Using Cliques with Higher-order Spectral Embeddings Improves Graph Visualizations

Huda Nassar*
Stanford University
hnassar@stanford.edu

Caitlin Kennedy*
Purdue University
caitlin.kennedy@alumni.purdue.edu

Shweta Jain
University of California, Santa Cruz
sjain12@ucsc.edu

Austin R. Benson
Cornell University
arb@cs.cornell.edu

David F. Gleich
Purdue University
dgleich@purdue.edu

ABSTRACT

In the simplest setting, graph visualization is the problem of producing a set of two-dimensional coordinates for each node that meaningfully shows connections and latent structure in a graph. Among other uses, having a meaningful layout is often useful to help interpret the results from network science tasks such as community detection and link prediction. There are several existing graph visualization techniques in the literature that are based on spectral methods, graph embeddings, or optimizing graph distances. Despite the large number of methods, it is still often challenging or extremely time consuming to produce meaningful layouts of graphs with hundreds of thousands of vertices. Existing methods often either fail to produce a visualization in a meaningful time window, or produce a layout colorfully called a "hairball", which does not illustrate any internal structure in the graph. Here, we show that adding higher-order information based on cliques to a classic eigenvector based graph visualization technique enables it to produce meaningful plots of large graphs. We further evaluate these visualizations along a number of graph visualization metrics and we find that it outperforms existing techniques on a metric that uses random walks to measure the local structure. Finally, we show many examples of how our algorithm successfully produces layouts of large networks. Code to reproduce our results is available.

KEYWORDS

graph visualization, graph layout, spectral methods, higher-order methods, cliques sampling

ACM Reference Format:
Huda Nassar, Caitlin Kennedy, Shweta Jain, Austin R. Benson, and David F. Gleich. 2020. Using Cliques with Higher-order Spectral Embeddings Improves Graph Visualizations. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3366423.3380059

*Joint first co-authorship.

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380059

1 INTRODUCTION

Graphs provide a succinct way of representing relationships among entities. Carefully designed visualizations of these graphs can reveal interesting properties and patterns, and this information is often useful in speeding knowledge discovery on graphs. Graph visualization is specifically helpful to network analysis tasks such as clustering and prediction [19], and there is a rich literature demonstrating the usage and techniques of visualization in bioinformatics [13, 17, 26]. The goal of graph visualizations is to make the structure in graphs human-readable such that it will be helpful to investigators who use visual interfaces to augment communications of their findings [17]. Ideally, a graph visualization method aims at grouping vertices that have a common value of some latent attribute that impacts the edges of the graph.

There have been several algorithms proposed and empirical studies conducted for visualizing small graphs (up to a few 100 vertices), such as the well known Gephi [4], but scaling beyond this size scale has proven to be much more challenging [17]. This is because, as the size of the graph increases, it becomes much more difficult to put additional data in the same space in a way that still makes intuitive sense and is intelligible. Indeed, the term "hairball" has been used to describe the resulting visualization because of its nature that is complex yet lacking structure. The recent work in [10] addresses the "hairball" phenomenon specifically, and proposes a method to reduce a graph to a skeletal structure by removing some of the edges and keeping only the essential relationships. Progress has been made in visualizing graphs at size up to a million vertices [1, 3]. These methods, however, tend to be fairly brittle and often simply fail to give any significant structural insights for large graphs and largely indistinguishable from a random layout.

One of the hallmarks of real world graphs is that they are not random and the edges are concentrated in dense pockets. For example, in social networks, groups with similar location or interests or belonging to some organization tend to be densely connected. This suggests that we could leverage the information of these dense subgraphs to group vertices in such a way that densely connected vertices are close-by in the representation. This is the starting point of this work. Several recent advances has led to efficient mining of these higher order structures (like cliques). We leverage the recently proposed TuránShadow algorithm [15] to obtain a sample of cliques in the graph to inform the visualization algorithm about the existence of these correlations, and to the best of our knowledge, none of the existing methods explicitly use this type of higher order structures in creating the visualization. The visualizations obtained

with our method have a clear improvement over an existing technique that does not involve incorporating higher order information about the nodes. We also outperform existing visualization techniques not just in how the graphs appear but even on a numerical metric that uses random walks to measure the local structure of a graph. We also demonstrate the ability of our algorithm to produce meaningful layouts of large graphs when other algorithms either fail to produce a result in a meaningful time frame, or produce a "hairball" type of visualization.

2 GRAPH LAYOUT ACCOUNTING FOR (N)KNOWN CLIQUES AND EDGES (GLANCE)

The goal of our algorithm, GLANCE, is to provide an aesthetically interesting two dimensional layout of an undirected graph $\boldsymbol{G}$ that can show structure and relevant clusters in the graph. The graphs we use are unweighted, undirected, and connected.

Recent work on large scale graphs provides evidence that incorporating higher order connections between nodes in a graph can improve standard tasks performed on graphs such as embedding, link prediction, or other forms of semi-supervised learning [11, 24, 29]. Our graph visualization method, too, takes advantage of higher order connections (cliques) in a graph to produce graph layouts and we make use of the recent TuránShadow [15] procedure to obtain random samples of cliques of varying sizes. We now briefly explain the three components: spectral embeddings, clique weightings, and t-SNE for more eigenvectors.

1. Spectral embeddings. Let G be an undirected graphs and let $\boldsymbol{A}$ be the adjacency matrix. A spectral embedding of G corresponds to using the generalized eigenvectors associated with the normalized Laplacian as coordinates of the graph. These can be computed from the symmetric normalized Laplacian matrix, $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$, where $\boldsymbol{D}$ is a diagonal matrix with $\boldsymbol{D}[i,i]$ equal to the degree of node i, and the matrix $\boldsymbol{I}$ is the identity matrix of matching dimensions. If $\boldsymbol{V}$ is an n-by-k matrix of the eigenvectors associated with the k smallest eigenvalues, then the spectral embedding is given by the matrix $\boldsymbol{X} = \boldsymbol{D}^{-1/2}\boldsymbol{V}$. (As an aside, we note that our results are largely equivalent with $\boldsymbol{V}$ instead of $\boldsymbol{X}$, but we use $\boldsymbol{X}$ for consistency.) The eigenvector associated with the smallest eigenvalue is well-known to be trivially related to the degree [9], so it is usually excluded from the embedding. To compute these eigenvectors, we use the ARPACK method available in Matlab, Python, and Julia [20].

This eigenvector embedding can be derived by trying to find coordinates for each node that minimize the weighted sum of edge-lengths, where the weight is inversely proportional to the degree. These embeddings are also called Fiedler embeddings due to the relationship with the Fiedler vector used for graph partitioning [12].

2. Clique weightings. We want to incorporate higher order information about cliques present in the graph by forming a new weighted version of the same graph that gives higher weights to edges that participate in cliques. We can use the *motif weighted adjacency matrix* formalism from [6] to accomplish this. In this case, the motifs are cliques of different sizes.

Full clique enumeration is computationally costly and so we achieve this by using the TuránShadow [15] sampling method to create a set of randomly selected cliques for varying clique sizes.

To build the weighted adjacency matrix, we introduce the clique-incidence matrix $\boldsymbol{C}_k$. This matrix has n columns and each row of this matrix is an indicator over the vertices of the cliques of size k found by the TuránShadow method. The order of the rows is arbitrary. Given the matrix $\boldsymbol{C}_k$, we use a modification of the weighted matrix

$$f(k)\, \boldsymbol{C}_k^T \boldsymbol{C}_k.$$

Intuitively, the value $[\boldsymbol{C}_k^T \boldsymbol{C}_k]_{i,j}$ indicates the number of times nodes i and j participate together in cliques of size k (among the randomly sampled k-cliques). Here, $f(k)$ is a weighting function to give a higher weight to large cliques compared to small cliques – because large cliques are rare. In our experiments we use a simple linear weighting function $f : k \to k$. Because this matrix can have diagonal entries that give weighted self-loops, which are not particularly interesting, we further remove any diagonal entries. So we define

$$\boldsymbol{W}_k = f(k)\,(\boldsymbol{C}_k^T \boldsymbol{C}_k) \text{ with all diagonal terms set to 0.}$$

Our new weighted version of the graph is created by summing over all k between 3 (for triangles) and K the largest clique we consider.

$$\hat{\boldsymbol{A}} = \boldsymbol{A} + \sum_{k=3}^{K} \boldsymbol{W}_k.$$

From the weighted version of the graph that accounts for cliques and edges, we then form the Laplacian matrix and find the eigenvectors corresponding to the second and third smallest eigenvalues. These two vectors are then used as the coordinates of the nodes in a two dimensional space for a clique-weighted Laplacian embedding.

3. Using more than two eigenvectors. Using only the two eigenvectors associated with the two smallest nonzero eigenvalues can be viewed as an embedding of each node in the graph into a two dimensional space. Often, and specially when the graph size is bigger, it is beneficial to use a larger embedding space of each node, and to do so, we use a larger number of eigenvectors associated with the smallest eigenvalues of the Laplacian matrix. We then use the dimensionality reduction technique, t-distributed Stochastic Neighbor Embedding (t-SNE) [22] on these eigenvectors to produce two dimensional coordinates for each node. As evidence that using more eigenvectors is helpful, we show a visualization of one network from the Facebook 100 dataset [27] with using 10 eigenvectors (top of Figure 1) and the two eigenvectors only (bottom of Figure 1).

The full GLANCE pipeline. Our method thus relies on accounting for edges, as well as higher order information in the graph in the form of cliques, and we call it "Graph Layout Accounting for (N)known Cliques and Edges" or GLANCE. We summarize the final pipeline in Algorithm 1. Note that the function *spectral-embedding(W, d)* returns the d eigenvectors of the Laplacian matrix of $\boldsymbol{W}$ associated with the smallest d eigenvalues. And the function *tsne-reduction(X,dims=2)* applies the t-SNE algorithm. When we draw graph layouts, we often find it useful to show regions where nodes have been collapsed to a near point. To expand these portions of the plot, we frequently plot the permutation induced by a vector x or y. That is, we assign new coordinates based on the rank of the node in the sorted list. This has the effect of spreading

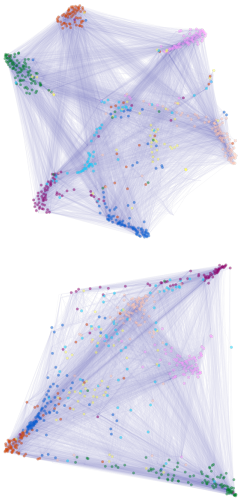

Figure 1: Visualizations of the same graph, the Caltech36 graph from [27]. The top figure uses an embedding space of size 10, and then uses a dimensionality reduction technique, t-SNE, to obtain the two dimensional coordinates. And the bottom figure uses the two eigenvectors associated with the smallest two eigenvalues of the Laplacian matrix. Colors in this graph refer to labeled information about which dorm a student in this graph belonged to, and the top figure provides a more apparent separation between colors.

out concentrated regions of the plot since it assigns a minimum difference in coordinates between nodes.

Algorithm 1: GLANCE. The matrix $\boldsymbol{A}$ is the adjacency matrix of the graph. The scalar K is maximum size clique to search for, we use $K = 20$. The function f is the weighting function to impose the importance of clique sizes, and we use a linear weighting function, $f(k) = k$. The scalar value d is the number of eigenvectors corresponding to the d smallest eigenvalues of the Laplacian matrix of $\boldsymbol{W}$. An implementation of our method is available at https://github.com/nassarhuda/GLANCE.

Input: $\boldsymbol{A}, K, f, d$
Output: x, y
$\boldsymbol{W} = \boldsymbol{A}$
for $k = 3, \ldots, K$ **do**
  $\boldsymbol{C}_k$ = TuránShadow-clique-sampling(A, k)
  $\boldsymbol{W}_k = f(k)\, \boldsymbol{C}_k^T \boldsymbol{C}_k$
  Set diagonal entries of $\boldsymbol{W}_k$ to 0.
  $\boldsymbol{W} \leftarrow \boldsymbol{W} + \boldsymbol{W}_k$
$\boldsymbol{X}$ = spectral-embedding(W, d)
x, y = tsne-reduction$(\boldsymbol{X}, dims = 2)$
return x, y

3 GRAPH VISUALIZATION METHODS

There is a large number of available tools for graph visualization online and scientifically backed methods in the literature. Nonetheless, these methods can often be a source of frustration when used to visualize graphs beyond a few nodes. For instance, methods, such as Tulip [3] or Hive [18] produce radial plots that are not relevant for understanding community structure or relations in a graph, and are mainly developed for domain-specific biological networks. Because we are interested in a generic algorithm here, we use three known generic algorithms that do not require domain specific knowledge about the nodes to compare GLANCE to. We provide a brief summary of each of them below.

Distributed Recursive Graph Layout (DRL). This method [23] falls under a class of graph drawing algorithms called Force-directed algorithms [16]. These algorithms tend to be aesthetically symmetric and their purpose is to have as few edge crossings as possible and to constrain all edges to have comparable lengths.

Large Graph Layout (LGL). This algorithm [1] applies a force-directed iterative layout guided by a minimal spanning tree of the graph. This algorithm was originally developed for large biological networks, and it can be particularly useful for tree-like graphs.

Node2Vec (N2V). One common way to visualize a network is via a network embedding that allows each node to have an *embedding* or vector that identifies it. What we use in GLANCE, is a form of spectral embedding. Here, we use a popular embedding technique, Node2Vec [14], which learns the feature representations for the nodes by simulating random walks around each node so that the feature representation of each node can capture the various connectivity patterns of each node.

Other methods. The diversity of graph visualization methods is large. A more closely related method is AROPE [31], which relies on higher order proximity between nodes. The key distinction between AROPE and GLANCE is that AROPE computes higher order distances in terms of path length (i.e. computing $\boldsymbol{A}^2, \boldsymbol{A}^3, \ldots$), and our method computes higher order relationships in terms of cliques. We tried using AROPE to produce network embeddings, but noticed that we need to fine tune parameters given the network at hand to produce meaningful visualizations. Here, we focus on comparing GLANCE to generic methods that require no changes in parameters given the network at hand.

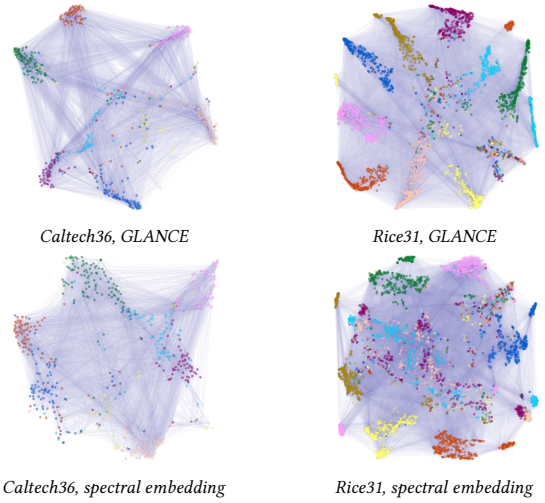

Figure 2: Graph visualization using GLANCE (top) and spectral embedding (bottom) for two networks from the FB100 dataset. Each node's color indicates its dorm label. The separation of nodes by colors is more apparent with GLANCE.

4 EXPERIMENTAL EVALUATION

4.1 Visual inspection of results

Here, we use the *Facebook100* dataset from [27], which contains 100 graphs of the Facebook social network from 100 U.S. universities.

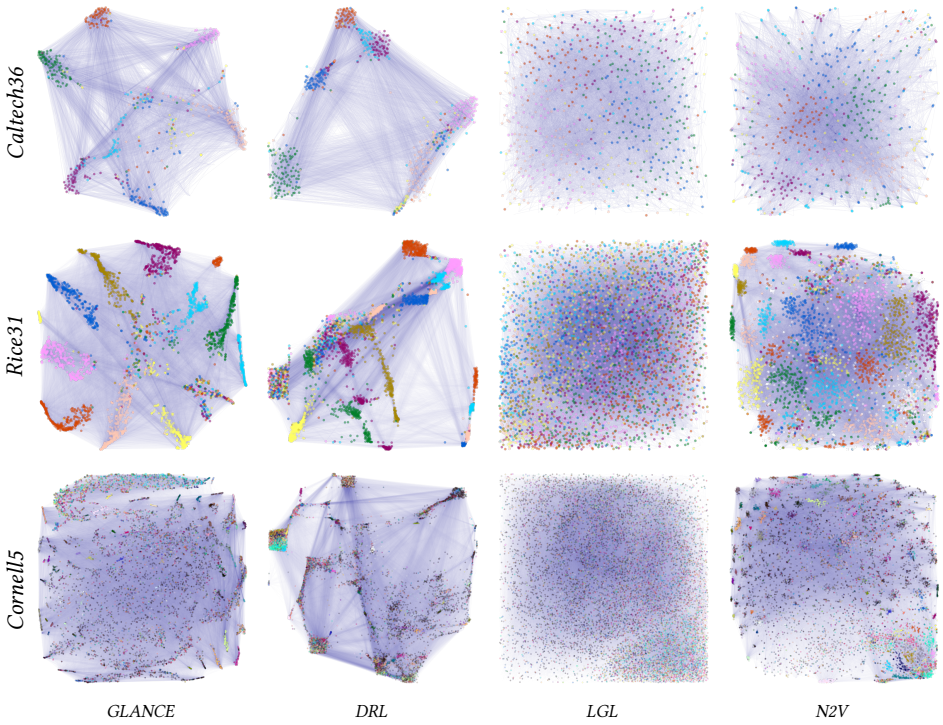

Figure 3: A visual catalog of three graphs from the Facebook100 dataset. From these, we find that the LGL algorithm provides the least visually interesting split of data. DRL, which is a method that performs a recursive force-directed visualization algorithm on clustered data, enforces a clustering on the graph even when clustering may not be apparent. This is less problematic for smaller datasets that have a clear split in clusters (top two rows of column two), but this becomes more problematic when there is no apparent cluster structure in the data (last two rows of column two). Arguably, N2V and GLANCE produce the best results.

Using higher order clique information improves the graph layout. Our main claim in this paper is that using a spectral embedding of a graph while accounting for higher order information (clique structure) improves the overall graph layout. Here, we show results of GLANCE, and GLANCE without clique sampling, which is just spectral embedding with t-SNE. (This can be accomplished by running GLANCE with $K = 2$). We show a visual illustration of our results in Figure 2 which shows the graph visualization results of GLANCE (top) and the visualization results of spectral embedding without cliques (bottom) on two Facebook100 networks (Caltech36 and Rice31). The color labels in these plots indicate dorm label colors, and the color separation is more apparent in the top row.

GLANCE provides useful visual outcomes when compared with other visualization methods. We now compare GLANCE to the three algorithms discussed in Section 3, on three FB100 datasets. We provide a catalog of the visualization results in Figure 3. In our experiments, we note that the behavior of these methods is similar among all the remaining datasets we tested. LGL often provided a visualization that spread the nodes evenly over the two dimensional space, but indicated little structure beyond this. DRL's visualization patters often revealed clusters of nodes that are very condensed. Node2Vec often generated a well spread of nodes with many small groups of nodes. And GLANCE often generated clusters of nodes and placed nodes that didn't seem to form clusters in the middle of the graph layout with a balanced spread. Figure 3 illustrates these conclusions on three datasets.

GLANCE is better than N2V at placing the small degree nodes. In the previous experiment, we saw that GLANCE and Node2Vec produced comparable results, and here, we compare them based on how well they place small degree nodes in the visualization. A general goal of graph visualization is placing similar nodes in close proximity to each other, and dissimilar nodes (nodes with large shortest path distance) far away from each other. A generally challenging set of nodes to place in a graph visualization is the set of low degree nodes because these nodes have very few connections, and thus we have very little information about them. To evaluate the performance of GLANCE and N2V on low degree nodes, we locate a random set of 50 degree-1 nodes in the FB100 graphs and show two results here.

In Figure 4, we show layouts with the degree-1 nodes highlighted in red and observe that Node2Vec locates all the degree-1 nodes together when they are not necessarily similar. In contrast, GLANCE places these nodes close to their connections in the graph (column 1 of Figure 4). Additional experiments show that this behavior appears in the rest of the Facebook100 dataset

GLANCE can avoid the "hairball" effect. We now use a large social network graph with 375K nodes from [28] and use our visualization algorithms on it. LGL fails on this graph, and takes a few days before it returns a nonnumerical result of all `NaN` coordinates. For the remaining algorithms, we show the visualizations in Figure 5. In this figure, N2V reveals the visualization that is closest to a "hairball" effect. DRL and GLANCE produced visualizations with clusters, but with the clusters produced by GLANCE being more coarse, and the clusters produced by DRL looking more condensed.

GLANCE identifies latent structure in Forest Fire Graphs. One of the most realistic network models is the forest fire graph generation model of Leskovec et al. [21]. The graph is initialized with a small graph, in our case, a single edge. For each vertex that arrives, the vertex chooses a hypothetical parent in the graph. This vertex is chosen uniformly at random. From the parent vertex, a

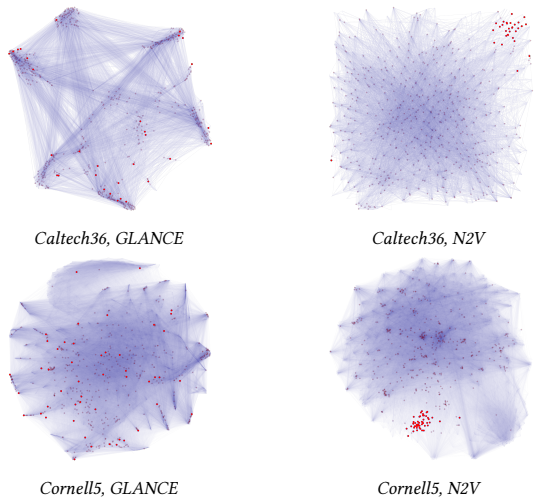

Figure 4: Node2Vec places all degree-1 nodes in close proximity to each other when they are not necessarily similar. In contrast, GLANCE places these nodes in close proximity to their similar nodes in the graph.

"fire" is started, which propagates through the graph like a forest fire: the vertices at the front of the fire select a geometrically distributed number of unburnt neighbors to burn based on a burning probability and add them to the front of the fire (old vertices are removed from the front). The process continues until there are no more vertices that can be added in this way (the fire burns out). The new vertex is connected to all vertices burned by the forest fire. This process is repeated for every new vertex to be added. Because the child nodes emulate much of the behavior of their parents, we would expect their neighborhoods to be highly correlated and hence, for the parent and child nodes to be placed close by in a visualization of the graph.

These parent child edges can thus be seen as the "ground truth" about which nodes should to be placed nearby. We run the visualization algorithms on these graphs and track the placement of parent-child (PC) edges vs. the non-parent-child (NPC) edges. Note that for an n-node graph constructed as above, there are exactly $n-2$ parent-child edges. Consequently, we compute the distance of each edge in the visualization and count how many of the smallest $n-2$ distances constitute parent-child edges. This yields a fraction between 0 and 1 where larger scores represent better alignment with the ground truth information. Note that this measure is actually the precision of using the $n-2$ smallest distance values to generate a result set.

We evaluate on a $10,000$ node graph generated with burning probability 0.48. This choice was created to give a graph with approximately $60,000$ undirected edges. We repeat the experiment 100 times and produce a kernel density estimate over the fraction of shortest edges that are parent-child edges, which is plotted in Figure 6. GLANCE has the best performance in that plot and consistently identifies around 55% of the total set of edges. Moreover, among all the methods, GLANCE is always the best method in each of the trials, in addition to having the best performance on average.

Note that just adding the clique weights to the Laplacian matrix (i.e. GLANCE without using t-SNE), results in an increase in performance in a large number of the trials. In contrast, adding t-SNE alone to the unweighted Laplacian improved the results much more consistently. But the combination of both in GLANCE gives the best result. A smaller preliminary run that used N2V showed similar or slightly better performance to GLANCE, although this took a long time. The other methods are uncompetitive on this task.

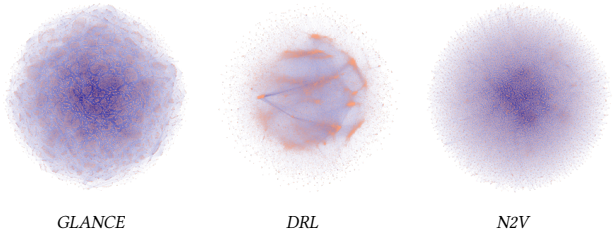

Figure 5: Visualization of a large Facebook graph with 375K nodes and more than a million edges from [28]. Among these, N2V produces the visualization least indicative of community structure. DRL produces a reliable visualization with well separated clusters. Our visualization produces a coarse type of visualization where communities can be seen separated, yet, there is an indication of each community size.

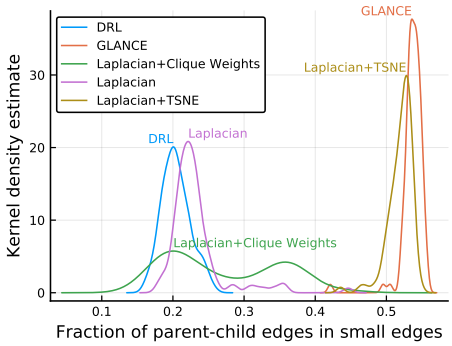

Figure 6: We synthetically generate a forest fire graph, which includes a set of ground truth edges that should be close. For each graph visualization method, we measure the fraction of the smallest edges that are in that ground truth set. This plot illustrates a kernel density estimate over these fractions on 100 distinct trials. GLANCE was computed with no parameter tuning. This figure clearly shows the benefit of using clique information, and the additional benefit of using t-SNE dimension reductions of Laplacian eigenvectors.

4.2 Numerical Evaluations

We now introduce a number of numerical metrics to evaluate the visualization methods we used in this paper. **Metric 1 (Closeness): Adjacent nodes in the graph are close geometrically.** For this metric, we examine each node u separately, by first obtaining the set

Table 1: Numerical evaluation of some of the Facebook100 datasets from [28]

Network	Closeness				Random walks				Spearman			
	GLANCE	DRL	LGL	N2V	GLANCE	DRL	LGL	N2V	GLANCE	DRL	LGL	N2V
American75	0.14	0.18	0.06	**0.25**	**0.26**	**0.26**	0.01	0.08	0.12	0.3	**0.57**	0.27
Cornell5	0.12	0.16	0.02	**0.23**	**0.76**	0.2	0.0	0.05	0.04	0.19	**0.51**	0.17
MSU24	0.08	0.15	0.01	**0.24**	**0.83**	0.23	0.0	0.03	0.0	0.17	**0.43**	0.16
Northeastern19	0.11	0.14	0.02	**0.27**	0.43	0.28	0.0	**0.65**	0.1	0.29	**0.52**	0.17
Northwestern25	0.17	0.19	0.05	**0.26**	**0.67**	0.23	0.0	0.15	0.01	0.27	**0.57**	0.14

Table 2: Numerical evaluation of some large graphs of different types.

Network	Closeness				Random walks				Spearman			
	GLANCE	DRL	LGL	N2V	GLANCE	DRL	LGL	N2V	GLANCE	DRL	LGL	N2V
com-Amazon, [30]	0.32	0.32	-	**0.35**	**0.66**	0.47	-	0.20	0.24	**0.40**	-	0.28
dblp-2010, [7] [8]	0.409	0.44	-	**0.46**	**0.50**	0.40	-	0.25	0.07	**0.24**	-	0.19
dictionary28, [5]	0.45	0.38	0.04	**0.54**	**0.82**	0.64	0.03	0.47	0.25	0.51	**0.62**	0.38
soc-Epinions1, [25]	0.15	0.10	0.004	**0.29**	**0.31**	0.102	0.001	0.28	0.02	0.30	**0.47**	0.17
web-ND, [2]	0.09	**0.15**	-	0.12	**0.13**	0.08	-	0.04	0.22	**0.29**	-	0.15
FBonemonth, [28]	0.15	0.17	-	**0.29**	**0.3**	0.21	-	**0.3**	0.04	**0.26**	-	0.18

$\Gamma(u)$, the set of nodes adjacent to u. We then find the set of closest $d = |\Gamma(u)|$ nodes to u geometrically by building a KDTree on the visualized data points and selecting the d nearest neighbors. The size of the intersection set of both sets normalized by d determines the quality of visualization of a single node. We repeat the same process for all nodes in the graph, and compute the mean value.

$$\text{score}(u) = \frac{|\Gamma(u) \cap NN(u, \text{degree}(u))|}{\text{degree}(u)}$$

$$\text{closeness-metric} = \frac{1}{n} \sum_{u \in G} \text{score}(u)$$

where $\text{NN}(u, d)$ returns the set of d nearest neighbors (geometrically) to a node u.

Metric 2 (Random walks): Nodes explorable via a random walk are close geometrically. In this metric as well, we examine each node u separately. For each node u, we run 100 random walks of size 20. Then, we find the nodes that appear in all 100 random walks, and place them in a set S. We then find the closest $|S|$ nodes to u geometrically. The intersection of both sets normalized by the size of S determines the score of u. Note that here, we can only evaluate nodes that return a non-empty set S. Let $S(u) = \{v \neq u | v$ appears in 100 random walks of size 20 started from node $u\}$

$$\text{score}(u) = \frac{|\text{S}(u) \cap \text{NN}(u, |\text{S}|)|}{|\text{S}|}$$

$$\text{randomwalk-metric} = \frac{1}{k} \sum_{u \in G, |S(u)| \neq 0} \text{score}(u)$$

Metric 3 (Spearman): Graph distance is correlated to geometric distance. For this metric, we examine multiple pairs of nodes at once. For a given graph, we use 10,000 randomly selected pairs of nodes. For each visualization method, we compute the graph shortest path distance as well as the Euclidean distance for every pair of nodes selected. We then find the Spearman correlation between both distributions.

Tables 1 and 2 show the results on some of the Facebook100 datasets and some other larger graphs respectively. In both tables, GLANCE is consistently one of the best methods under the random walks metric. LGL failed to produce coordinates for larger graphs, but whenever applicable, LGL was able to produce the best Spearman metric, while N2V and DRL performed best on the closeness metric. What this suggests is that other methods are perhaps more useful when network proximity in terms of degree of separation or relationship between graph distance and geometric distance is more important, whereas GLANCE is better at detecting nodes that would be deemed similar semantically by the random walks procedure (and not just nodes that are close to each other in terms of graph distance), arguably a more meaningful way of representing the graph when there is latent semantic structure.

4.3 Running time discussion

So far, we have explored all visualization methods from a visual and numerical perspective, and here, we summarize our findings on running time information. In Table 3 we show the running time in seconds for the three methods GLANCE, DRL, and N2V on two of the most challenging networks in terms of size (more than 100K nodes). Generally, DRL was the fastest, and GLANCE's bottleneck was the usage of t-SNE, but it still finished running in a reasonable amount of time compared to N2V. The LGL algorithm often failed to run on the large graphs.

Table 3: Running time results of some of some of the larger datasets used.

Network name	GLANCE	DRL	N2V
com-Amazon	11785.94	2466.78	104749.37
FBonemonth	12107.73	2043.69	149688.82

REFERENCES

[1] Alex Adai, Shailesh Date, Shannon Wieland, and Edward Marcotte. 2004. LGL: Creating a Map of Protein Function With an Algorithm for Visualizing Very Large Biological Networks. *Journal of molecular biology* 340 (07 2004), 179–90. https://doi.org/10.1016/j.jmb.2004.04.047

[2] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 1999. Internet: Diameter of the world-wide web. *nature* 401, 6749 (1999), 130.

[3] David Auber, Daniel Archambault, Romain Bourqui, Maylis Delest, Jonathan Dubois, Antoine Lambert, Patrick Mary, Morgan Mathiaut, Guy Melançon, Bruno Pinaud, Benjamin Renoust, and Jason Vallet. 2017. TULIP 5. In *Encyclopedia of Social Network Analysis and Mining*, Reda Alhajj and Jon Rokne (Eds.). Springer, 1–28. https://doi.org/10.1007/978-1-4614-7163-9_315-1

[4] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 2009. Gephi: An Open Source Software for Exploring and Manipulating Networks. (2009). http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154

[5] Vladimir Batagelj and Andrej Mrvar. [n.d.]. Pajek datasets. http://vlado.fmf.uni-lj.si/pub/networks/data/

[6] Austin Benson, David F. Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. Submitted. *Science* 353, 6295 (2016), 163–166. https://doi.org/10.1126/science.aad9029

[7] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. 2011. Layered Label Propagation: A Multiresolution Coordinate-free Ordering for Compressing Social Networks. In *Proceedings of the 20th International Conference on World Wide Web* (Hyderabad, India) *(WWW '11)*. ACM, New York, NY, USA, 587–596. https://doi.org/10.1145/1963405.1963488

[8] P. Boldi and S. Vigna. 2004. The Webgraph Framework I: Compression Techniques. In *Proceedings of the 13th International Conference on World Wide Web* (New York, NY, USA) *(WWW '04)*. ACM, New York, NY, USA, 595–602. https://doi.org/10.1145/988672.988752

[9] Fan R. L. Chung. 1992. *Spectral Graph Theory*. American Mathematical Society.

[10] Darren Edge, Jonathan Larson, Markus Mobius, and Christopher White. 2018. Trimming the Hairball: Edge Cutting Strategies for Making Dense Graphs Usable. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 3951–3958.

[11] Alessandro Epasto and Bryan Perozzi. 2019. Is a Single Embedding Enough? Learning Node Representations that Capture Multiple Social Contexts. *CoRR* abs/1905.02138 (2019). arXiv:1905.02138 http://arxiv.org/abs/1905.02138

[12] Miroslav Fiedler. 1989. Laplacian of graphs and algebraic connectivity. *Banach Center Publications* 25, 1 (1989), 57–70. http://eudml.org/doc/267812

[13] Nils Gehlenborg, Seán I. O'Donoghue, Nitin S. Baliga, Alexander Goesmann, Matthew A. Hibbs, Hiroaki Kitano, Oliver Kohlbacher, Heiko Neuweger, Reinhard Schneider, Dan Tenenbaum, and Anne-Claude Gavin. 2010. Visualization of omics data for systems biology. *Nature Methods* 7, 3 (2010), S56–S68. https://doi.org/10.1038/nmeth.1436

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

[15] Shweta Jain and C. Seshadhri. 2017. A Fast and Provable Method for Estimating Clique Counts Using TuráN's Theorem. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 441–449. https://doi.org/10.1145/3038912.3052636

[16] Stephen G Kobourov. 2012. Spring embedders and force directed graph drawing algorithms. *arXiv preprint arXiv:1201.3011* (2012).

[17] Martin Krzywinski, Inanc Birol, Steven JM Jones, and Marco A Marra. 2011. Hive plots—rational approach to visualizing networks. *Briefings in Bioinformatics* 13, 5 (12 2011), 627–644. https://doi.org/10.1093/bib/bbr069 arXiv:http://oup.prod.sis.lan/bib/article-pdf/13/5/627/1147399/bbr069.pdf

[18] Martin Krzywinski, Inanc Birol, Steven JM Jones, and Marco A Marra. 2011. Hive plots—rational approach to visualizing networks. *Briefings in Bioinformatics* 13, 5 (12 2011), 627–644. https://doi.org/10.1093/bib/bbr069 arXiv:http://oup.prod.sis.lan/bib/article-pdf/13/5/627/1147399/bbr069.pdf

[19] Jérôme Kunegis, Stephan Schmidt, Andreas Lommatzsch, Jürgen Lerner, Ernesto William De Luca, and Sahin Albayrak. 2010. Spectral Analysis of Signed Graphs for Clustering, Prediction and Visualization. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*. 559–570. https://doi.org/10.1137/1.9781611972801.49

[20] R. B. Lehoucq and D. C. Sorensen. 1996. Deflation Techniques for an Implicitly Restarted Arnoldi Iteration. *SIAM J. Matrix Anal. Appl.* 17, 4 (1996), 789–821. https://doi.org/10.1137/S0895479895281484

[21] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (Chicago, Illinois, USA) *(KDD '05)*. ACM, New York, NY, USA, 177–187. https://doi.org/10.1145/1081870.1081893

[22] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[23] Shawn Martin, W. Michael Brown, and Brian N. Wylie. 2007. Dr.L: Distributed Recursive (Graph) Layout, Version 00. https://www.osti.gov//servlets/purl/1231060

[24] Huda Nassar, Austin R. Benson, and David F. Gleich. 2019. Pairwise Link Prediction. *CoRR* abs/1907.04503 (2019). arXiv:1907.04503 http://arxiv.org/abs/1907.04503

[25] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. 2003. Trust Management for the Semantic Web. In *Proceedings of the Second International Conference on Semantic Web Conference* (Sanibel Island, FL) *(LNCS-ISWC'03)*. Springer-Verlag, Berlin, Heidelberg, 351–368. https://doi.org/10.1007/978-3-540-39718-2_23

[26] Matthew Suderman and Michael Hallett. 2007. Tools for visually exploring biological networks. *Bioinformatics* 23, 20 (08 2007), 2651–2659. https://doi.org/10.1093/bioinformatics/btm401 arXiv:http://oup.prod.sis.lan/bioinformatics/article-pdf/23/20/2651/16858965/btm401.pdf

[27] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. 2011. Social Structure of Facebook Networks. *CoRR* abs/1102.2166 (2011). arXiv:1102.2166

[28] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, and Ben Y. Zhao. 2009. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems* (Nuremberg, Germany) *(EuroSys '09)*. ACM, New York, NY, USA, 205–218. https://doi.org/10.1145/1519065.1519089

[29] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Pratim Talukdar. 2018. HyperGCN: Hypergraph Convolutional Networks for Semi-Supervised Classification. *CoRR* abs/1809.02589 (2018). arXiv:1809.02589 http://arxiv.org/abs/1809.02589

[30] Jaewon Yang and Jure Leskovec. 2012. Defining and Evaluating Network Communities Based on Ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics* (Beijing, China) *(MDS '12)*. ACM, New York, NY, USA, Article 3, 8 pages. https://doi.org/10.1145/2350190.2350193

[31] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-Order Proximity Preserved Network Embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.