Efficiently Clustering Dense Network Graphs by Sampling and Counting Organized Substructures

Dylan Lee, Matthew Wilson, Karthikeya Manchala, Jonathan Li

March 14, 2023

Abstract

Network graphs have emerged to be a relevant tool in organizing and investigating the relationship among individual agents in systems within numerous domains of study. Yet in certain contexts, clustering a network to identify unique sub-populations can uncover hidden insights of the data to encourage further analysis. Although clustering isn't a relatively novel concept in network science, partitioning around recurring substructures, or motifs, that cumulatively construct these dense graphs remains unexplored. In order to perform such clustering, our main endeavor is to develop an efficient pipeline for counting the number of occurrences of specific substructures in any densely concentrated network graph. Primarily we will be examining the frequency of triangular motifs, all of which contain three separate nodes, but vary in the number and direction of edges. While required for spectral clustering, motif counting imposes a bottleneck as the most time-consuming step which we aim to accelerate by introducing sampling algorithms to reduce a dense network to a sparser, but still representative subset.

Introduction

Networks have become a widely popularized method for representing interconnected systems of data across various academic disciplines - mapping the transmitted activations between neurons in the brain, tracking the behavior of social media users, and countless more. While an individual node is commonly considered to be the elementary building block of a typical network graph, many density-rich networks demonstrate connectivity patterns when considering higher-order motifs as the fundamental unit of analysis. Examining the properties of a network can often require its partition into separate sub-graphs which represent individual populations. Methods to cluster network graphs on the lower level of individual nodes have already been thoroughly discussed, but few have investigated clustering a network upon the higher level of motifs, which we wish to further explore in this study. A preliminary step we must take however, is to efficiently measure the frequency that a given motif appears in our network of inquiry. By counting each unique occurrence of a motif and recording the specific nodes that instance is composed of, we are now enabled to utilize a scalable, yet efficient algorithm to perform higher-order clustering.

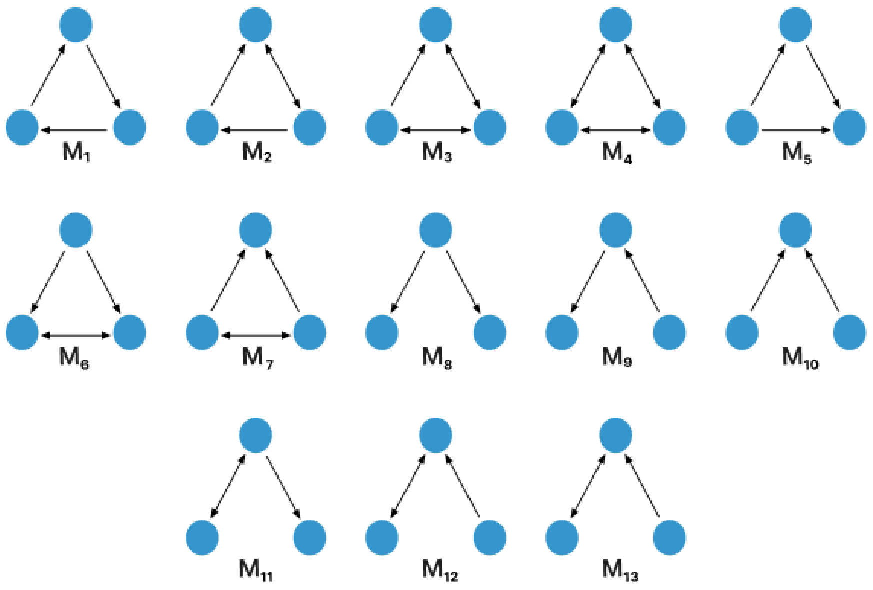
Much of our endeavors are based upon the substantial work done by Austin R. Benson, David F. Gleich, and Jure Leskovec in their work, *Higher-order organization of complex networks*. The authors most notably establish the central framework by which to partition clusters that are maximally 'separated.' According to Benson, Glech, and Leskovec, sub-graphs should be separated on the basis of minimizing motif conductance, quantified by the number of motifs with nodes spanning more than one cluster. Motif conductance results in what the authors refer to as cuts, where edges of a motif 'cut' or intersect multiple clusters. Deriving the authors' algorithm to perform approximately optimal clustering according to this framework first requires an adjacency matrix of motif frequencies, and thus comes our primary objective of efficiently counting their recurring instances.

As discussed earlier, with networks remaining to be a prevalent metric to analyze complex systems for many scientific disciplines, the methodologies for motif counting and clustering will be applicable to virtually any domain. However, in demonstrating our algorithmic procedures, we will be referencing our work done with a wide range of graph datasets ranging in topics from airline travel, cryptocurrency, internet weblinks, and email communications. Many if not all, of these datasets are originally sourced as a raw text file that were pre-processed into an unweighted, directed network graph for analysis. Prior to performing clustering, exploratory data analysis will also be executed to calculate elementary statistics such as the number of independent nodes and connecting edges, as well as the distribution of node degrees. Regarding the motifs themselves, our investigation will count the frequency of 13 unique triangular substructures, all of which are composed of three distinct nodes with variations in the number and direction of connecting edges.

Pseudocode outlining the steps to implement our algorithm to count all instances of motif 1 for example are outlined below:

Motif Counting Algorithm

For reference, shown below are the 13 triangular motifs for which we are counting and classifying upon alongside pseudocode of our native counting algorithm for specifically motif 1:

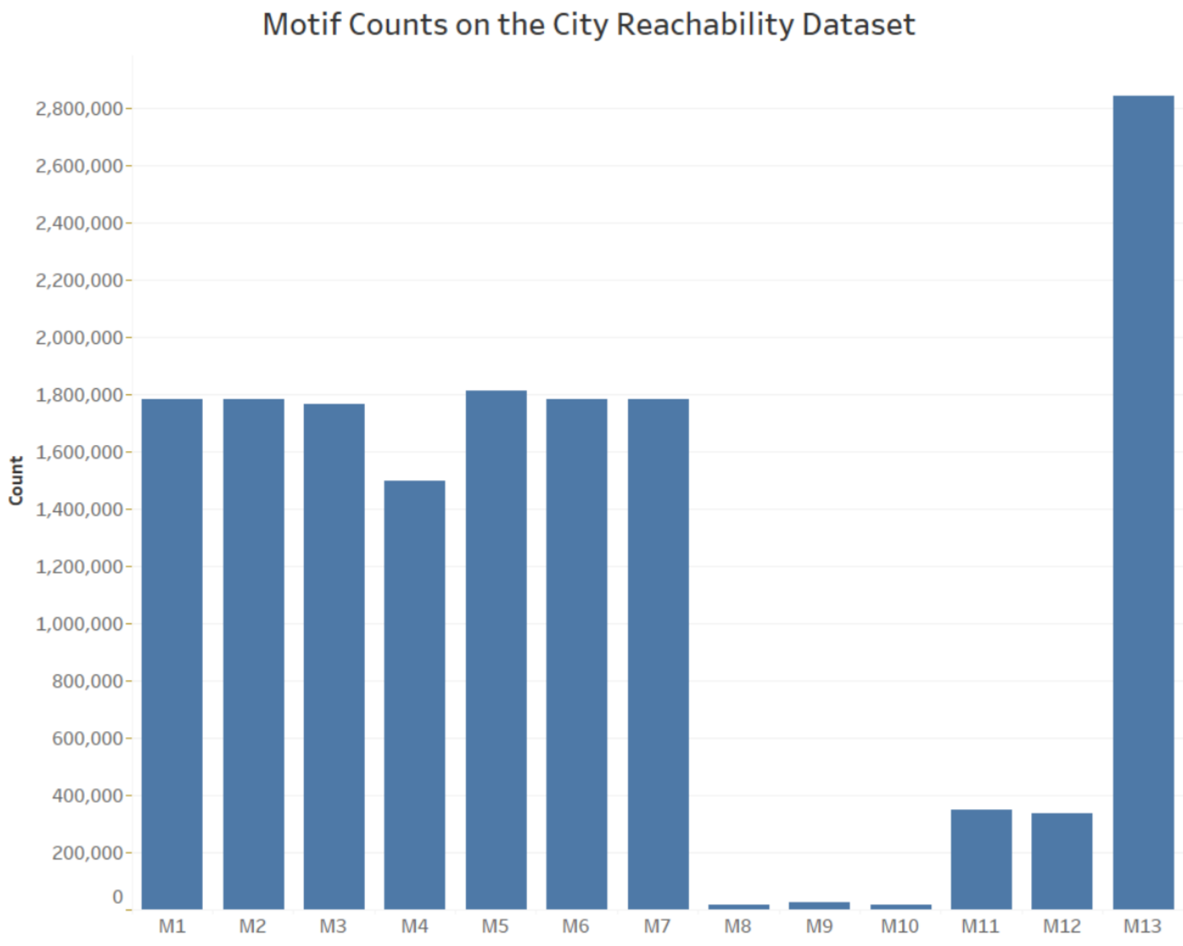


Input: Directed, unweighted graph G and motif M
Output: Set of all uniquely counted instances of motif M in G
adj_list_away : dictionary containing all the nodes for which the given node in G has an edge going towards
adj_list_in : dictionary containing all the nodes which have an edge pointing towards the given node in G
motif_occurrences : array to store all counted instances of motif 1 in G

```
Iterate for each unique vertex $v_1$ in adj_list_away :
|   Iterate for each unique vertex $v_2$ which $v_1$ points to (obtained via adj_list_away) :
|       Iterate for each unique vertex $v_3$ which $v_2$ points to (obtained via adj_list_away) :
|           If ($v_3$ points to $v_1$) and ($v_1$ does not point towards $v_3$) and
|           ($v_2$ does not point towards $v_3$) and ($v_2$ does not point towards $v_1$) :
|               Store $\{v_1, v_2, v_3\}$ as an instance of motif 1 in $G$.
Remove all duplicate permutations of any counted motif in motif_occurrences.
```

As can be seen above, the motif counting algorithm sequentially iterates through each possible combination of connected node triplets that exist in the network graph G using two adjacency lists, ***adj_list_away*** and ***adj_list_in***, which are both used to keep track of directed edges between nodes. In fact, all of our counting algorithms for the 13 unique motifs follow this similar structure of iterating through three connected vertices to detect an possible occurrence of a given motif. Thus, the computational time complexity for all of our counting algorithms is $O(n^3)$, where n is the number of nodes in G. Where each of the 13 counting algorithms differ for their specific motif is in the terminating step, where it checks to see if the edges among the three connected vertices follow the same directional patterns as outlined by the given motif.

For demonstration, one of the datasets for which the counting algorithms of each motifs were applied to was a network comprising the reachability of airline travel among cities in the United States and Canada. As shown in the visualization below, the structure of certain motifs can attribute towards a greater prevalence in specific contextual domains than others. More specifically, we see that for triangular motifs, M_1 through M_7 all have a similar number of occurrences due to the overall density of this particular network and similarity of structure, only seemingly differing in edge direction. However, a significant outlier can be witnessed for wedges (motifs M_8 to M_{13}), for which M_{13} appears much more prominently than the rest.



Often times, first counting all the substructures in any network can provide useful insight when deciding the choice of motif to cluster upon, as we would intuitively start our analysis initially with the motif most prominent in frequency. With this in mind, subsequent sections will thoroughly delve into the process of clustering based on motif occurrences.

Spectral Motif Clustering

One of the primary motivations behind our exploration into higher-order spectral clustering is the reconsideration of motifs (in our particular case, triangular substructures) to be the primary building blocks of complex networks, rather than individual nodes themselves. Analyzing network-structured data through a motif-centric representation can be especially useful in understanding modular connectivity patterns in many contextual domains of study - some of which include peer networks on social media, neuron connections in the brain, and even flight patterns between adjacent cities. Since clustering networks on the basis of motif composition as opposed to individual nodes is a relatively novel endeavor, we hope to promote further exploration by discussing possible algorithms to handle this task.

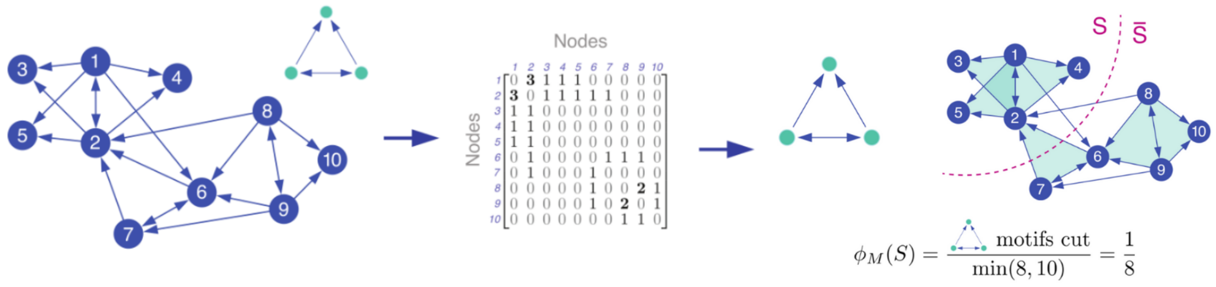
When identifying possible sub-communities within a complex network, we want reoccurring motifs to stay centrally located only within a single class, as opposed to spanning across multiple clusters. This follows natural intuition that these communities are well separated, such that motifs do not intersect across multiple clusters yet remain densely self-contained. Building upon this framework, an optimal cluster S is thus a set of nodes which minimizes the motif conductance, defined to be

$$\boldsymbol{\phi(S) = cut_M(S, \overline{S})/min(vol_M(S), vol_M(\overline{S}))}$$

where $\boldsymbol{cut_M(S, \overline{S})}$ is the number of motifs whose nodes lie in more than one unique cluster (i.e, the number of motifs whose edges cross the boundary delineating separate clusters apart) and $\boldsymbol{vol_M(S)}$ is the number of nodes counted in all occurrences of a given motif M that are located in cluster S (this can be simply counted by the number of motif end points in S).

Ideally, given a network and a motif to construct its corresponding graph by, we'd want to find the cluster that maximizes separability by minimizing the motif conductance. However, empirically determining the exact cluster which absolutely minimizes this metric is a computationally infeasible problem (NP-Hard). To workaround this obstacle, Austin R. Benson, David F. Gleich, and Jure Leskovec in their paper, *Higher-order organization of complex networks* present two algorithms which utilize spectral clustering to find near-optimal sub-communities.

The following is a diagram outlining our overall motif counting and clustering pipeline. First starting with the network graph to be analyzed, all unique instances of the desired motif are counted, for which each and every participating edge is used to form the motif adjacency matrix shown. This matrix is then passed an input into either of two viable spectral clustering algorithms that output the near-optimal clusters with minimal motif conductance.


Nodes
Nodes
S
S̄
$\phi_M(S) = \frac{\text{motifs cut}}{\min(8, 10)} = \frac{1}{8}$

$$\begin{array}{c|cccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \hline 1 & 0 & 3 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 7 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 1 \\ 9 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 1 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}$$

Both of these spectral clustering algorithms are explored in much further detail in the following sections.

Algorithm 1: Motif-based clustering algorithm for finding a single cluster

Pseudocode outlining the steps to implement algorithm 1 are outlined below:

```
**Input:** Directed, unweighted graph G and motif M
**Output:** Motif-based cluster (subset of nodes in G)
$(W_M)_{ij}$ : number of instances of M that contain nodes i and j
$G_M$ : weighted graph induced $W_M$
$D_M$ : diagonal matrix with $(D_M)_{ii} = \sum_j (W_M)_{ij}$
$z$ : eigenvector of the second smallest eigenvalue for $\mathcal{L}_M = I - D_M^{-1/2} W_M D_M^{-1/2}$
$\sigma_i$ : to be index of $D_M^{-1/2} z$ with the $i$th smallest value
**Sweep over all prefixes of $\boldsymbol{\sigma}$ :**
$S$ : $argmin_l \phi^{(G_M)}(S_l)$, where $S_l = \{\sigma_1, ..., \sigma_l\}$
**if** $S < |\overline{S}|$**:**
|   **return** $S$
**else:**
|   **return** $\overline{S}$
```

To begin the derivation for the first clustering algorithm, which calculates a single cluster S (and thereby its complement $\overline{S}$), we begin with a directed, unweighted graph G, and a motif M as inputs. From G, we form W_M, the motif adjacency matrix, such that each value in coordinates (i, j) corresponds to the number of times the edge between node i and node j participates in all instances of motif M. In fact, by using the motif counting algorithms discussed earlier, we can obtain the counts for every edge between every possible combination of nodes i and j where $i \neq j$. These counts will be used to form W_M once the designated motif counting algorithm is conducted on G.

W_M will then be used to form a diagonal matrix $(D_M)_{ii} = \sum_j (W_M)_{ij}$, where each diagonal entry is the cumulative sum of all values in every row of W_M. Afterwards we perform Laplacian Matrix Normalization, obtaining the matrix $\mathcal{L}_M = I - D_M^{-1/2} W_M D_M^{-1/2}$. The Fielder vector, $f^{(M)} = D_M^{-1/2} z$ is calculated where z is the eigenvector corresponding to the second smallest eigenvalue of $\mathcal{L}$.

In the final step, we must sort the entries of $f^{(M)}$ in ascending order to create a set $S_l = \{\sigma_1, ..., \sigma_l\}$, where σ_i represents the index of the ith smallest element in $f^{(M)}$. Finally, the algorithm recursively calculates the motif conductance of the set $S_r = \{\sigma_1, ..., \sigma_r\}$ where $r = 1, ..., l$. To demonstrate, this would first involve calculating the motif conductance of $\{\sigma_1\}$, then $\{\sigma_1, \sigma_2\}$, up to $\{\sigma_1, ..., \sigma_l\}$. This algorithm completes by returning the set with the smallest calculated motif conductance as the near-optimal cluster S.

Cheeger Inequality

An important fact regarding the first algorithm is that Benson, Gleich, and Leskovec specifically reference the **Cheeger Inequality** to demonstratively prove that it will return a near-optimal cluster, since finding the most optimal cluster is computationally intractable.

The **Cheeger Inequality** is as follows:

$$\phi_M(S) \leq 4\sqrt{\phi_M^*}$$

where $\phi_M(S)$ is the motif conductance of the set returned by algorithm 1 and ϕ_M^* is similarly the motif conductance of the theoretically most optimal cluster. Specifically, this inequality states that the motif conductance of the algorithm's returned cluster S is only within a quadratic factor of the most optimal motif conductance theoretically possible.

Algorithm 2: Motif-based clustering algorithm for finding several clusters

Pseudocode outlining the steps to implement algorithm 2 are outlined below:

```
Input: Directed, unweighted graph G, motif M, number of clusters $k$
Output: $k$ disjoint motif-based clusters
$(W_M)_{ij}$ : number of instances of M that contain nodes i and j
$D_M$ : diagonal matrix with $(D_M)_{ii} = \sum_j (W_M)_{ij}$
$z_1, ..., z_k$ : eigenvectors of the $k$ smallest eigenvalues for $\mathcal{L}_M = I - D_M^{-1/2} W_M D_M^{-1/2}$
$Y_{ij}$ : $z_{ij} / \sqrt{\sum_{j=1}^{k} z_{ij}^2}$
Embed node $i$ into $\mathbb{R}^k$ by taking the $i$th row of the matrix $Y$
Run $k$-means clustering on the embedded nodes
```

Before we delve into the implementation for algorithm 2, it's important to note several key differences between it and the first algorithm. Most notably, while algorithm 1 allows the user to only find a single cluster, the second algorithm can be used to determine multiple clusters at once. However, a tradeoff is that it's not as theoretically robust as algorithm 1, lacking the same guarantee on optimal motif conductance granted by the Cheeger inequality.

From visual inspection alone, it can be seen that both algorithms share many of the same inputs (with the exception of algorithm 2 requiring an additional input k for the number of desired clusters to be determined) and introductory steps, up to performing Laplacian Matrix Normalization to get $\mathcal{L}_M = I - D_M^{-1/2} W_M D_M^{-1/2}$. From there, algorithm 2 deviates by calculating the eigenvectors $\{z_1, ..., z_k\}$ of the k smallest eigenvalues of $\mathcal{L}_M$.

With the set of eigenvectors $\{z_1, ..., z_k\}$, we can form a matrix Z using them as column vectors, such that $Z = [z_1 \ ... \ z_k]$. Then to create the matrix Y, divide each value in every row of Z by the square root of the square sum of that row's entries. This is equivalent to $Y_{ij} : z_{ij} / \sqrt{\sum_{j=1}^{k} z_{ij}^2}$. Here, Y is an $n \times k$ matrix, where each row represents the coordinates of one of the n nodes in k-dimensional space. For example, if the first row of Y is the row vector $[1 \ \ 2 \ \ 3]$, then node 1 is represented by a point located at $(1, 2, 3)$ in $\mathbb{R}^3$. After embedding each of the n nodes onto points in $\mathbb{R}^k$, the k-means clustering algorithm is run on these points to classify each node to one of the k clusters.

Sampling Graph Networks

To recap, our pipeline for the transforming a raw dataset of a network graph into an organized and partitioned dataset is as follows:

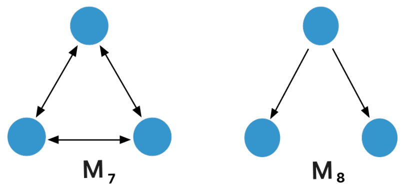
1. Clean and pre-process the raw data, which comes in a text file format.
2. Count the number of occurrences of each motif in the network, identifying motifs which are
| especially prevelant.
3. Utilize the edges of every observed motif to form the motif adjacency matrix.
4. Pass in the motif adjacency matrix as inputs into the spectral clustering algorithms.

However, when executing these steps on datasets of considerable magnitude, practical concerns regarding efficient runtime and space demands can arise which impart an obstacle in a successful completion. In particular, the second step of motif counting becomes a major bottleneck for computational execution time. Thus, one possible workaround is to explore various network sampling methods such that we will only have to perform motif counting on only a smaller subset of the original network which ideally minimizes the total runtime while still preserving as much contextual information as possible to obtain near-optimal clusters.

Our proposed sampling technique we've recently investigated is outlined below:

1. Establish a sampling threshold k for the number of edges sampled.
2. Depending on the motif of interest, designate a "central" node p.
3. Like the naive algorithm, repeatedly iterate through each edge in the adjacency lists until vertex p | is reached.
4. Count the number of edges n extending from vertex p which satisfy the fulfilled motif as its degree.
5. If $n \leq k$, count all of those edges towards instances of the desired motif to cluster on.
6. Otherwise, randomly sample k of those edges towards instances of the target motif.
7. When forming the motif adjacency matrix for the spectral clustering algorithms, the counts of any | sampled motif edges are upscaled by a factor proportional to the degree of p.

Note that the above algorithm mentions that the choice of the node for p largely depends on the structure of the triangular motif in regards to the number and direction of its edges. Take for example, motifs M_7 and M_8, shown below for convenience of reference:

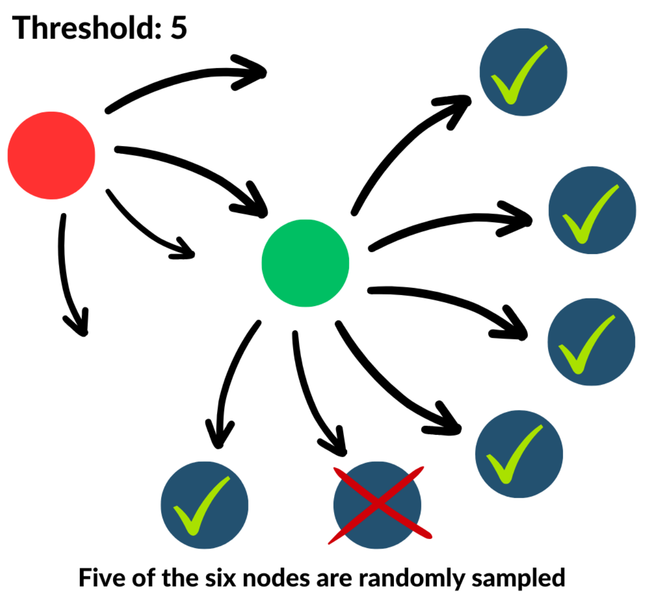


Because M_7 has a symmetrical structure, having bidirectional edges between all three vertices, the choice of node p for which to sample on if required, doesn't necessarily matter on the result of the output counts (or the subsequent cluster for that matter). On the other hand, being a wedge, M_8 is only symmetric around its top-most node in the middle, so when counting M_8 specifically, that node in particular is selected to be p. Certain motifs may lack any form of symmetricity at all, so when choosing a "central" node, we opt for the vertex that has the most connections (of similar direction if possible) either extending from or pointing towards it.

The sampling algorithm additionally contains a parameter k which designates the number of nodes that can be counted for the desired motif to cluster the network upon from what we call the "central" node p of that motif. The smaller the value of k, the sparser the subset of the original network will be counted for motif occurrences, inducing a greater speed up of execution time by reducing complexity further away from $O(n^3)$ by essentially omitting one of the iterations of the naive counting algorithm. Naturally as k increases, a larger number of edges are accounted for, making the subset of the network approach the original but results a greater execution time of the clustering pipeline as the bottleneck.

But not only do we want to decrease execution time, we aim to still preserve accuracy. We consider the accuracy yielded by our sampling algorithm at a threshold k to intuitively be a measure of how close the cluster obtained from a sampled network is from that obtained by the entire original graph which we consider our baseline. Naturally a larger value for k increases accuracy at the cost of minimizing execution time, and vise-versa for a lower threshold.

Calculating the accuracy is inversely related to the dissimilarity of outputted clusters when sampling, found by first taking the absolute value of the element-wise difference between the adjacency matrices of motif counts from the non-sampled and sampled graphs. Afterwards, the sum of these absolute differences is divided by the sum of edge counts in the adjacency matrix from the original, non-sampled network to obtain a normalized proportion. Although this metric doesn't involve utilizing the clusters themselves directly, quantitatively measuring the dissimilarity between the inputs of the clustering algorithms provides an approximation of the relative closeness between the clusters they output. Reducing the total difference indicates that the sampling method resulted in a cluster similar to the baseline, thus having a higher accuracy. Our current objective is to find the Pareto-optimal parameter which ideally balances our objectives of maximizing cluster accuracy while minimizing the duration of execution.

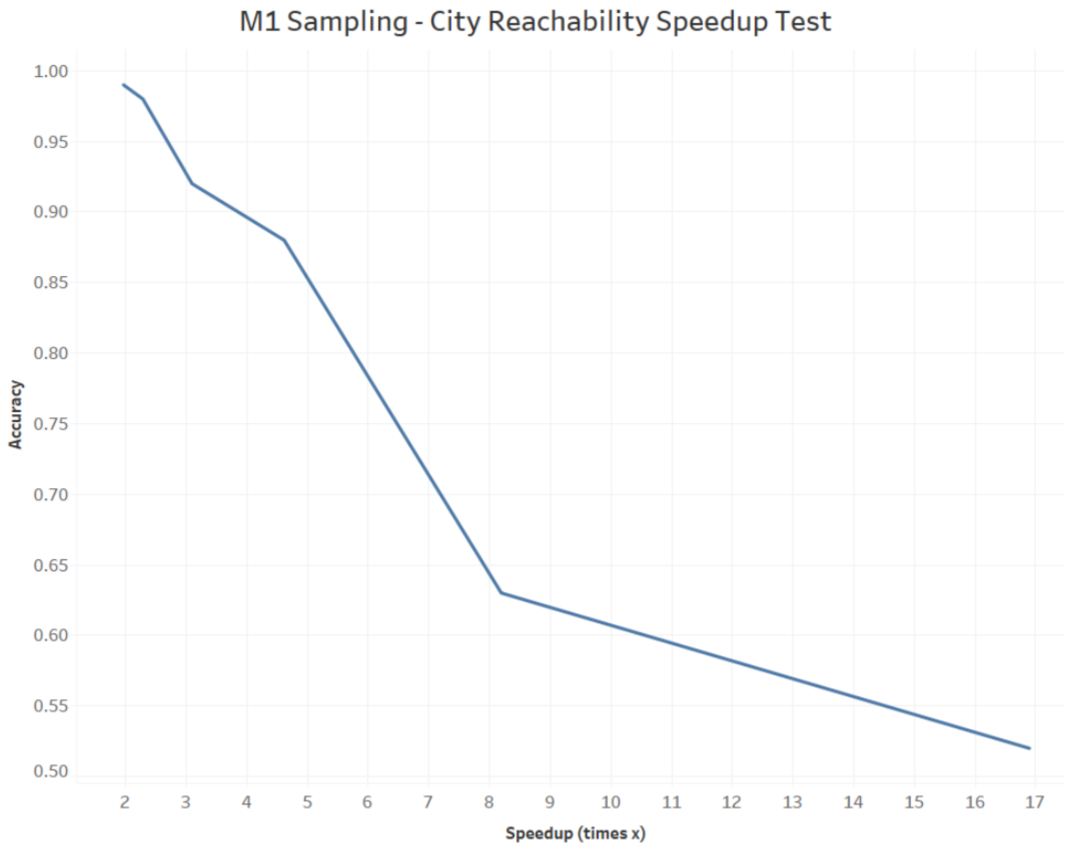


In the above figure, the sampling threshold is set to a value of 5, denoting that at most 5 edges can be considered from any node towards the occurrence of a target motif at any time. Assume that in this subset network, the central green node is chosen to be vertex p in regards to the sampling algorithm, and that all 6 of its edges are used in distinct occurrences for the motif of interest. Having a larger degree than the threshold, only 5 of the 6 edges are randomly sampled to count towards unique motif appearances while the remaining edge is disregarded. However, the edge counts for the motifs still observed despite the sampling are then upscaled by a factor proportional to the degree of node p to account for a lower skew due to the sparsity of the subset. On the other hand, if we take the red node to be vertex p, all of its four edges (assumed to be part of individual motif occurrences) will be accounted for with the threshold of 5.

Results

To gauge the effectiveness of our proposed sampling algorithm with respect to the execution time and preserved accuracy of our motif counting and clustering pipeline, the accuracy of numerous clusters were measured from various datasets for multiple parameter combinations between the choice of motif and the value of threshold k. To examine the effects of raising the sampling threshold, we first started with the network of city reachability via airline travel introduced earlier, and calculated the accuracy of clusters computed after sampling at increasing intervals for k. The results of this analysis are shown in the following table, accompanied by the visualization on the following page:

Threshold	Accuracy	Speedup
2	0.52	16.893891
5	0.63	8.192926
10	0.88	4.614148
15	0.92	3.096463
20	0.98	2.286174
25	0.99	1.967846

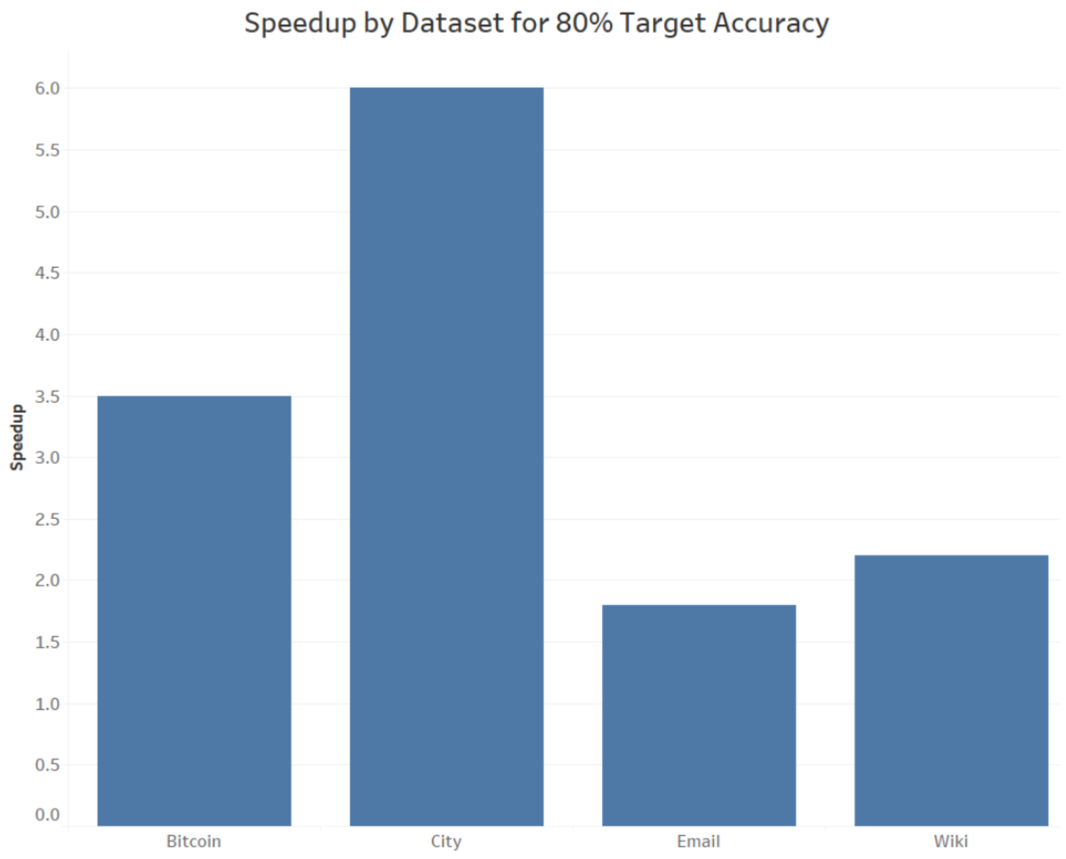


Spectral clustering was performed upon occurrences of motif M_1 after sampling the graph at various values for the threshold k, which range from the interval $[2, 25]$. It can be seen that the largest threshold of 25 obtained a near perfect accuracy of approximately 99%, almost matching the input adjacency matrix obtained from the original graph, while executing twice as fast. Reducing the threshold size does visibly improve execution speed by a inversely proportional multiplicative factor, but at the cost of accuracy as expected from earlier discussion.

Rather than constraining our analysis to just one graph however, we also investigated how the density and structure of multiple networks from various domains affected the obtainable speedup possible via sampling while still hitting a benchmark accuracy. This was investigated using again the city reachability network, alongside three other graphs about Bitcoin transactions, messages between email users, and links between various pages on Wikipedia. For each network, we've measured the speedup of the counting and clustering pipeline for a motif prominent to each at a sampling threshold which met an accuracy of approximately 80%.

The results of this analysis are displayed in the following table and visualization below:

Dataset	Speedup
City	6.0
Wiki	2.2
Email	1.8
Bitcoin	3.5



The city reachability dataset allows for a remarkable 6× speedup while still obtaining 80% whereas the Wikipedia and Bitcoin graphs each manage around 2× to 3× increase in execution speed respectively. The lowest speedup obtainable was for the email network, only performing counting and clustering 1.8× times as fast, which while not necessarily an impressive magnitude at first glance, can still provide valuable practical usefulness in preserving computational resources of time and space-wise.

Discussion

Our objective to explore and optimize motif spectral clustering began with first naively counting the unique instances of a target motif in the input graph, later used to form the motif adjacency matrix. Such matrices are relied upon as the inputs for the clustering algorithms introduced by Austin R. Benson, David F. Gleich, and Jure Leskovec in *Higher-order organization of complex networks*. However, because motif counting is a costly procedure for especially more dense networks, sampling methods derived from modifying the counting algorithm provide a substantial improvement in the pipeline's completion time while achieving clusters with considerable accuracy. From the results of our investigation, it's evident that a lower threshold marginally reduces the computation time at the cost of accuracy, where as a larger threshold prioritizes accuracy at a lower speedup. Although it's best to find a Pareto-optimal sampling threshold that compromises both accuracy and speedup, the value of such a parameter inevitable varies with each network, dependent on factors such as structure and density.

Given ample time to further expand this study, we'd be eager to work with motifs of a variety of shapes, beyond that of triangular structure, alongside datasets of a much greater magnitude in size. We additionally encourage further experimentation with alternative counting methods beyond the $O(n^3)$ naive approach that when coupled with sampling, could perhaps accelerate already existing speedups in execution. Lastly, this endeavor could be extended to empirically determine the relationship between network structure and the optimal motif for clustering, both from a context-specific and a graph theory lens. Ultimately, the topic of motif counting and clustering of networks still remains to be a relatively new domain left with much to be researched. But nevertheless, we hope that by achieving efficient motif spectral clustering, new avenues will open for meaningful and insightful analysis of networks and scientific discovery for any academic domain of study.

References

Higher-order organization of complex networks
Austin R. Benson, David F. Gleich and Jure Leskovec (July 7, 2016)

Supplementary Materials for Higher-order organization of complex networks
Austin R. Benson, David F. Gleich and Jure Leskovec (July 1, 2016)