Crawling the Community Structure of Multiplex Networks (Supplemental Materials)

Ricky Laishram
Syracuse University
Syracuse NY, USA
rlaishra@syr.edu

Jeremy D. Wendt
Sandia National Laboratories
Albuquerque, NM, USA
jdwendt@sandia.gov

Sucheta Soundarajan
Syracuse University
Syracuse NY, USA
susounda@syr.edu

1 Running Time of `MCS`

For a given sampling problem, the layer costs, number of layers, and graph properties are fixed. The user specifies the total budget allocated for the sampling. In this section, we discuss how the running time of `MCS` scales with the properties of M(the underlying multiplex network) and the budget given B.

For simplicity, let us assume that `RNDSample` and `MABSample` are performed η times; and the budget B is split up evenly each time. Then the budget allocated for each iteration is

$$B' = \frac{B}{\eta}. \tag{1}$$

Let $c_0, c_1, \ldots, c_l$ be the cost of a neighbor query in each of the layers, and let

$$\zeta = \sum_{y=[0,l]} \frac{\Lambda_{y,0}}{c_y} \tag{2}$$

$$\text{and, } \overline{\zeta} = \sum_{y\in[0,l]} \frac{\Lambda_{y,0}}{c_y^2} \tag{3}$$

where $\Lambda_{y,0}$ is the edge overlap of L_y w.r.t. L_0.

During the budget allocation, the budget for a layer L_x is proportional to the edge overlap, $\Lambda_{x,0}$, and inverse of layer cost, $\frac{1}{c_x}$.

$$B_x = \frac{\frac{\Lambda_{x,0}}{c_x}}{\sum_{y=[0,l]} \frac{\Lambda_{y,0}}{c_y}} B' = \frac{\Lambda_{x,0} B}{c_x \eta \zeta} \tag{4}$$

Suppose there is a layer L_z such that $\frac{\Lambda_{z,0}}{c_z^2}$ is the maximum. That is

$$z = \max_{y\in[0,l]} \frac{\Lambda_{y,0}}{c_y^2} \tag{5}$$

$$\text{and, } \lambda = \frac{\Lambda_{z,0}}{c_z^2}. \tag{6}$$

Then the number of nodes queried in L_x in one iteration is,

$$n_x = \frac{\Lambda_{x,0} B}{c_x^2 \eta \zeta} \leq \frac{B\lambda}{\eta\zeta} = n_z. \tag{7}$$

Let us assume that a neighborhood query takes constant time. During one iteration of `RNDSample`, a random walk is performed on $(l-1)$ layers. So, the number of nodes queried is

$$\sum_{x\in(0,l]} n_x \leq \sum_{x\in(0,l]} n_x = \frac{B\,(l-1)\,\lambda}{\eta\zeta}. \tag{8}$$

`RNDSample` is performed η times. So, total number of nodes queried in all layers is

$$N_r \leq \frac{B\,(l-1)\,\lambda}{\zeta}.$$

So, the running time due to `RNDSample` is

$$\mathcal{O}\left(\frac{B\,(l-1)\,\lambda}{\zeta}\right) \approx \mathcal{O}\left(\frac{lB\lambda}{\zeta}\right). \tag{9}$$

After the j-th iteration of `RNDSample`, the number of nodes in L_x^S is[1],

$$n_x^S \approx j n_x + n_x' \tag{10}$$

where n_x' is the number of nodes in L_x^S that have been observed but not queried.

If the number of queried nodes is much smaller than the total number of nodes the original graph, the number of observed but unqueried nodes is proportional to the number of queried nodes. That is, if d is the average degree, every queried node can bring in $(d-1)$ nodes. So,

$$n_x' \approx (d-1)\, j n_x \implies n_x^S \leq d j n_z. \tag{11}$$

During the initialization of `MABSample`, community detection is performed on all the layers. So, time it takes to perform community detection during that iteration is[2]

$$\mathcal{O}\left(\sum_{x\in[0,l]} n_x^S \log\left(n_x^S\right)\right) \approx \mathcal{O}\left(ldjn_z \log\left(djn_z\right)\right). \tag{12}$$

Then, the total time taken by this step in the η iterations is,

$$\mathcal{O}\left(\sum_{j=1}^{\eta} \left(ljdn_z \log\left(djn_z\right)\right)\right) \approx \mathcal{O}\left(\eta^2 ldn_z \log\left(d\eta n_z\right)\right). \tag{13}$$

[1] The value will be lower in the case of unreliable query response.

[2] Assuming the community detection algorithm is the Louvain method.

Substituting for n_z, the time taken by the community detection steps is

$$\begin{aligned}&\mathcal{O}\left(\frac{\eta^2 ldB\lambda}{\eta\zeta}\log\left(\frac{d\eta B\lambda}{\eta\zeta}\right)\right)\\ \approx&\mathcal{O}\left(\frac{ldB\lambda}{\zeta}\log\left(\frac{dB\lambda}{\zeta}\right)\right).\end{aligned}\tag{14}$$

During the node selection through `RBandit`, we need to compute various node centralities. Among all the type of roles that we use, the one that is most expensive to compute is the betweenness centrality. So, for simplicity let us assume that an arm involving the betweenness centrality is always selected. We use an approximate method of computing the betweenness centrality as described in (Chehreghani 2014). The run time contribution from this step is

$$\mathcal{O}\left(\eta lkn_z\right)\approx\mathcal{O}\left(\frac{lB\lambda}{\zeta}\right).\tag{15}$$

When we compute the community update distance, the two partitions are very close. So, the time it takes to compute it is (Porumbel, Hao, and Kuntz 2011)

$$\mathcal{O}\left(\frac{lB}{\zeta}\right).\tag{16}$$

The time it takes to compute the edge overlap is also

$$\mathcal{O}\left(\frac{lB}{\zeta}\right).\tag{17}$$

So, the running time of `MCS` is

$$\begin{aligned}&\mathcal{O}\left(\frac{lB(\lambda+1)}{\zeta}+\frac{lB}{\zeta}+\frac{ldB\lambda}{\zeta}\log\left(\frac{dB\lambda}{\zeta}\right)\right)\\ \approx&\mathcal{O}\left(\frac{ldB\lambda}{\zeta}\log\left(\frac{dB\lambda}{\zeta}\right)\right).\end{aligned}\tag{18}$$

2 Sensitivity Analysis of `MCS`

There are several of factors that affects the performance of `MCS`. The main ones are: (1) the edge overlap between the different layers (not just between cheaper layer and expensive layer), (2) the number of layers, and (3) the relative cost of a query in the cheaper layer compared to the expensive layer. Of these factors, the number of layers and query costs are known beforehand, but the edge overlap is not.

In general, if we have more information (edges) about L_0 collected with the same algorithm, the community structure of L_0^S will be more similar to that of L_0 (Maiya and Berger-Wolf 2010).

Assume that there are l layers and the cost of a query in a layer L_x is c_x. Because the network is undirected, we will observe duplicate edges when we query multiple nodes in the same layer. The amount of duplicates will depend on: (1) density of edges in the layer, and (2) the number of queries already made. During the sampling process, the number of queries already made is the only variable. So, for a layer L_x on which there have already been i queries, let $o_x\left(i\right)$ be the expected fraction of unobserved nodes in the next query response.

$$o_x\left(i\right)\geq o_x\left(i+1\right).\tag{19}$$

For simplicity, let us assume that the budget allocated to a layer is the same in every iteration. Then, for budget B, the expected number of edges found after using up the allocated budget is,

$$d_x\sum_{i=0}^{\eta B_x}o_x(i).\tag{20}$$

Unless L_x is the first layer that is being queried, there is some probability that edges found in L_x have already been observed in a query on another layer. This probability depends on the edge overlay between the other layers and L_x. Suppose the layers are queried in sequence. (That is layer L_0 first, L_1 next and so on.) The probability of finding an edge not observed in an earlier layer is,

$$\prod_{y=0}^{x-1}\left(1-\Lambda_{y,x}\right).\tag{21}$$

Then the expected number of unobserved edges on querying layer L_x is

$$d_x\left(\prod_{y=0}^{x-1}\left(1-\Lambda_{y,x}\right)\right)\left(\sum_{i=0}^{\eta B_x}o_x(i)\right).\tag{22}$$

Recall that we are interested in the edges in L_0, and not all the edges found in the other layers will exist in L_0. The edge overlap $\Lambda_{x,0}$ gives us the fraction of edges found in L_x that also exist in L_0. So, the expected number of previously unobserved edges found in L_x that also exist in L_0 is

$$d_x\Lambda_{x,0}\left(\prod_{y=0}^{x-1}\left(1-\Lambda_{y,x}\right)\right)\left(\sum_{i=0}^{\eta B_x}o_x(i)\right).\tag{23}$$

Then if all the layers are queried, the expected number of edges in L_0 found is

$$\mathbb{E}\left[\left|E_0^B\right|\right]=\sum_{x=0}^{l}\left(d_x\Lambda_{x,0}\left(\prod_{y=0}^{x-1}\left(1-\Lambda_{y,x}\right)\right)\left(\sum_{i=0}^{\eta B_x}o_x(i)\right)\right).\tag{24}$$

Now we can examine the effect of number of layers and query cost on the performance of `MCS`.

2.1 Effect of Number of Layers

For simplicity let us assume that all the cheaper layers have the same cost ratio, and η is fixed.

Let $\mathbb{E}\left[E_0^B\right]$ be the the expected number of edges in L_0^B when there are l layers. Suppose we introduce a new layers such that the total layers count is now l'. Then, assume $\mathbb{E}\left[\overline{E}_0^B\right]$ be the expected number of edges when there are l' layers. Then,

$$\sum_{y\in[0,l]}\frac{\Lambda_{x,0}}{c_x}\leq\sum_{y\in[0,l']}\frac{\Lambda_{x,0}}{c_x}\tag{25}$$

$$\zeta\leq\zeta'\tag{26}$$

$$B_x\geq B'_x\tag{27}$$

where B'_x is the budget allocated to layer L_x in the case where we have l' layers.

Then,

$$\begin{aligned}
&\mathbb{E}\left[|E_0^S|\right] - \mathbb{E}\left[|\overline{E}_0^S|\right] \\
&= \sum_{x=0}^{l}\left(d_x\Lambda_{x,0}\left(\prod_{y=0}^{x-1}(1-\Lambda_{y,x})\right)\left(\sum_{i=0}^{\eta B_x} o_x(i)\right)\right) \\
&\quad - \sum_{x=0}^{l'}\left(d_x\Lambda_{x,0}\left(\prod_{y=0}^{x-1}(1-\Lambda_{y,x})\right)\left(\sum_{i=0}^{\eta B'_x} o_x(i)\right)\right) \\
&= \sum_{x=0}^{l}\left(d_x\Lambda_{x,0}\left(\prod_{y=0}^{x-1}(1-\Lambda_{y,x})\right)\left(\sum_{i=\eta B'_x+1}^{\eta B_x} o_x(i)\right)\right) \\
&\quad - \sum_{x=l+1}^{l'}\left(d_x\Lambda_{x,0}\left(\prod_{y=0}^{x-1}(1-\Lambda_{y,x})\right)\left(\sum_{i=0}^{\eta B'_x} o_x(i)\right)\right).
\end{aligned} \tag{28}$$

Case 1: l is large For large values of x, it can be shown that

$$\prod_{y=0}^{x-1}(1-\Lambda_{y,x}) \approx 0. \tag{29}$$

Then, if l is large enough

$$\sum_{x=l+1}^{l'}\left(d_x\Lambda_{x,0}\left(\prod_{y=0}^{x-1}(1-\Lambda_{y,x})\right)\left(\sum_{i=0}^{\eta B'_x} o_x(i)\right)\right) \approx 0. \tag{30}$$

It can be shown that

$$\sum_{x=0}^{l}\left(d_x\Lambda_{x,0}\left(\prod_{y=0}^{x-1}(1-\Lambda_{y,x})\right)\left(\sum_{i=\eta B'_x+1}^{\eta B_x} o_x(i)\right)\right) \geq 0. \tag{31}$$

So,

$$\mathbb{E}\left[|E_0^S|\right] - \mathbb{E}\left[|\overline{E}_0^S|\right] \geq 0 \tag{32}$$

$$\implies \mathbb{E}\left[|E_0^S|\right] \geq \mathbb{E}\left[|\overline{E}_0^S|\right] \tag{33}$$

.

Case 2: l is small Since $o_x(i) \geq o_x(i+1)$, it can be shown that for large enough value of i

$$o_x(i) \approx 0. \tag{34}$$

And for small l, ζ is small, and consequently B_x is large. If this value is large enough,

$$\sum_{i=1+\eta B'_x}^{\eta B_x} o_x(i) \approx 0 \tag{35}$$

$$\text{and, } \sum_{i=0}^{\eta B'_x} o_x(i) \geq 0 \tag{36}$$

Then,

$$\sum_{x=0}^{l}\left(d_x\Lambda_{x,0}\left(\prod_{y=0}^{x-1}(1-\Lambda_{y,x})\right)\left(\sum_{i=\eta B'_x+1}^{B_x} o_x(i)\right)\right) \approx 0 \tag{37}$$

$$\sum_{x=l+1}^{l'}\left(d_x\Lambda_{x,0}\left(\prod_{y=0}^{x-1}(1-\Lambda_{y,x})\right)\left(\sum_{i=0}^{\eta B'_x} o_x(i)\right)\right) \geq 0. \tag{38}$$

Then,

$$\mathbb{E}\left[|E_0^S|\right] - \mathbb{E}\left[|\overline{E}_0^S|\right] \leq 0 \tag{39}$$

$$\implies \mathbb{E}\left[|E_0^S|\right] \leq \mathbb{E}\left[|\overline{E}_0^S|\right]. \tag{40}$$

So, as the number of layer increases the expected performance of MCS increases initially, but after a certain value it decreases with increasing number of layers.

2.2 Effect of Relative Layer Cost

For simplicity assume that all the cheaper layers have the same query cost. Assume relative layer costs r, r' such that $r > r'$. Then, the layer cost of $L_{x>0}$ are

$$c_x = rc_0 \tag{41}$$

$$\text{and, } c'_x = r'c_0. \tag{42}$$

And let,

$$\zeta = \sum_{x\in[0,l]} \frac{\Lambda_{x,0}}{c_x} \tag{43}$$

$$\text{and, } \zeta' = \sum_{x\in[0,l]} \frac{\Lambda_{x,0}}{c'_x}. \tag{44}$$

Then,

$$r\zeta = r\sum_{x\in[0,l]} \frac{\Lambda_{x,0}}{c_x} = \frac{r}{c_0} + \sum_{x\in(0,l]} \frac{\Lambda_{x,0}}{c_0} \tag{45}$$

$$\text{and, } r'\zeta' = r'\sum_{x\in[0,l]} \frac{\Lambda_{x,0}}{c'_x} = \frac{r'}{c_0} + \sum_{x\in(0,l]} \frac{\Lambda_{x,0}}{c_0}. \tag{46}$$

So,

$$r\zeta > r'\zeta \implies B_x < B'_x. \tag{47}$$

Suppose $\mathbb{E}\left[|E_0^S|\right]$ and $\mathbb{E}\left[|\overline{E}_0^S|\right]$ are the expected number of edges in L_0^S in the case of layer cost r an r' respectively.

Then,

$$\begin{aligned}
&\mathbb{E}\left[|E_0^S|\right] - \mathbb{E}\left[|\overline{E}_0^S|\right] \\
&= \sum_{x=0}^{l}\left(d_x\Lambda_{x,0}\left(\prod_{y=0}^{x-1}(1-\Lambda_{y,x})\right)\left(-\sum_{i=\eta B_x+1}^{\eta B'_x} o_x(i)\right)\right).
\end{aligned} \tag{48}$$

Since, $o_x(i) \geq 0$,

$$\mathbb{E}\left[|E_0^S|\right] \leq \mathbb{E}\left[|\overline{E}_0^S|\right]. \tag{49}$$

That is, the performance of MCS increases if the relative cost of the cheaper layers drops.

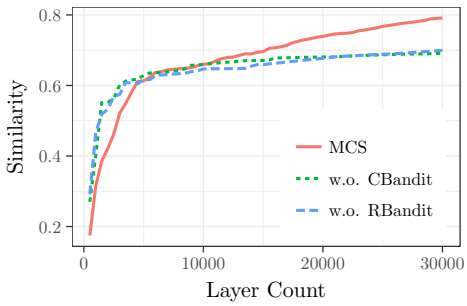

Figure 1: Performance comparison between `MCS` (red), `MCS` wihout `CBandit` (green), and `MCS` without `RBandit` (blue).

3 Effect of Role and Community Bandits

One can ask the question of how much the different bandits contribute to the performance of `MCS`. To answer this, we compare `MCS`, which has all the bandits, against ones without `CBandit` and `RBandit`. In the absence of `CBandit` all the unqueried nodes in the layer are in the candidate set, and in the absence of `RBandit` a random node is selected from the candidate set.

We cannot remove `LBandit`, because we need to know which layer to query on. So, in these experiments, we compare `MCS`, which has all the bandits, against ones without `CBandit` and `RBandit`. In the absence of `CBandit` all the unqueried nodes in the layer are in the candidate set, and in the absence of `RBandit` a random node is selected from the candidate set.

Figure 1 shows the comparison between `MCS` with all three bandits (shown in red), without `CBandit` (green) and without `RBandit` (blue) on the DBLP dataset. It can be observed that the methods with only two bandits perform well initially. This might be due to the fact that with fewer bandits, there are fewer combinations of arms to learn. However, `MCS` with all three bandits soon catches up, and then outperforms the others.

References

Chehreghani, M. H. 2014. An efficient algorithm for approximate betweenness centrality computation. *The Computer Journal* 57(9):1371–1382.

Maiya, A. S., and Berger-Wolf, T. Y. 2010. Sampling community structure. In *Proceedings of the 19th international conference on World wide web*, 701–710. ACM.

Porumbel, D. C.; Hao, J. K.; and Kuntz, P. 2011. An efficient algorithm for computing the distance between close partitions. *Discrete Applied Mathematics* 159(1):53–59.