model. This is because even counting the number of triangles in a stream can require \( \Omega(|E|) \) space for \( |E| = \Omega(n^2) \) [27] and computing a motif sparsifier, in particular when the motif is a triangle, easily allows us to recover the global triangle count by querying the sparsifier on the \( n \) singleton cuts.

**b) Importance sampling:** A common approach to different graph and hypergraph sparsification algorithms (see [19], [17], [28], [29], [30], [31], [32] and references within) is to define a sampling probability \( p(e) \) and a weight \( w(e) \) for each edge \( e \) and then sample each edge independently with probability \( p(e) \). If \( e \) is sampled, it is also assigned weight \( w(e) \); for appropriately defined probabilities and weights, the resulting graph is a sparsifier with a near linear number of edges.

For motif sparsifiers, such an approach **cannot** yield a cut sparsifier of near linear size, as the example of a clique on \( n \) vertices with the motif being a triangle shows. Indeed, if we sample every edge with probability \( o(1/n^{2/3}) \), then the expected number of triangles incident to a given vertex is \( o(1) \). Then, it is straightforward to show that the resulting graph is typically not a triangle sparsifier. However, for a sampling probability of \( \Omega(1/n^{2/3}) \), the expected number of sampled edges is \( \Omega(n^{2/3}) \), i.e. the resulting graph does not have near linear size. Since a clique is also completely symmetric, it is unclear how one could assign different probabilities to each edge. However, there is still a simple argument that a sampling probability of roughly \( p = \log n/n^{2/3} \) results in a sparsifier such that w.h.p. no vertex is incident to more than \( \log^{O(1)} n \) distinct triangles. Since every triangle has three edges, this implies that there are only \( n \log^{O(1)} n \) edges that are involved in a triangle. Thus, removing the remaining edges yields a triangle sparsifier of near linear size.

While our construction still samples every edge with the same probability, in the special case of a clique, we can only obtain a sparsifier if we remove most of the unused edges in a cleaning step. It is unclear whether such an approach generalizes to other less structured graphs and motifs. Nevertheless, the main result of this paper is that there does exist an algorithm producing a motif sparsifier of nearly linear size from an arbitrary input graph:

**Theorem 2** (follows from Corollary 12 and Theorem 13 in Section IV). For every graph \( G = (V,E), |V| = n, \) every constant integer \( r \geq 2, \) and \( \epsilon \in (0,1) \), there exists an \( \epsilon \)-motif sparsifier \( G' \) of \( G \) with respect to all connected motifs \( M \) of with at most \( r \) vertices simultaneously that contains \( \tilde{O}(n/\epsilon^2) \) edges.

Furthermore, there is an algorithm which outputs a \( G' \) which is an \( \epsilon \)-motif sparsifier with high probability. Its running time is \( \tilde{O}(\min(T(r), n^{64/r^3})) \), where \( T(r) \) is the time needed to enumerate all of the motif instances and \( n^r \) is the matrix multiplication time.

Note that the resulting graph \( G' \) is automatically a cut sparsifier of \( G \), as an \( M \)-sparsifier is exactly a cut sparsifier when \( M \) is a single edge. Beyond that, however, \( G' \) approximately preserves the sizes of all motif cuts in \( G \) with respect to constant size motifs. Theorem 2 also applies to directed graphs.

The running time – \( \tilde{O}(n^{64/r^3}) \) in particular – is sublinear in the number of motif instances in some settings. This shows a clear advantage of motif sparsification over simply sparsifying the motif hypergraph, which would take time at least proportional to the number of hyperedges (i.e. motif instances).

**c) Induced Motifs:** In the final section of the paper we consider the setting where we require motif instances to be induced subgraphs of input graph \( G \). This is also a natural
definition of motifs which likewise has been extensively studied in literature; see [33], [4], [34], [35] and the references within. We show that no analogue of Theorem 2 exists in this setting. Even for constant size motifs we can construct an example where any non-trivial sparsification is impossible.

**Theorem 3** (Informal version of Theorem 14). There exists a graph \( G = (V, E) \) on \( n \) vertices and a motif of constant size such that it is impossible to approximate the induced-motif-cut structure of \( G \) to within a multiplicative error of \((1 \pm \epsilon)\) for \( \epsilon \leq 1/500 \) using a (non-negative) weighted graph with \( o(n^2) \) edges.

A. Related Work

As stated in the introduction, motifs have been widely adopted for study of higher order networks due to their ubiquitous presence [1], [2], [3]. Since the network literature concerning motifs is too vast to properly summarize, we mainly focus on algorithms and applications of motifs and higher order structures. Note that a majority of the papers we reference are application oriented papers; relatively few works offer strong theoretical guarantees.

Applications where motif analysis has become impactful include graph clustering (both local and global clustering) [5], [3], [6], [4], [9] and community detection [5], [3], [7], [4], [10], [11], [12]. These applications are based on exploiting the motif-cut structure of a given graph. For example in works such as [3], [7], [4], various alternative notions of conductance are introduced which take into account the influence of motifs. In particular, the definition of conductance is redefined in terms of the number of motifs, for example triangles, crossing the cut. Therefore, one direct application of our results is to provide solid theoretical understanding of motif-based cut structure via graph sparsification.

In graph and network data visualization, it has been empirically observed that motif based embeddings provide more meaningful low-dimensional representations over their counterparts which do not employ motifs, such as spectral embeddings [36], [37]. Indeed, [37] shows that performing spectral embeddings on adjacency matrices which are motif based, for example using matrices which are weighted sums of higher powers of the adjacency matrix, leads to better inductive bias as these presentations better capture the rich underlying community or cluster structures; see the visualizations given in [36], [37].

In graph classification, motifs have provided more meaningful characterizations for graphs at both micro (local) and macro (global) scales [38]. Motifs have also become popular in the related area of learning on graphs which has further downstream applications such as recommender systems, fraud detection, and protein identification [39], [40], [41]. Additional applications of motif-based graph learning include link prediction [42], [43], [44] and computing network-based node rankings [45], [46]. Indeed in the active area of graph neural networks, motif counts are an extremely popular feature augmentation technique as graph neural networks often struggle to identify motifs and higher order structures [46], [47], [48]. Lastly, there has also been empirical and theoretical work on efficiently counting motifs and summarizing motif statistics. This literature is also quite vast but an excellent reference is the tutorial [49] given at the WWW 2019 conference.

Note that which motifs are important for a given complex network strongly depends on the underlying network properties [1], [13], [3]. One of the most fundamental and well studied motifs is the triangle and its directed variants [50], [5], [3], [4], [11]. Indeed, some of the work closest to ours concerns triangle motifs.

Objects close to triangle sparsifiers, which we precisely define and give theoretical guarantees in our work, have also been studied [50], [12]. The main difference is that in [50], their goal is to acquire a sparse subgraph which only preserves the global triangle count; in contrast, our task is much more difficult as we wish to preserve the triangle counts (and arbitrary motif counts) for all cuts simultaneously. Note that preserving motif cut values automatically implies preservation of the global number of triangles by querying \( n \) singleton cuts. Furthermore, [50] employ a one-shot uniform sampling of the edges whereas we use careful importance-based sampling based on edge importance over multiple rounds. Similarly in [12], their goal is to get a sparsifier with respect to edges which has better space bounds for graphs containing many triangles. Our work achieves nearly linear space bounds for preserving motifs cuts for arbitrary motifs.

a) Clique enumeration results: Our first algorithm makes use of a primitive that enumerates all of the instances of a given motif. Unfortunately in general, this can take time exponential in the size of the motif, since even deciding if certain motifs are contained in a graph, such as a clique, is NP-complete [51].

The clique enumeration problem is one of the most studied motif enumeration problems. The most notable results here include [52], giving an algorithm working in time \( O(r\alpha(G)r^{-2}m) \), where \( \alpha(G) \) is the arboricity of the graph \( G \) for enumerating all cliques of size \( r \). By utilizing the bound \( \alpha(G) \leq m^{1/2} \) for connected graphs from the same paper, this yields an \( O(n^{r/2}) \) time algorithm for a general graph.

There are also works which achieve faster runtimes for graph enumeration for subgraphs with special structures, such as planar graphs or bounded tree-width graphs [53], and bounded arboricity graphs [52]. Lastly, see [54] and references within for a survey on applied algorithms for subgraph enumeration.

II. TECHNICAL OVERVIEW

We illustrate our main algorithmic ideas by considering a simple example, namely when \( G = (V, E) \) is an undirected unweighted graph and the motif \( M \) is a triangle \( \Delta = (V_\Delta, E_\Delta) \), i.e. a clique on three vertices. Our approach is inspired by the techniques introduced by Karger [16] and Benczur and Karger [17] in the context of sparsification of undirected graphs. We recall these techniques now, then show why their immediate extension fails, and finally present our algorithm.
We start by recalling Karger’s cut sampling bound and its application to graph sparsification. Karger [16] shows that in a graph \( G \) with min-cut \( k \), the number of cuts of size \( \alpha k \) for \( \alpha \geq 1 \) is bounded by \( \binom{n}{2} \alpha \). The bound is then applied to show that a sample of edges of \( G \) which contains every edge independently with probability \( p = \min\left\{ \frac{C\log n}{k^2}, 1 \right\} \) (with weight \( 1/p \)) is an \( \epsilon \)-cut sparsifier, i.e., preserves all cuts up to multiplicative precision \( 1 \pm \epsilon \), with high probability. The latter claim follows by noting that the probability that a cut of size \( \alpha k \) is not appropriately preserved is exponential in \( \epsilon^2 \frac{\alpha k}{4} \geq \Omega \frac{1}{n} \), which suffices for the union bound. This uniform sampling approach leads to a reduction in the number of edges when the min-cut \( k \) in \( G \) is large. In the general case [17] show that sampling edges with probabilities proportional to the inverse of their strong connectivity and reweighting appropriately leads to a cut sparsifier with high probability. Here the strong connectivity \( k_e \) of an edge \( e \) is equal to the maximum \( k \) such that there exists a vertex induced subgraph \( C \) of \( G \) containing \( e \) such that the size of its minimum cut in \( C \) is at least \( k \).

In what follows we discuss two natural approaches to using these techniques to obtain motif sparsifiers, explain some of the issues with them, and then outline our approach. The first approach is based on a hypergraph version of motifs and the second one is based on graphs. In the following discussion we assume for simplicity that the input graph \( G \) is undirected and unweighted and the motif \( M \) is a triangle.

a) Shortcomings of motif sparsification based on hypergraphs: As already mentioned in the introduction one can compute a motif hypergraph by creating a hyperedge for every motif. We could then simply use hypergraph cut sparsification algorithms, such as [55] or [20]. Although in general, we cannot transform a sparsified hypergraph back into a graph, we could still try to adapt some hypergraph sparsification techniques to our problem. For example, we could sample all edges of a motif whenever its corresponding hyperedge gets picked. To give a concrete example, in the case of triangle motifs, we may first find all triangles in the input graph, select some of them and then construct a new graph, containing only the selected triangles with some edge re-weightings. This would be a way to simulate some hypergraph sparsification approaches. However, it is easy to see that some of the discarded triangles might appear again. For example, consider a case of the graph on Figure 1: if you take only triangles 1, 2 and 3 and reconstruct the graph, the final graph will still contain triangle 4. Therefore, we cannot hope to directly transform hypergraph sparsification approaches into motif sparsifiers.

b) Shortcomings of motif sparsification based on a triangle-weighted graph: A natural way to apply Karger’s approach to our motif sparsification problem (or triangle sparsification in the following discussion) is to use it on the triangle weighted graph \( G_\Delta = (V, E, w_\Delta) \), where \( w_\Delta(e) \) is the number of triangles containing edge \( e \) that has been used in the context of graph clustering [3], [4]. Indeed, triangle weighted graphs have the useful property that the size of the cut \( (S, V \setminus S) \) in \( G_\Delta \) is exactly twice the number of triangles that cross this cut in \( G \). Therefore, if we were to sparsify \( G \) to \( G' \) in such a way that \( G'_\Delta \) is a cut sparsifier of \( G_\Delta \), \( G' \) would be a motif cut sparsifier of \( G \). It is a seemingly natural approach to try to use triangle weighted graphs to obtain triangle sparsifiers. However, we will now show in a series of examples that a number of simple approaches which use the triangle weighted graph fail.

A naive approach using the triangle weighted graph would be to sparsify the triangle weighted graph, and then construct \( G'_\Delta \) by taking the remaining edges in \( G'_\Delta \) with some weights. However, this does not work, as a situation could easily arise where all of the triangles in some cuts are deleted. Consider the example in Figure 2. Here, \( G'_\Delta \) is clearly a 1/2-cut sparsifier of \( G_\Delta \), but since it contains no triangles, no motif sparsifier of \( G \) can be constructed from it without adding new edges.

A better approach is to apply Karger’s cut counting bound to the triangle weighted graph \( G_\Delta \) and use it to prove that...
an appropriate random sample of edges of $G$, denoted by $G'$, will satisfy
\[ G_\Delta \approx_e G'_\Delta. \] (1)

First, in order to make this approach work, we need to assume that $G_\Delta$ is $k$-connected for some reasonably large $k$. We make this assumption now to illustrate the challenges that arise even in this special case. Following [16], we could sample each edge with probability $\approx \frac{\log n}{k}$. That would unfortunately lead to each triangle staying in the graph with probability $\approx \frac{\log n}{k}$ only, and in particular some vertices may end up participating in no triangles in the sample with high probability. The latter means that the corresponding singleton cuts in $G'_\Delta$ would be empty, and (1) would certainly not be satisfied. Naturally, we can also try to sample each edge with a lower probability, say $\approx \frac{\log n}{k^2}$, but in this case the number of edges in the sparsifier of a $k$-regular graph would be $\approx k^{1/3}n \log n$, which is in superlinear in $n$.

In general, the above attempts point to the fact that edge weights in $G_\Delta$ are a non-linear function of the random variables that govern the presence or absence of various edges in $G'$, making ‘one-shot’ sparsification not easy to achieve.

c) Essential problem of triangle-weighted graph: Although we have already outlined several problems that we encounter in our attempts to construct a sparsifier using the triangle-weighted graph, there is another fundamental problem which arises directly from its structure as the following example demonstrates.

Let the graph $G$ (see Figure 3) consist of a clique on vertices in $V(G) \setminus \{v_1, \ldots, v_l\}$ where $l = \lceil \sqrt{n} \rceil$, and let $h \leq \sqrt{n}/4$ be an integer. For $i \in [l]$, let vertex $v_i$ be connected with vertices $u_{i,1}, \ldots, u_{i,l}$ such that the sets of neighboring vertices of $v_1, \ldots, v_l$ don’t intersect. Notice that the subgraph induced by $V(G) \setminus \{v_1, \ldots, v_l\}$ has connectivity in $G_\Delta$ of at least $n(n - 1)/8$, forming a connectivity component, while vertices $v_1, \ldots, v_l$ are not a part of this component because they are only connected to the clique with at most $O(n)$ triangles each.

In this situation, the triangles $v_i u_{i,j i}, u_{i,j2}$ for $i \in [l]$, $j_1, j_2 \in [h]$, and $j_1 \neq j_2$ are “dangling”, i.e. one of their edges is part of a component with a high connectivity, while there is no such component containing the whole triangle.

We know from the first part of the introduction that there is a way to get a clique sparsifier with almost linear number of edges, and graph $G$ is a clique with additional $O(\sqrt{n})$ vertices and $O(n)$ edges. Since this is an insignificant part of the whole graph, one might think that it is still easy to get a sparsifier with almost linear number of edges, for example by taking all edges $v_i u_{i,j}$ with probability 1, and sampling the clique as we did before. But we will now show that additional caution must be taken to handle the “dangling” triangles.

First, suppose that we sample all of the edges in the clique with the probability $\leq 1/2$. Consider the case $h = 2$. Then for all $i \in [l]$, edge $u_{i,1} u_{i,2}$ is contained in the only triangle in the cut $(\{v_i\}, V(G) \setminus \{v_i\})$. With high probability, at least one of those cuts will have motif size 0 and therefore, the resulting graph would not be motif sparsifier.

On the other hand, suppose that we were to take all of the edges $u_{i,j1} u_{i,j2}$ with probability 1. Consider the case of $h = \lceil \sqrt{n}/4 \rceil$. Then, the number of those edges would be $O(n^{3/2})$, and the sparsification would not produce any significant results. This shows that to take care of “dangling” triangles, we would need to sample the edges in them with different probabilities according to the situation at hand.

Under closer examination, one might discover that this problem stems from the following fact: consider a connectivity component $C$ in $G_\Delta$. If we were to take an induced subgraph of $G$ on vertices of $C$ and then build a triangle-weighted graph for it, the connectivity of this new triangle-weighted graph would most likely be lower than the connectivity of $C$.

As the above examples show, approaching motif sparsification purely from the point of view of sparsification of motif weighted graphs is difficult. Instead, we show, somewhat surprisingly, that a judicious composition of graph and hypergraph sparsification methods leads to a very clean approach, which we describe next. After that, we demonstrate that motif weighted graph can still be used in the proposed framework to achieve a speed-up in running time for dense graphs.

A. Strength-based sparsification

As we have discussed, it seems that we can neither use hypergraph nor graph sparsification ideas directly to obtain motif sparsifiers. The reason for this is probably that a motif is an object that — similarly to a hyperedge — usually lives on sets of more than 2 vertices, but at the same time is composed of edges, i.e. it is closely related to graphs. As a consequence motif sparsification may be viewed as an intermediate problem between hypergraph and graph sparsification.

Indeed, our main contribution is to properly combine ideas from graph and hypergraph sparsification and to overcome some motif specific technical obstacles. Our starting point will be to extend the notion of strong connectivity that is an important ingredient to many sparsification approaches (see, for example, [17], [56]) to the realm of motifs. Here we follow the hypergraph view and conceptually treat motifs as hyperedges. This way we can immediately extend the notion of connected components in hypergraphs [56] to motifs by saying that a $k$-connected component is a maximal induced subgraph such that every cut is crossed by at least $k$ motif instances. This will allow us to define for each motif its importance as a measure of the amount it contributes to various cuts in the graph. The hypergraph view of motifs will also supply us with hypergraph cut counting arguments from [29] that can be easily transferred to motif cuts and that will be useful for the analysis.

Once we have the definition of motif importance, it will be beneficial to switch to a graph-based view and think about how to compute the sparsifier. Our approach will be to sample edges but — similarly to earlier work in graph sparsification — we now need to identify important edges that we cannot miss for sparsification. In order to do so, we define the importance weight of an edge as the sum of the importance weights of its containing motifs. Edges whose importance weight is above a
certain threshold will always be kept as sampling them would result in a variance that is too high.

For the remaining edges, we want to apply a sampling approach. Here, there are two more challenges. First, we need to deal with the non-linear behaviour of motif cut sizes and then we also need to address the fact that a motif is composed of several edges, which means that the events that two intersecting motifs are sampled is not independent, which means that we cannot use Chernoff bounds that are often used in the analysis of other sparsifying constructions. To deal with the non-linearity we observe that sparsifying by a constant factor is still possible and so we iteratively sparsify the graph $O(\log n)$ times by a constant. To deal with the dependencies in the sampling process and prove concentration, we use Azuma’s inequality. During the different stages, edges that are no longer contained in any motif will receive a weight of 0 and will then be dropped.

Finally, we observe that except for the sets of critical edges, all edges are sampled with the same probability and so we can use our approach to compute a sparsifier that works simultaneously for a set of motifs.

B. Connectivity-based sparsification

A major drawback of the strength-based algorithm is the need to enumerate all instances of a given motif. This task is hard, since enumeration takes time that is at least proportional to the number of motif instances, which in dense graph ($|E| = O(n^2)$) can easily reach $O(n^r)$.

However, the motif cut sparsification task doesn’t implicitly require enumerating all of the motifs, and we show that by modifying an algorithm for exact subgraph counting [57], we can achieve sparsification in time $\tilde{O}(n^{r-1/3})$, which is sublinear to the number of motifs in dense graphs.

The key idea is to move away from using the importances based on motif strengths to importances based on motif connectivities, where the connectivity of a motif instance is the minimal motif size of a cut crossing this instance. This new measure of importance can be approximated without needing to enumerate all motifs, which leads to the faster (in some settings) running time of our second algorithm.

In more detail, we adopt the sparsification approach of [32] for our setting. A key object here is the motif weighted graph $G_M = (V, E, w_M)$, where, similarly to the triangle weighted graph, each edge $e$ is reweighted to $w_M(e)$ – the sum of weights of motifs containing $e$. The main challenge is the approximation of motif-connectivity-based edge importance. This is done in two steps. First, we show that the connectivity of a motif instance is multiplicatively approximated by the minimum of motif connectivities of all edges in this instance, where the motif connectivity of an edge is the minimal size of a motif cut crossing this edge. Then, by dividing the graph into several layers, we are able to approximate the minimum-motif-connectivity-of-an-edge-based importance for each layer, and then we combine to get the final approximation.

The rest of the algorithm works in the same way as the first one, however we also use a result by [58] to compute all-pairs connectivities in $\tilde{O}(n^2)$ time. Our algorithm requires the motif connectivities of edges to be computed with multiplicative precision, which existing subquadratic approximation algorithms cannot deliver.

C. Overview of Lower Bound

We also study the feasibility of producing a motif-cut sparsifier, similar to the one guaranteed by Theorem 2, in the setting where motif instances are required to be induced subgraphs. The main difficulty in attempting to sparsify induced motifs is that the act of removing edges from $G$ may result in new motif instances being created. This is not something we had to worry about in the proof of Theorem 2, and we could simply focus on preserving important motif instances that already existed in the original graph.

In fact, this difference turns out to result in a fundamental barrier, and we are able to show that any non-trivial sparsification may be impossible even for a motif as simple as the undirected 2-path (see Theorem 3).

In our lower bound construction, the input graph will be the undirected, unweighted clique with the three edges of a specific triangle $(a,b,c)$ removed. More formally, we define $\Delta^- = (V, E^-)$ as an unweighted, undirected graph on $n$ vertices, where

$$E^- = \binom{V}{2} \setminus \{ \{a,b\}, \{b,c\}, \{c,a\} \},$$

for distinct special vertices $a, b, c \in V$.

Note that while our Graph $\Delta^-$ is dense, the number of induced motifs is small, and each motif is of constant size. In the
case of the non-induced motif sparsification, this setting would be trivial, as we could simply keep all edges contributing to any motif, thereby sparsifying the graph, and retaining the exact cut structure. In the case of induced motifs, however, this doesn’t work, since removing edges may introduce additional motifs – as it would in this example.

Specifically, the number of induced motif instances of the 2-path motif is exactly 3(n – 3), with each motif instance containing 2 of the special vertices. We essentially prove that any graph \( \bar{G} \) that would sparsify \( \Delta^- \) should have (at least some of) these same 2-path motifs present. As it does in \( \Delta^- \), this would result in a very large (quadratic) number of not-necessarily-induced 2-paths in \( \bar{G} \). In order to insure that these aren’t induced (and hence don’t count as motif instances) \( \bar{G} \) must be dense.

**Example 1.** We give a slightly simpler – but ultimately incorrect version of our above lower-bound construction for intuition. Consider the unweighted clique, with a single edge \( (u,v) \) removed. More formally, \( G = (V,E) \) is an unweighted undirected graph on \( n \) vertices with

\[
E = \binom{V}{2} \setminus \{(u,v)\}.
\]

Attempting to sparsify this for the induced 2-path motif, we can observe some of the same things as we do in our lower bound construction: Even though \( G \) contains only a small, \( \Theta(n) \), number of motifs, one cannot simply sparsify it by removing all edges that contribute to no motifs. The act of removing edges can create new induced 2-paths, and we end up with a sparse graph whose induced-motif-cut structure doesn’t resemble that of \( G \) at all.

In fact, one can prove (in a similar manner to the proof of Theorem 14) that no reweighted subgraph of \( G \) approximates its induced-motif-cut structure, for some small constant \( \epsilon \). Surprisingly however, there does exist a weighted graph which achieves an arbitrarily close estimation: Let \( \bar{G} = (V,\bar{E},w) \) consist of the edge \( \{u,v\} \) with weight \( n^2 \), and the \( n-2 \) edges in \( u \times (V \setminus \{u,v\}) \), each with weight \( n^{-2} \). (This specifically gives a \((1 \pm n^{-3})\)-sparsifier, but the approximation can be arbitrarily improved by reweighting.) We leave the verification of the validity of this sparsifier to the reader.

**III. Preliminaries**

Let \( G = (V,E,w) \) be a directed weighted graph with vertex set \( V = \{1,\ldots,n\} \) and edge set \( E \subseteq V \times V \), \( m := |E| \). We will assume that the edge weights are always positive. Denote by \( W = \max_{e \in E} w(e)/\min_{e \in E} w(e) \). In this paper, we study the connectivity structure of higher order patterns in the graph. More precisely, we consider a given directed graph \( M = (V_M,E_M) \) which we assume to be a frequently occurring subgraph of \( G \) and which we refer to as a network motif or motif for short [1]. While the idea behind motifs is that they are more frequently occurring than what one would expect in a random graph [1], we are not making any formal assumption of this kind during the paper. Still, our motivation is that the motifs are common subgraphs. We will always assume that motifs are weakly connected, i.e. the undirected version of the motif is connected. We make this assumption since we are interested in graph cuts; there is no convincing definition of a motif cut for motifs that have more than one weakly connected component. Formally, we define motifs as follows.

**Definition 4 (Motifs and Motif Instances).** Let \( M = (V_M,E_M) \) be a weakly connected directed graph which we refer to as a motif. A subgraph of \( G \) that is isomorphic to \( M \) is called an instance of motif \( M \) in \( G \).

The definition of motifs extends to undirected graphs in a straightforward way by encoding undirected edges as two directed edges².

We will be interested in weighted graphs and therefore require a definition of weights of motif instances. In order to obtain such a definition, we first consider integer weighted graphs. A common interpretation of such graphs is that they can be viewed as unweighted multigraphs in which the multiplicity of each edge equals its weight. This view can be immediately generalized to define the weight of a motif of integer weighted graphs. We simply think of replacing every weighted edge by a corresponding number of copies and then count the number of distinct motifs. That is, the weight of a motif becomes the product of its edge weights. The extension to real non-negative weighted edges is straightforward.

**Definition 5 (Weight of Motif Instance).** Let \( G = (V,E,w) \) be a directed weighted graph. The weight \( w(I) \) of a motif instance \( I = (V_I,E_I) \) in \( G \) is defined as

\[
w(I) = \prod_{e \in E_I} w(e).
\]

Let \( (S,V \setminus S) \) be a cut in \( G \). We say that motif instance \( I = (V_I,E_I) \) crosses this cut if one of its edges crosses this undirected cut. Since the motifs are weakly connected by definition, this is equivalent to \( V_I \cap S \neq \emptyset \) and \( V_I \cap (S \setminus V) \neq \emptyset \).

**Definition 6 (Motif Size of a Cut).** Let \( G = (V,E,w) \) be a directed weighted graph. For a motif \( M \) the \( M \)-motif size of cut \( (S,V \setminus S) \) is defined as

\[
\text{Val}_{M,G}(S,V \setminus S) = \sum_{I \in M(G,M): I \text{ crosses } (S,V \setminus S)} w(I).
\]

Note that the previous definition is directly influenced by applied works such as [3], [7], and 4 which also redefine the cut size in terms of the number of motifs crossing a cut.

Our goal is to construct an algorithm for sparsifying a graph in such a way that the motif sizes of all cuts are \((1 \pm \epsilon)\) preserved. We formalize this notion as follows.

**Definition 7.** Let \( M = (V_M,E_M) \) be a motif and let \( G = (V,E,w) \) be a directed weighted graph. A directed weighted

²Note that if the graph is weighted, the weight assigned to the two resulting edges should be equal to the square root of the weight of the original edge. This is because in Definition 5 weight of a motif instance will be defined as the product of its edge weights.
graph $\hat{G}$ is an $(M, \epsilon)$-motif cut sparsifier of $G$, if for every cut $(S, V \setminus S)$, the following holds:

\[ (1 - \epsilon) \text{Val}_{M,G}(S, V \setminus S) \leq \text{Val}_{M,\hat{G}}(S, V \setminus S) \leq (1 + \epsilon) \text{Val}_{M,G}(S, V \setminus S). \]

A. Strong Motif Connectivity

We now extend the notion of strong connectivity used in graph cut sparsification [17] to motifs. For a given motif $M$ we will define the concepts of strong $M$-connectivity as well as $M$-connected components, which both follow naturally from the standard notion of strong connectivity. Our notion is also closely related to strong connectivity in hypergraphs [29], if we view a motif as a hyperedge. The main difference is that motifs are composed of simpler objects, i.e., edges. Similarly to the case of graphs and hypergraphs, strong motif connectivity will allow us to get bounds on the number of distinct cuts that we need to consider in the analysis.

In graphs and hypergraphs one can now define sampling probabilities for edges or hyperedges and sample them accordingly. These probabilities are based on a definition of the strength of edges. It is tempting to follow the same approach for motif instances, however, as already discussed in the technical overview, there is a problem. If we sample a set of motif instances then their union may contain other motif instances, however, as already discussed in the technical overview, there is a problem. If we sample a set of motif instances then their union may contain other motif instances that were not contained in the sample. The reason is simply that motifs are composed of edges. Therefore, later on, we will define an edge-based sampling procedure. It will still be useful for our purposes to define a notion of motif strength. We now give the formal definitions.

Definition 8 (Motif Connectivity). Let $M = (V_M, E_M)$ be a motif, let $G = (V, E, w)$ be a directed weighted graph. $G$ is $(k, M)$-connected if every cut $(S, V \setminus S)$ in $G$ has $M$-motif size at least $k$.

Definition 9 (k-Strong M-Connected Component). Let $M = (V_M, E_M)$ be a motif, let $G = (V, E, w)$ be a directed weighted graph. For a value $k \in \mathbb{R}_+$, an induced subgraph $C = (V_C, E_C, w)$ of $G$ is called a $k$-strongly $M$-connected component of $G$, if

(a) $C$ is $(k, M)$-connected and
(b) there is no induced subgraph $C' = (V_{C'}, E_{C'}, w)$ of $G$ that is $(k, M)$-connected and has $V_C \subseteq V_{C'}$.

We will consider two $M$-connected components distinct if their sets of vertices are distinct.

Definition 10 (Motif Strength). Let $M = (V_M, E_M)$ be a motif, let $G = (V, E, w)$ be a directed weighted graph. Let $I \in \mathcal{M}(G, M)$ be a motif instance. The motif strength $\kappa_I$ of $I$ is the maximum size $k$ such that there exists a $(k, M)$-connected component that contains $I$ as a subgraph.

IV. MAIN RESULTS

In this section we present the main results of this paper. We express runtime and size bounds in $O$ notation, which hides factors polynomial in $\log n$ and motif size. We start by stating the upper bound results in full generality:

Theorem 11. Let $L > 0$ be an integer. For every directed weighted graph $G = (V, E, w)$, $|V| = n$, every set of motifs $\{M_i\}_{i \in [L]}$ and every $\epsilon \in (0, 1)$, a graph $G'$ such that it is an $(M_i, \epsilon)$-motif sparsifier of $G$ for all $i \in [L]$ with $O(Ln/\epsilon^2)$ edges can be computed in time

\[ O \left( L|E| + \sum_{i=1}^{L} T(G, M_i) \right), \]

where $T(G, M_i)$ for $i \in [L]$ is the time required to enumerate all instances of $M_i$ in $G$. The algorithm succeeds with probability at least $1 - (L + 1)n^{-c_1}$ for an arbitrarily large global constant $c_1$.

The main result of the paper is an immediate corollary:

Corollary 12. For every graph $G = (V, E)$, $|V| = n$, every constant integer $k \geq 2$, $\epsilon \in (0, 1)$, there exists an $\epsilon$-motif sparsifier $G'$ of $G$ with respect to all motifs $M$ of size at most $k$ simultaneously that contains $O(n/\epsilon^2)$ edges can be computed in time

\[ O\left(Ln^r + n\omega(r/3^r)\right)\log W), \]

where $r$ is the maximum number of vertices in $M$, $i \in [L]$, $W = \max_{e \in E} w(e)/\min_{e \in E} w(e)$ and $\omega < 2.37286$ is the matrix multiplication constant [59]. The algorithm succeeds with probability at least $1 - (L + 1)n^{-c_1}$ for an arbitrarily large global constant $c_1$.

Although the two algorithms are very similar, there are cases when the first algorithm is faster than the second one. It would still be so even if we were to construct the motif weighted graph through enumeration. This is because computing all-pairs connectivities takes $O(n^3)$ time, while the first algorithm can work in nearly-linear time with respect to the number of motifs. This is relevant when, for example, we have only one motif — triangle — and $|E| = O(n^{4/3 - \delta})$ for $\delta > 0$. Then enumeration can be done in time $O(|E|^{3/2}) = O(n^{2 - 3\delta/2})$ producing at most $O(|E|^{3/2})$ motif instances.

Last but not least, we derive a negative result on the possibility of constructing a motif cut sparsifier for induced motif instances. The definitions of motif cut size and motif cut sparsifier are straightforwardly adapted from non-induced case by counting only induced motif instances as motif instances.

Theorem 14. Let $f(n) = o(n^2)$ and let $\epsilon, 0 < \epsilon \leq 1/500$. There exists a motif $M = (V_M, E_M)$ such that for every sufficiently large integer $n$, there exists a graph $G = (V, E)$ on
n vertices, such that it is impossible to construct an $(\Lambda, \epsilon)$-induced-motif cut sparsifier for $G$ with $f(n)$ non-negatively weighted edges.

Notice that this also includes graphs that are not subgraphs of the original graph.

ACKNOWLEDGMENTS

Mikhail Makarov and Jakab Tardos are supported by ERC Starting Grant 759471. Michael Kapralov is supported in part by ERC Starting Grant 759471. Sandeep Silwal is supported by an NSF Graduate Research Fellowship under Grant No. 1745302, NSF TRIPODS program (award DMS-2022448), and Simons Investigator Award.

REFERENCES

[1] R. Milo, S. S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. B. Chklovskii, and U. Alon, “Network motifs: simple building blocks of complex networks,” Science, vol. 298, 5594, pp. 824–7, 2002.

[2] O. N. Yaveroglu, N. Malod-Dognin, D. R. Davis, Z. Levnajic, V. Janicj, R. Karapandza, A. Stojmiorvic, and N. Przulj, “Revealing the hidden language of complex networks,” Scientific Reports, vol. 4, 2014.

[3] A. R. Benson, D. F. Gleich, and J. Leskovec, “Higher-order organization of complex networks,” Science, vol. 353, no. 6295, pp. 163–166, 2016.

[4] C. E. Tsourakakis, J. Pachocki, and M. Mitzenmacher, “Scalable motif-aware graph clustering,” in Proceedings of the 26th International Conference on World Wide Web, ser. WWW ’17. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, p. 1451–1460. [Online]. Available: https://doi.org/10.1145/3038912.3052653

[5] P. Li and O. Milenkovic, “Inhomogeneous hypergraph clustering with applications,” in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/a50babba152a7a719719719390ca3ebf9e7-Paper.pdf

[6] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, “Local higher-order graph clustering,” in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 555–564. [Online]. Available: https://doi.org/10.1145/3097983.3098060

[7] P. Li, H. Dau, G. J. Puleo, and O. Milenkovic, “Motif clustering and overlapping clustering for social network analysis,” IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, pp. 1–9, 2017.

[8] P. Li, E. Chien, and O. Milenkovic, “Motif clustering and overlapping clustering for social network analysis,” IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, pp. 1–9, 2017.

[9] A. Paranjape, A. R. Benson, and J. Leskovec, “Motifs in temporal networks,” Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017.

[10] C. Seshadhri, A. Sharma, A. Stolman, and A. Goel, “The impossibility of low-rank representations for triangle-rich complex networks,” Proceedings of the National Academy of Sciences, vol. 117, no. 11, pp. 5631–5637, 2020. [Online]. Available: https://www.pnas.org/content/117/11/5631

[11] K. Sotiroopoulos and C. E. Tsourakakis, “Triangle-aware spectral sparsifiers and community detection,” in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, ser. KDD ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1501–1509. [Online]. Available: https://doi.org/10.1145/3474548.3467260

[12] S. Mangan and U. Alon, “Structure and function of the feed-forward loop network motif,” Proceedings of the National Academy of Sciences of the United States of America, vol. 100, no. 11, pp. 11 980 – 11 985, 2003.

[13] S. Wasserman and K. Faust, “Social network analysis - methods and applications,” in Structural analysis in the social sciences, 2007.

[14] E. Wong, B. Baur, S. Quader, and C.-H. Huang, “Biological network motif detection: principles and practice,” Briefings in Bioinformatics, vol. 13, no. 2, pp. 202–215, 06 2011. [Online]. Available: https://doi.org/10.1093/bib/bbr033

[15] D. R. Karger, “Random sampling in cut, flow, and network design problems,” Mathematics of Operations Research, vol. 24, no. 2, pp. 383–413, 1999.

[16] A. Benczur and D. R. Karger, “Randomized Approximation Schemes for Cuts and Flows in Capacitated Graphs,” SIAM Journal on Computing, vol. 44, no. 2, pp. 290–319, Jan. 2015. [Online]. Available: http://epubs.siam.org/doi/10.1137/070705970

[17] D. A. Spielman and S. Teng, “Spectral sparsification of graphs,” SIAM J. Comput., vol. 40, no. 4, pp. 981–1025, 2011. [Online]. Available: https://doi.org/10.1137/07074489X

[18] A. Benczur and D. R. Karger, “Approximating s-t minimum cuts in Õ(n2) time,” in STOC ’96, 1996.

[19] Y. Chen, S. Khanna, and A. Nagda, “Near-linear size hypergraph cut sparsifiers,” in 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS). IEEE, 2020, pp. 61–72.

[20] K. J. Ahn and S. Guha, “Graph sparsification in the semi-streaming model,” in International Colloquium on Automata, Languages, and Programming. Springer, 2009, pp. 328–338.

[21] M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford, “Single pass spectral sparsification in dynamic streams,” 2014 IEEE 55th Annual Symposium on Foundations of Computer Science, pp. 561–570, 2014.

[22] A. Rubinstein, T. Schramm, and S. M. Weinberg, “Computing exact minimum cuts without knowing the graph,” in 9th Innovations in Theoretical Computer Science, ITCS 2018. Schloss Dagstuhl-Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, 2018, p. 39.

[23] S. Assadi, Y. Chen, and S. Khanna, “Polynomial pass lower bounds for graph streaming algorithms,” Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 2019.

[24] S. Mukhopadhyay and D. Nanongkai, “Weighted min-cut: sequential, cut-query, and streaming algorithms,” Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, 2020.

[25] S. Assadi and A. Dudeja, “A simple semi-streaming algorithm for global minimum cuts,” in SODA, 2021.

[26] V. Braverman, R. Ostrovsky, and D. Vilenchik, “How hard is counting triangles in the streaming model?” in Automata, Languages, and Programming, F. V. Fomin, R. Freivalds, M. Kwiatkowska, and D. Peleg, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 244–254.

[27] I. Newman and Y. Rabinovich, “On multiplicative lambda-approximations and some geometric applications,” SIAM J. Comput., vol. 42, pp. 855–883, 2015.

[28] D. Kogan and K. Krauthgamer, “Sketching cuts in graphs and hypergraphs,” in Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, 2015, pp. 367–376.

[29] T. Soma and Y. Yoshida, “Spectral sparsification of hypergraphs,” in Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM, 2019, pp. 2570–2581.

[30] M. Kapralov, R. Krauthgamer, J. Tardos, and Y. Yoshida, “Towards tight bounds for spectral sparsification of hypergraphs,” in Proceedings of the 53rd Annual ACM-SIAM Symposium on Theory of Computing, ser. STOC 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 598–611. [Online]. Available: https://doi.org/10.1145/3406325.3451061

[31] W.-S. Fung, R. Hartharan, N. J. Harvey, and D. Panigrahi, “A general framework for graph sparsification,” SIAM Journal on Computing, vol. 48, no. 4, pp. 1196–1223, 2019.

[32] N. Alon, P. Dao, I. Hajirasouliha, F. Hormozdiari, and S. C. Sahinalp, “Biomolecular network motif counting and discovery by color coding,” Bioinformatics, vol. 24, pp. i241–i249, 2008.

[33] M. Bressan, “Efficient and near-optimal algorithms for sampling connected subgraphs,” Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science, 2019.

[34] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, and W. Zhu, “Arbitrary-order proximity preserved network embedding,” in Proceedings of the 24th
[37] H. Nassar, C. Kennedy, S. Jain, A. R. Benson, and D. Gleich, “Using cliques with higher-order spectral embeddings improves graph visualizations,” in Proceedings of The Web Conference 2020, ser. WWW ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 2927–2933. [Online]. Available: https://doi.org/10.1145/3366423.3380059

[38] N. Ahmed, J. Neville, R. A. Rossi, N. G. Duffield, and T. L. Wilke, “Graphlet decomposition: framework, algorithms, and applications,” Knowledge and Information Systems, vol. 50, pp. 689–722, 2016.

[39] R. A. Rossi, N. Ahmed, and E. Koh, “Higher-order network representation learning,” Companion Proceedings of the The Web Conference 2018, 2018.

[40] D. Eswaran, S. Kumar, and C. Faloutsos, “Higher-order label homogeneity and spreading in graphs,” Proceedings of The Web Conference 2020, 2020.

[41] F. Tudisco, A. R. Benson, and K. Prokopchik, “Nonlinear higher-order label spreading,” Proceedings of the Web Conference 2021, 2021.

[42] A. R. Benson, R. Abbe, M. T. Schaub, A. Judaibaie, and J. M. Kleinberg, “Simplicial closure and higher-order link prediction,” Proceedings of the National Academy of Sciences, vol. 115, pp. E11221 – E11230, 2018.

[43] F. Arrigo, D. J. Higham, and F. Tudisco, “A framework for second-order eigenvector centralities and clustering coefficients,” Proceedings of the Royal Society A, vol. 476, 2020.

[44] R. A. Rossi, A. Rao, S. Kim, E. Koh, and N. Ahmed, “From closing triangles to closing higher-order motifs,” in Companion Proceedings of the Web Conference 2020, ser. WWW ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 42–43. [Online]. Available: https://doi.org/10.1145/3366424.3382688

[45] A. R. Benson, “Three hypergraph eigenvector centralities,” SIAM Journal on Mathematics of Data Science, vol. 1, no. 2, pp. 293–312, 2019.

[46] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” in International Conference on Learning Representations, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6uA5Km

[47] T. Zhao, Y. Liu, L. Neves, O. J. Woodford, M. Jiang, and N. Shah, “Data augmentation for graph neural networks,” in AAAI, 2021.

[48] S. Liu, H. Dong, L. Li, T. Xu, Y. Rong, P. Zhao, J. Huang, and D. Wu, “Local augmentation for graph neural networks,” 2022. [Online]. Available: https://openreview.net/forum?id=3FvF1db-bKT

[49] C. Seshadhri and S. Tirthapura, "Scalable subgraph counting: The methods behind the madness," in Companion Proceedings of The 2019 World Wide Web Conference, ser. WWW ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1317–1318. [Online]. Available: https://doi.org/10.1145/3308560.3320092

[50] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller, “Triangle sparsifiers,” J. Graph Algorithms Appl., vol. 15, pp. 703–726, 2011.

[51] R. M. Karp, "Reducibility among combinatorial problems," in Complexity of computer computations. Springer, 1972, pp. 85–103.

[52] N. Chiba and T. Nishizeki, " Arboricity and subgraph listing algorithms," SIAM Journal on computing, vol. 14, no. 1, pp. 210–223, 1985.

[53] N. Alon, R. Yuster, and U. Zwick, "Color-coding," Journal of the ACM (JACM), vol. 42, no. 4, pp. 844–856, 1995.

[54] P. Ribeiro, P. Paredes, M. E. Silva, D. Aparicio, and F. Silva, "A survey on subgraph counting: concepts, algorithms, and applications to network motifs and graphlets," ACM Computing Surveys (CSUR), vol. 54, no. 2, pp. 1–36, 2021.

[55] D. Kogan and R. Krauthgamer, "Sketching cuts in graphs and hypergraphs," in Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, 2015, pp. 367–376.

[56] C. Chekuri and C. Xu, "Minimum cuts and sparsification in hypergraphs," SIAM Journal on Computing, vol. 47, no. 6, pp. 2118–2156, 2018.

[57] V. V. Williams and R. Williams, "Finding, minimizing, and counting weighted subgraphs," SIAM Journal on Computing, vol. 42, no. 3, pp. 831–854, 2013.

[58] A. Abboud, R. Krauthgamer, J. Li, D. Panigrahi, T. Saranurak, and O. Trabelsi, "Gomory-hu tree in subcubic time," arXiv preprint arXiv:2111.04958, 2021.

[59] J. Alman and V. V. Williams, "A refined laser method and faster matrix multiplication," in Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA). SIAM, 2021, pp. 522–539.