Beyond community detection on undirected, unweighted graphs

Vipul Pandey vpandey1@stanford.edu
Juthika Dabholkar juthika@stanford.edu
Rex Kirshner rbk@stanford.edu

November 17, 2011

1 Abstract

Community detection has been rigorously explored for undirected and unweighted graphs, and modularity has emerged as the most prevalent metric for verification and optimization. However, we have shown that modularity is an inaccurate measure of community structure for more complicated graphs, in particular directed networks. We came up with a metric based on the ratio of cycles within and across communities. In this paper, we also propose an extension of this metric to directed weighted graphs and discuss further extension to directed signed networks.

Keywords - Community detection, directed graphs, shortest cycles, weighted graphs, signed graphs

2 Introduction

Community detection is a very important problem in network analysis. The primary metric for identification is modularity (number of edges falling within groups compared to the expected number in an equivalent random network). However, this metric only generates and evaluates quality communities in unweighted, undirected and unsigned networks. While there have been proposals and extensions of modularity to account for other types of graphs, there are no good solutions to the problem.

This paper defines a new metric for community detection, based on the number of cycles within a community compared to the number of cycles across ommunities. Intuitively, we are measuring the speed at which information travels back to the source via nodes within a community compared to the speed at which information leaves a community and comes back via nodes outside the community. Good communities should have quick speeds within the community, but have low speeds ouside of the community.

We also discuss extensions to directed weighted and directed signed, although we do not rigorously explore these avenues, instead leaving them for future work.

3 Prior Work

Community detection on unweighted, undirected, unsigned graphs has been well researched on and various algorithms and measures for optimization and verification of community structure have been identified. The technique was introduced by Girvan and Newman using a betweenness metric in [6]. They also suggested the notion of modularity for quantifying the effectiveness of a partition of the graph. The metric of modularity (Q) proposed by Girvan and Newman in [6], is defined as follows:

$$Q = \Sigma_i(e_{ii} - a_i^2) \tag{1}$$

Here e_{ij} is the fraction of edges in the network that connect vertices in partition i to those in partition j, and

$$a_{ij} = \Sigma_i e_{ij}$$

In [3], the validation measure of modularity itself was optimized using spectral techniques to arrive at effective splits. It was proposed in both [3] and [6] that these techniques can be trivially extended to directed graphs and we can get the community structure by just ignoring directionality of edges.

Modularity has has been shown to be a bad metric for community detection for directed graphs because it throws away edge direction information and computes the modularity values based on the undirected

version on the graph. This removes valuable information, and, especially in the cases we are interested in, creates bad partitions. Besides, it is not trivial to extend the metric to directed graphs since the adjacency matrix for directed graphs is asymmetric. However, Leicht and Newman [1] propose a technique for community detection in directed grpahs by systematically transforming the directed (non symmetric) adjacency matrix to a symmetric matrix and a slight variation of the spectral methods used for the undirected case in [3]. They define the directed version of modularity in terms of a suprise value where a directed edge from a node of high in-degree to a node of high out-degree is considered a bigger surprise and makes a bigger contribution to the modularity score. The modified version of modularity is defined as follows:

$$Q = (1/2m)s^T Bs \tag{2}$$

Here, s is the vector whose elements define which community each node belongs to, and B is the modularity matrix with elements $B_{ij} = A_{ij} - (k_i^{in} k_j^{out})/m$. Since this matrix B is not symmetric, they add B^T to B creating a symmetric matrix. Thus, the new definition is

$$Q = (1/4m)s^T(B + B^T)s \tag{3}$$

This can be solved using a similar technique as the spectral technique proposed in [3].

4 Hypothesis

We believe that the same techniques and measures as have been proposed for undirected graphs are not as intuitive when we apply them to directed, weighted or signed graphs. The intuition for this is explained as follows:

In the directed case, consider the example (Figure 1) where one node accepts connections from a community, but does not give any back. Without taking direction in to account, Z belongs in the community with A, B and C. However, this does not make intuitive sense. Example: a group of friends follows Obama on Twitter.

4.1 Initial Experiments

For our initial experiments to verify our hypothesis, we created a sample graph of 44 nodes, 300 edges. The fictitious graph consisted of 4 subgraphs (of 10 nodes each) built on the small-world model, with each node having an egde pointing to its neighbors and neighbor of neighbors. Additionally, within each subgraph, we added 10 short range random links.

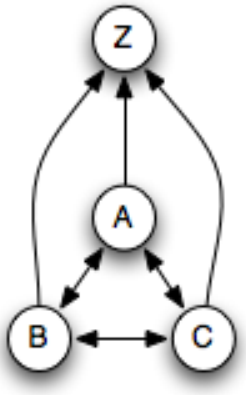

Figure 1: Communities in directed graphs

Beyond these 4 communities, we added 4 individual nodes (hubs), which had a large in-degree but an out-degree of 0. Each of these nodes had links pointing to them from a random set of 25 nodes from the earlier 40 nodes.

Figures 2 and 3 depict the structure of the fictitious graph and its subgraph (built on the small world model).

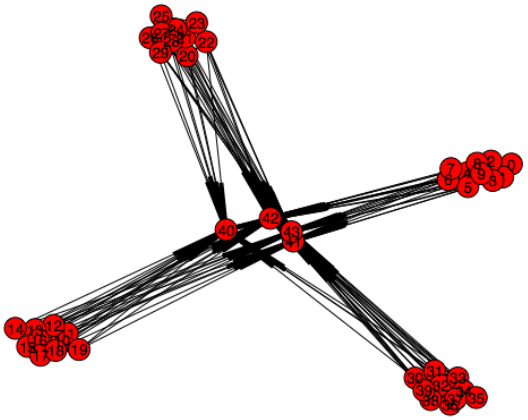

Figure 2: Fictitious directed graph consisting of 4 communities and 4 hubs

As can be seen intuitively, there are 4 tightly knit communities in this network. Additionally, the hubs should be 4 singleton communities, since they do

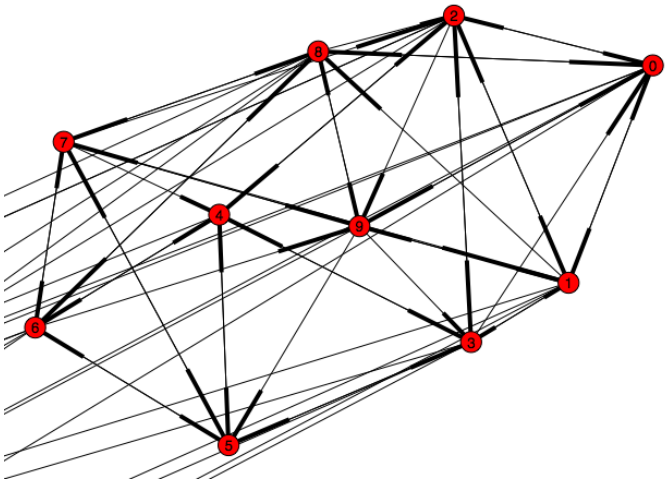

Figure 3: Small world model based 10 node directed subgraph

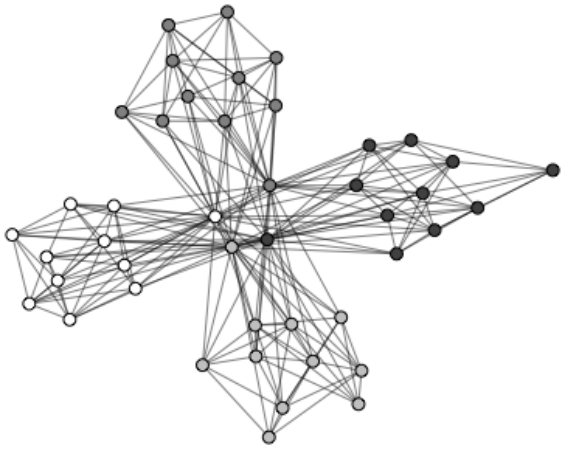

Figure 4: Four communities detected on the undirected graph

not "follow" anyone. We assigned each node in the network a community label based on this intuition. Nodes 0-9 were in one community, 10-19 in the second, 20-29 in the third, 30-39 in the fourth and the nodes 40-43 were individual singleton communities.

4.2 Observations

To run our existing community detection algorithms on this graph, it first needed to be transformed to an undirected graph, since the present community detection algorithms do not take directionality into account. Figure 4 shows the outcome of community detection on the corresponding undirected graph of our fictitious graph.

As can be seen from Figure 4, the network is partitioned into 4 communities - the 10 node subgraphs are correctly partitioned and each of the hubs is now assigned to one of these 4 partitions, which does not tally with our intuition above. Thus we can see that directionality is indeed an important signal for community detection, and we cannot trivially ignore it if we wish to get intuitive results.

For the purpose of complete analysis, we also ran the same algorithm on a G(n, m) random graph having the same number of nodes and edges. Figure 5 shows the community structure (or the lack thereof) for this random graph.

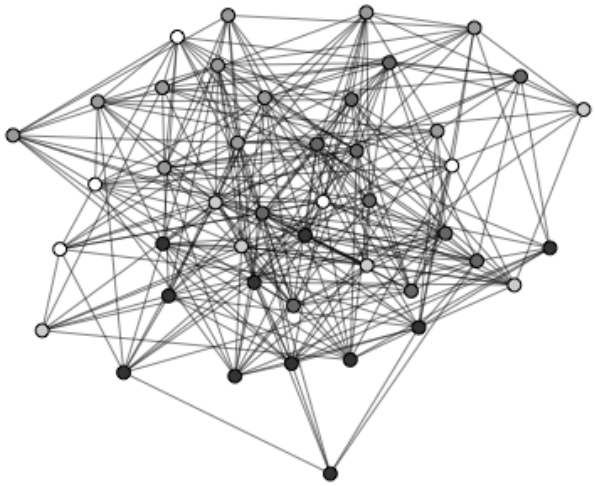

Figure 5: Community structure for the random graph

4.3 Analysis

As we saw visually, the community structure obtained by ignoring directions does not tally with our intuition. We also analyzed the graphs using modularity that quantifies the effectiveness of community structure in undirected graphs. We observed that this metric is also not very good at quantifying the effectiveness of community partitions for directed graphs.

Table 1 shows the value of the modularity measure on our data. As expected, the modularity score for the undirected random graph is low, far from optimal. However, it is observed that the modularity score for the partition returned by the community

detection algorithm (with just 4 communities) has a higher score than that for our ground truth partition (with 4 communities of 10 nodes each and 4 singleton communities).

Graph	Partition	Modularity
Directed	Ground truth	0.379
Undirected	Algorithm output	0.447
Random	Algorithm output	0.156

Table 1: Modularity scores

Ideally, our metric should be such that the best partition should have the most optimal score. Thus we can infer that the modularity measure used for identifying or evaluating community partitions in undirected graphs does not hold for the case of directed graphs. Therefore, most of the community detection algorithms that try to optimize this measure would also fail to give us the expected partitions.

We also experimented with the directed version of modularity proposed by Leicht and Newman in [1]. Even this metric did not give us the expected (higher) value for the ground truth as compared to the partition returned by the modularity based algorithm on the corresponding undirected version of the graph. This can be explained by considering the case when there is a single edge between two singleton communities with a high in-degree each (similar to the ones above). Since (based on their degree distribution) it is unlikely that there is an outgoing edge from one of these nodes, it would result in a high contribution to the overall modularity score tending to combine these two singleton communities into a single community. However, this does not intuitively make sense, since there is an edge from one of the hubs to the other in one direction only and not in the reverse. For the two nodes to be close enough to be in a single community, we believe that there needs to be a link in the opposite direction as well, without which the nodes should still be deemed independent of each other at a community level.

5 Intuition

Modularity uses the density of edges within a community, but it does not use the extra information given to us by direction. We try to capture this information in the directionality by following paths. Every arbitrary path does not give useful information, so we decided to focus on cycles. Using cycles we were able to capture the inherent structure of the community by concentrating on paths that return information back to the sender. Intuitively, good communities should be structured such that information sent from any node to the community should come back relatively quickly, and any information sent out should take a long time to come back, if it comes back at all.

The first important contribution to our metric is cycle length within a community. As previously stated, communities should be structured so that information within can quickly reach its source. This is fairly consistent with modularity, as denser areas will be more likely to have short cycles; our measure is just a stronger requirement.

The second important contribution is the length of each cycle that starts within a community and leaves it. A good community structure should have all related nodes within, and any information sent out is rare and infrequent. Therefore, as these paths pass through many other hubs, the lengths are much longer. Again, this follows with the ideas laid down by modularity, as the low density of the edges will cause long paths, but is a stronger requirement, as edges must not only exist but move in the right direction.

Therefore, as we want to optimize average shortest cycle lengths within and average shortest cycle lengths outside, we naturally moved to ratio of these two measures. Thus, as average intra cycle length goes down and average inter cycle length goes up, our score decreases to a minimum of zero (as neither will be negative). Now, we have a measure with a lower bound that we can use to rate communities: smaller is better.

There are two methods of gathering cycle length data before averaging. The first is to gather the shortest cycle for each node, while the second is to pick an edge, and find the shortest path from the destination to the source. The later is actually the better for our metric, as nodes with one cycle of length 2 and ten cycles of length 50 is actually not the best fit in the community. Calculating just the shortest cycle for each node loses this important information. While holding these long cycles against a densely connected community may seem detrimental, the short cycles will average the long outliers out.

6 Algorithm

6.1 Overview

We build on our intuition of finding cycles within and outside the community to evaluate the validity of a given split of a graph. We take one community at a time and try to find shortest cycle within. For this we ignore every edge that comes in or goes out of the community and consider only intra-community edges. Then we iterate over the the outgoing edges of each node (to other nodes within the community) and find the shortest path from the destination node of the edge back to the source node. That gives us the shortest cycle starting from that edge. We thus find all the shortest cycles within a community. We repeat this process for each community to get all the intra-community cycles in the entire graph and calculate average intra-community cycle length.

We carry out a similar exercise for calculating average length of the cycles going outside communities taking one community at a time and iterating over all its nodes. We then iterate over all the outgoing edges from the node to a node of some other community and find the the shortest path back from the destination node, thus completing a cycle. This gives us a shortest cycle for an edge going out of the community. We then calculate the average of all the shortest cycles computed. Our metric score is a ratio of average intra-community cycle length to average length of cycles traveling outside communities. It can be clearly seen that better the partitioning, lower will be the score of our metric. This is because we expect the intra-community cycle length to drop as we get better communities and the extra-community cycle lengths to increase. This is opposite to other measures like modularity where higher score means better partitioning.

6.2 Pseudo-code

Figure 6 illustrates our algorithm by way of a pseudo code.

6.3 Handling infinite cycle lengths

One thing to note above is that while finding cycles both within and without, we may come across destination nodes of an edge with no path back to the source node, thus representing cycles with infinite lengths. We represent them with -1 in our lists and we have different ways to handle them. One way is to remove all -1s from the list and take the average, but that would be incorrect as we can not ignore them completely. There could be many infinite length cycles and we need to know about them to find the efficacy of the partitioning. Below are some other ways of handling -1s that we considered:
(a) Replace -1s with a constant - a big enough value that represents infinite length cycles aptly for the graph but not so big that the proper cycles are underrepresented. Although it does not incorporate any property of the graph, making it invariant to the type of graph, it gave the best results and was independent of certain edge cases as well (for eg. cases where none of the outgoing paths make their way back to the source).
(b) Replace -1s with a multiple of the max cycle length detected for each category (within and without). This is a good option as it does incorporate a property of the graph, although it doesn't work well in cases where there are many infinite length cycles and a handful of short length ones, which will give a very wrong view.
(c) Ignore all the -1s and calculate the average of the remaining elements in both the lists. Then factor in the count of -1s in some way to penalize the score. An increase in the number of -1s should worsen the score. The issue with this approach is that it treats all graphs of all sizes in the same way. e.g. metric score for 100 infinite length cycles in a 100 node graph should be different from another graph of 1million nodes having same number of infinite cycles.
(d) Ignore all the -1s and calculate the average of the remaining elements in both the lists. Then factor in the proportion of -1s in some way to penalize the score. A higher proportion of -1s should worsen the score.

We report option (a) in our experiments as it seemed to be the most promising.

7 Runtime Complexity

The upper bound of our metric is only bound by the single source shortest path algorithm. As this must be computed for the destination node of each edge, and single source shortest path is bounded by $O(n^2)$, verification of community structure using our metric runs in worst case $O(mn^2)$. One optimization that can be done in our process is to compute all pair shortest paths once for the entire graph and use it

```
function getCycleRatio():
Input : Graph G
        listCycleLengths = list()
        for each community of G :
                for each node i in the community :
                        for all edges of the node going out to another node j of the same community :
                                listCycleLengths.add(shortest path from j to i + 1 )
        avgWithin = average(listCycleLengths)

        listCycleLengths = list()
        for each community of G :
                for each node i in the community :
                        for all edges of the node going out to another node j of some other community :
                                listCycleLengths.add(shortest path from j to i + 1 )
        avgWithout = average(listCycleLengths)

        return avgWithin/avgWithout
```

Figure 6: Pseudo-Code

for multiple subsequent calculations and iterations, which would drop the overall time complexity to $O(n^3)$.

8 Observations and Analysis

We generated fictitious graphs of three different sizes (approximately 40, 250 and 1000 nodes), having communities of various structures, and labeled them with the intuitive ground truth partitions for each node. We also ran a modularity based community detection algorithm on these generated graphs and obtained a different partition, ignoring edge directions. We compared the effectiveness of these two splits according to three measures: Girvan-Newman (undirected) Modularity from [3], Leicht-Newman (directed) modularity score from [1] and our metric based on directed cycles within and across communities.

8.1 Observations

We describe our findings for each of the different types of generated fictitious graphs in the following sections [1]:

8.1.1 Small World Communities

We generated fictitious graphs where each of the communities was inherently a small world network. These graphs were created by initially generating several small communities where all the nodes we first connected in the form of a bi-directional ring structure, following which each node was connected to its neighbor's neighbor. Additionally, each of the communities had some random links connecting the nodes within the community. The number of nodes per community were randomly chosen based on a triangular distribution. The structure of one of the communities is shown in Figure 7. Finally, each of the nodes in the network were chosen for inter-community links with a low probability and then were randomly connected to one of the nodes in the network, not in the same community.

Table 2 summarizes our observations for this type of graph.

[1] In all tables, GN = Girvan Newman Modularity on undirected graphs, LN = Leicht Newman Modularity for directed graphs and CY = our measure based on the ratio of average shortest cycle lengths within and outside communities.

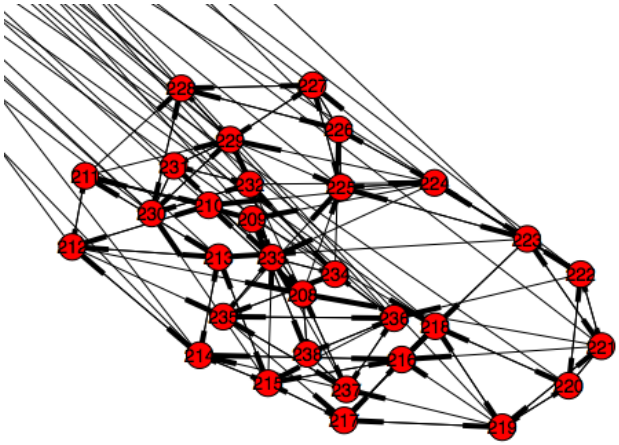

Figure 7: Community based on Small World Model

Graph	**Metric**	**Modularity based partition**		**Ground Truth partition**	
Nodes, Edges		No. of Comm	Score	No. of Comm	Score
44, 312	GN LN CY	4	0.412 0.481 0.195	5	0.352 0.447 0.043
256, 1576	GN LN CY	11	0.682 0.751 0.081	12	0.666 0.745 0.048
1012, 5714	GN LN CY	18	0.776 0.832 0.069	19	0.766 0.828 0.054

Table 2: Scores for Small World Communities

8.1.2 Erdos-Renyi Communities

In this case we generated graphs of multiple sizes each having multiple communities with each community in itself being an ErdosRenyi network. We generate Stochastic Kronecker Graphs using the following initiator matrix :

$$\begin{pmatrix} 0.7 & 0.7 \\ 0.7 & 0.7 \end{pmatrix}$$

Each such graph represents a community, and edges across communities are added with probability chosen from a triangular distribution of low values, thus sparsely connecting the communities. For each graph we generated what we call a *super community* which is a complete graph of 4-5 nodes with no links going out of that community and almost all the nodes of other communities having an outgoing edge pointing to almost each member of *super community*. The number of nodes per community and the number of communities was again chosen using a triangular distribution. Figure 8 shows such a community.

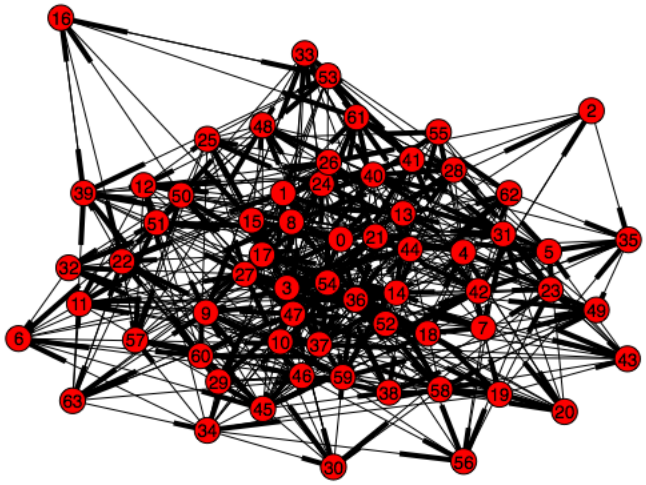

Figure 8: Community based on Erdos Renyi structure

Graph	**Metric**	**Modularity based partition**		**Ground Truth partition**	
Nodes, Edges		No. of Comm	Score	No. of Comm	Score
53, 402	GN LN CY	3	-0.04 0.255 0.404	4	-0.01 0.151 0.071
266, 5555	GN LN CY	11	-0.022 0.205 0.453	12	-0.024 0.064 0.087
956, 41693	GN LN CY	18	0.016 0.137 0.41	19	-0.013 -0.03 0.114

Table 3: Scores for Erdos Renyi Communities

8.1.3 Core-Periphery communities

In this case we generated graphs having multiple communities, with each community having a dense

core and a light periphery. As before, we generated Stochastic Kronecker Graphs using the following initiator matrix :

$$\begin{pmatrix} 0.99 & 0.8 \\ 0.8 & 0.5 \end{pmatrix}$$

Each such graph represents a community, and the nodes in only the core of different communities are connected to each other with a probability chosen from a triangular distribution of much lower values, thus sparsely connecting the communities. As above, for each graph we generated what we call a *super community* which is a complete graph of 4-5 nodes with no links going out of that community and almost all the nodes of other communities having an outgoing edge pointing to almost each member of *super community*. The number of nodes per community and the number of communities was again chosen using a triangular distribution. Figure 9 shows such a community. We summarize the results of our experiments in Table 4.

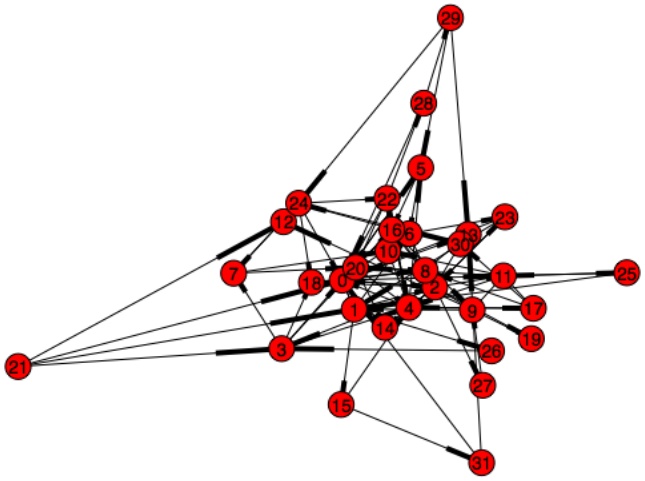

Figure 9: Community based on Core Periphery structure

8.1.4 Random graphs with unidirectional edges between communities

1. **Forward-many graph**
   The forward many graph is constructed first by creating n communities by randomly generating edges between all nodes in both directions with a probability of p. Then, edges are generated with the same probability p from nodes in the community i to communities i + 1, i + 2, , n - 1, n. In this way, communities are always connected in a single direction, and nodes in later communities will have the most connections. This case is very interesting because when it is collapsed in to a undirected graph, as necessary for community detection based on modularity, all ground truth community structure is lost, and the graph looks like a random undirected graph. Thus, there is a major difference between the ground truths and the modularity based communities.

Graph	**Metric**	**Modularity based partition**		**Ground Truth partition**	
Nodes, Edges		No. of Comm	Score	No. of Comm	Score
52, 503	GN LN CY	3	0.008 0.349 0.289	4	0.042 0.32 0.069
276, 9117	GN LN CY	6	0.011 0.216 0.369	8	-0.025 -0.053 0.062
1260, 75341	GN LN CY	10	0.048 0.236 0.335	21	0.051 0.151 0.109

Table 4: Scores for Core Periphery Communities

Graph	**Metric**	**Modularity based partition**		**Ground Truth partition**	
Nodes, Edges		No. of Comm	Score	No. of Comm	Score
40, 499	GN LN CY	4	0.113 0.116 0.77	5	0.092 0.107 0.047
250, 18685	GN LN CY	5	0.072 0.084 0.113	5	0.071 -0.083 0.049
986, 275636	GN LN CY	6	0.028 0.028 0.949	17	0.027 0.034 0.049

Table 5: Scores for Forward-many graph

2. **Forward-once graph**

The forward once graph is similar to the forward many, except each community i is only connected to community i + 1, instead of all communities after. Again, this is interesting because ground truth community structure is lost without direction, as all allowable edges are placed with equal probability.

Graph	**Metric**	**Modularity based partition**		**Ground Truth partition**	
Nodes, Edges		No. of Comm	Score	No. of Comm	Score
40, 349	GN LN CY	3	0.334 0.359 0.592	4	0.253 0.305 0.047
250, 11308	GN LN CY	3	0.357 0.415 0.319	5	0.278 0.344 0.049
986, 55757	GN LN CY	4	0.617 0.635 0.462	17	0.385 0.455 0.05

Table 6: Scores for Forward-once graph

8.2 Analysis

In all the above cases, we observed that Girvan Newman doesnt do a good job of getting to the ground truth. Also the modularity as proposed by Girvan Newman [3] drops when we go from suboptimal communities to ground truth. A similar behaviour is exhibited by the modularity metric as proposed by Newman [1] for directed networks. On the other hand our metric shows a clear improvement when moving from suboptimal partitions to ground truth.

9 Analysis on weighted directed networks

Extension of this approach of evaluating communities based on the number and length of cycles can be extended to weighted graphs with relative ease. The weights can be incorporated by taking their reciprocal for the corresponding edge, and adding that value to the path, instead of incrementing the path length by one as we did for unweighted directed graphs. This will make sure that a cycle with higher edge weights is considered shorter than an equal length cycle with relatively lighter set of edges. We claim that this will help us in separating communities that are densely connected within by edges of high weights, and also densely inter-connected but with edges of lighter weights.

Our experiments confirm our claim that our metric is applicable to this configuration. We generated graphs with multiple communities where a community is as densely intra-connected as interconnected. However, each edge within the community is assigned a weight around 0.85 and the edges between the communities are assigned a weight of around 0.15. Table **??** describes our results on a fictitious graph of around 500 nodes. We compare the performance of our metric on the divisions as generated by the Girvan-Newman algorithm, which does not separate the communities based on weights, and the ground truth and confirm that our metric gives a far better values for the ground truth.

Graph	**Metric**	**Modularity based partition**		**Ground Truth partition**	
Nodes, Edges		No. of Comm	Score	No. of Comm	Score
516, 26240	GN LN CY	4	0.011 0.216 0.359	9	-0.05 -0.035 0.1

Table 7: Scores for Weighted graph

Even though the average cycle lengths are comparable within and without communities, considering the reciprocal of weights condenses the within cycle lengths and expands the ones without thus creating a clear demarcation.

10 Extension to signed directed networks

Extending our metric of community verification to signed networks is actually much more complicated than it first appears. Our metric depends on the length of cycles, making it hard to account for negative edges. There are a few obvious ways include these types of graphs, but each have a major flaw.

The first is to remove all negative edges from the graph, and run the metric on only positive

edges. This method has two advantages. First, it is the simplest way to deal with the negative edges. Second, it can solve many of the intuitive problems we had with signed graphs. Two communities connected internally with positive edges and to each other with negative edges will provide a better score as two communities than as one. However, the reason this method is not optimal is because it throws away information, which is the same reason we found modularity to be sub optimal for community detection in directed graphs. For example, consider a ring of nodes connected by positive edges, and then connected within by negative edges. This would make a decent community without the negative edges, but the negative edges show it is a community full of animosity, and therefore a bad community.

Another method of dealing with negative edges is to count them as negative contributions to cycle lengths. This has the advantage of taking in to account negative signs. However, this ultimately does not make sense. A cycle of length 10 containing 4 negative links would be counted as a cycle of length 2, which is a better cycle than counting them as positive or not counting negative edges at all. Cycles that contain more negative links than positive further complicate this method. This would lead to negative cycle lengths, something undefined in our metric.

Perhaps the most promising method is to count positive and negative cycles separately. This would change the formula to look something like this (avg positive cycle length within + avg negative cycle length within)/(avg positive cycle length without + avg negative cycle length without). This method has the advantage of accounting for both positive and negative cycles in the proper way. However, the major drawback of dealing with signed edges this way is that it does not provide a clear and intuitive way to deal with both positive and negative edges, something that is common in most signed graphs.

Needless to say, extending our metric to signed graphs is not as simple as it is for weighted graphs. However, the potential is defintely there. Due to time constraints we were not able to explore this avenue further, but we have looked at a few options, and laid the groundwork for future study.

11 Limitations and Special Cases

There are some limitations to our approach, some of which can be overcome by more research, while others have to be the considerations while applying our metric. The runtime complexity of the algorithm, which is $O(n^3)$, is rather high and prohibitive for very large graphs. Even though it is comparable to Girvan-Newman for undirected graphs, we believe it could be too high for many real world networks.

We need to explore the special conditions and corner cases further, and come up with ways to handle them. One example would be that of singleton communities with a single node with no edges going out in which case there is no path to explore for cycles and the metric would fail to give a genuine value. The other example is that of taking the entire graph as one community, which again poses the same problem wherein there is no outbound edges from that one community thus breaking our metric. We believe these cases can be easily handled by dealing with them on a case by case basis.

Another area to explore is to see if optimization of our metric leads to minuscule communities in the quest to minimize intra-community cycle lengths. Although this would also decrease the outer cycle lengths again boosting up and worsening the score. But we still need to explore how to regularize the optimization based on the number and size of communities thus discouraging very small or/and very high number of communities.

12 Future Work

The next step is to come up with an algorithm to optimize our metric for community generation, rather than just use it for community validation as we have done. Right now the only way to generate quality communities based on our metric is a brute force approach of separating nodes in to every possible community, running the metric, and picking the partitions that generate the lowest score. Unfortunately, this is absolutely infeasible with the high run time complexity of both the brute force community partitioning and our metric. Ultimately we would like to use a meta-heuristic algorithm in order to create communities based on our metric as opposed to just leaving it as a verifier.

The other way we would like to extend our project is by looking at using MapReduce in order to increase scalability. We have already observed that

graphs with thousands of nodes and hundreds of thousands of edges can run upwards of an hour. This is a very low and very breachable ceiling. While there is some easy potential for running our metric in parallel (by giving each computer a different subset of the edges), this still does not help with memory scalability, and does not begin to offer the same benefits that MapReduce does. Some of the techniques we could use such as all pairs shortest path have already been implemented in MapReduce. This would defnitely result in increase in scalability of the validation of the partitioning for larger graphs. Additionally, several approximate meta-heuristic optimization algorithms in MapReduce could also be explored for further extending this metric to obtain communities for large scale networks, besides just verifying their efficacy.

13 Conclusion

We started with exploring solutions to the problem of community detection in directed graphs as outlined in [3] and [1] and found that in specific, but not uncommon cases, they failed to find appropriate communities. Instead, we proposed that cycles are a better property to exploit for community detection because they represent the complete flow of information within the communities, and potential loss or dilution of information going outside. We then proposed a metric that is built on this intuition. We validated our hypothesis on multiple types and sizes of graphs with different kinds of embedded communities. While both the modularity based measures degrade from suboptimal partitions to ground truth, our metric shows a marked improvement. We showed how to potentially extend it to wighted directed and signed directed networks and opened up the avenues for further research in these directions. Our metric, as it stands, is only valid for verifying the community structures but we believe that further work on optimizing directly on our metric would lead to good community partitioning.

14 References

[1] Leicht, E. A., and M. E. J. Newman. "Community Structure in Directed Networks." Physical Review Letters 100.11 (2008).

[2] Traag, V. A., and Jeroen Bruggeman. "Community Detection in Networks with Positive and Negative Links." Physical Review E 80.3 (2009).

[3] M. E. J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69, 066133 (2004).

[4] Newman, M. E. J., Analysis of weighted networks Phys. Rev. E 70, 056131 (2004).

[5] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. Journal of the ACM, 51(3):497515, 2004.

[6] M. E. J. Newman and M. Girvan, Finding and evaluating community structure in networks. Preprint cond-mat/0308217 (2003)