Finding Network Motifs Using MCMC Sampling*

Tanay Kumar Saha and Mohammad Al Hasan

Department of Computer and Info. Science
Indiana University-Purdue University
Indianapolis, Indiana, USA
{tksaha,alhasan}@cs.iupui.edu

Abstract. Scientists have shown that network motifs are key building block of various biological networks. Most of the existing exact methods for finding network motifs are inefficient simply due to the inherent complexity of this task. In recent years, researchers are considering approximate methods that save computation by sacrificing exact counting of the frequency of potential motifs. However, these methods are also slow when one considers the motifs of larger size. In this work, we propose two methods for approximate motif finding, namely SRW-rw, and MHRW based on Markov Chain Monte Carlo (MCMC) sampling. Both the methods are significantly faster than the best of the existing methods, with comparable or better accuracy. Further, as the motif size grows the complexity of the proposed methods grows linearly.

1 Introduction

Studying the local topology is an important step for modeling the interaction among the entities in a network. In a seminal work around a decade ago, Shen-orr et al. [14] hypothesized that network motifs play an important role in carrying out the key functionalities that are performed by the entities in a biological network. Since then, researchers have also discovered that network motifs are building block for complex networks from many diverse disciplines including biochemistry, neurobiology, ecology, engineering [11], proteomics [1], social sciences [6] and communication [5].

Finding network motifs is computationally demanding. To identify whether a given subgraph topology is a motif, we need to count the topology's frequency in the input network as well as in many randomized networks. Counting a topology's frequency in a single network is a challenging task as it requires solving subgraph isomorphism, a known $\mathcal{NP}$-complete problem. As the size of the motif grows, the number of candidate motifs increases exponentially, and the task becomes more challenging. To cope with the enormous computation cost of exhaustive counting of the frequency of candidate motifs, researchers consider various sampling based methods that obtain an approximation of relative frequency measure (which we call concentration) over all the candidates of a given size. Most notable among these methods are MFinder [8], MODA [12], and RAND-ESU [16]. Besides these approximate methods, exact motif counting methods are also available, such as, GTrieScanner [13], ESU [16], Grochow-Kellis algorithm [4],

* This research is supported by Mohammad Hasan's NSF CAREER Award (IIS-1149851).

G. Mangioni et al. (eds.), *Complex Networks VI,*
Studies in Computational Intelligence 597, DOI: 10.1007/978-3-319-16112-9_2

Kavosh [7], and NetMODE [9]; However, their application is limited to small networks only. In this work, our focus is on finding concentration of prospective motifs using a novel sampling based method.

The quality of a sampling based method depends on three critical performance metrics: accuracy, convergence, and execution time. Existing sampling based methods are poor in one or more of the above performance metrics. For instance, MFinder is costly and it scales poorly with the size of the desired motifs. Authors in [16] have shown that the cost of subgraph sampling of MFinder increases exponentially with the size (number of vertex) of the subgraph. It is also poor in terms of accuracy and convergence. A similar method, RAND-ESU [16] is significantly faster than MFinder and yet its scalability is also not that satisfactory. Besides, its sampling accuracy and convergence behavior are also poor.

Another important fact about the existing sampling based methods is that they require random access to any of the vertices or the edges in the networks. This becomes a severe limitation for networks for which such unrestricted access is not available. For an instance, consider the Web network or a hidden network, a user may not have access to any arbitrary vertex/edge in the input network for security reason; rather, the desired node can only be accessed from another node which is one-hop away from it; such scenarios are common in real-life and are considered in the task of snowball sampling [3]. None of the existing methods can be used for finding motifs in a graph that only allows restricted access, such as crawling.

In this work, we propose two random walk based methods, namely MHRW (Metropolis-Hastings random walk) and SRW-RW (Simple Random Walk with Reweighting) for approximating the concentration of arbitrary-sized pattern graphs in a large network. The underlying mechanism of both the methods is a Monte Carlo Markov Chain (MCMC) sampling over the candidate motif space, which is guaranteed to compute an unbiased estimate of concentration of all the candidate motifs of a given size simultaneously. Since, our methods are based on random walk over the edges of the input graph, they only require a restricted access over the network such that at any given time of the walk the one-hop neighboring nodes of currently visiting candidate are accessible. Besides, the methods are scalable and are significantly faster than the existing methods. They also have better convergence property and small memory footprint. While preparing for the final manuscript of this work, we have found another work [15], where the authors propose methodologies that are similar to our work.

2 Background

2.1 Graph, Subgraph, Induced Subgraph

Let $G(V,E)$ is a *graph*, where V is the set of vertex and E is the set of edges. Each edge $e \in E$ is denoted by a pair of vertices (v_i, v_j) where, $v_i, v_j \in V$. A graph without a self-loop or multi edge is a simple graph. In this work, we consider simple, connected, and undirected graphs.

A graph $G' = (V', E')$ is a *s*ubgraph of G (denoted as $G' \subseteq G$) if $V' \subseteq V$ and $E' \subseteq E$. A graph $G' = (V', E')$ is a *v*ertex-induced subgraph of G if G' is a subgraph of G, and for any pair of vertices $v_a, v_b \in V'$, $(v_a, v_b) \in E'$ if and only if $(v_a, v_b) \in E$. In other words,

a *vertex-induced* subgraph of G is a graph G' consisting of a subset of G's vertices together with all the edges of G whose both endpoints are in this subset. In this paper, we have used the phrase *induced subgraph* for abbreviating the phrase vertex-induced subgraph. If G' is an induced subgraph of G and $|V'| = p$, we call G' a *p-subgraph* of G. An *embedding* of a graph G' in another graph G is a subgraph S of G such that S and G' are isomorphic;

For a given vertex count, the number of distinct graph topologies is fixed. We use the symbol Λ_p to denote the set of all such topologies. To denote one specific topology in Λ_p we use the symbol $\omega_{p,q}$, where q is the order of that topology (considering an arbitrary but fixed ordering) among all the size p topologies. The set of induced embeddings of all graphs in Λ_p in graph G is the collection of p-subgraphs of G. Figure 1 shows all the elements of the sets Λ_3, Λ_4 and Λ_5. Using the order of the topologies in this figure, $\omega_{3,1}$ is the 3-node line graph.

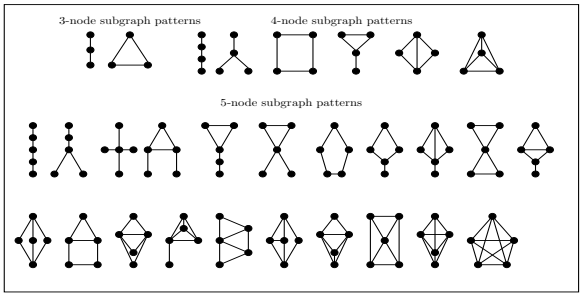


Fig. 1. All 3, 4 and 5 node topologies

2.2 Subgraph Concentration

The frequency of a particular p-subgraph topology g in an input graph G is the number of times it appears in G. We denote it by $f_G(g)$. The concentration of g in G is $C_G(g)$, which is defined as the normalized frequency over the cumulative frequency of all the subgraph topologies in the set Λ_p. Mathematically,

$$C_G(g) = \frac{f_G(g)}{\sum_{h \in \Lambda_p} f_G(h)} \tag{1}$$

2.3 Motif

A Motif is a subgraph topology which occurs in an input network at a significantly higher frequency than it occurs in a set of random networks with identical characteristics. For this purpose, the random networks are generated from the input network by imposing the constraint that the vertices of a random network has the identical degree distribution as that of the input network. There are several methods for generating random networks with identical degree distribution, but the most popular is the switching algorithm [10], which we use in this work. The significance of frequency deviation between the input network and the set of random networks is typically measured using z-score and p-value. If $\overline{f_{G_r}(g)}$ is the mean frequency of g in a set of randomized

graphs G_r (constructed from G), and $\sigma_{G_r}(g)$ is the corresponding standard deviation, then z-score of g for the input network G is defined as:

$$z_G(g) = \frac{f_G(g) - \overline{f_{G_r}(g)}}{\sigma_{G_r}(g)} \tag{2}$$

If the z-score of g is greater than some pre-specified threshold then we call g a motif. Since, setting this threshold requires domain expertise, all the existing motif finding methods consider it as a run-time parameter; we also follow the same in our work. For sampling based solution, we use concentration of subgraph instead of their frequency. Hence, z-score is defined as below:

$$\widehat{z}_G(g) = \frac{\widehat{C}_G(g) - \overline{\widehat{C}_{G_r}(g)}}{\widehat{\sigma}_{G_r}(g)} \tag{3}$$

In equation 3, we use $\widehat{C}_G$, and $\widehat{\sigma}_G$ to denote that they are statistics obtain from random sample of size-p embeddings.

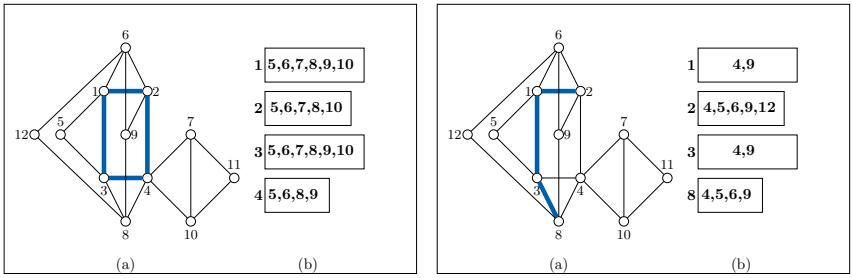
2.4 Markov Chains, and Metropolis-Hastings (MH) Method

A Markov chain is the sequence of Markov process over the state space S. The state-transition event is guided by a matrix, T, called *transition probability matrix*. The chain is said to reach a stationary distribution π, when the probability of being in any particular state is independent of the initial condition, it is reversible if it satisfies the *reversibility condition* $\pi(i)T(i,j) = \pi(j)T(j,i), \forall i,j \in S$ and it is *ergodic* if it has a stationary distribution. The main goal of the MH is to draw samples from some distribution $\pi(x)$, called the *target distribution*, where, $\pi(x) = f(x)/K$; here K is a normalizing constant which may not be known and difficult to compute. It can be used together with a random walk to perform MCMC sampling. For this, the MH algorithm calculates the *acceptance probability* using the following equation:

$$\alpha(x,y) = \min\left(\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\right) \tag{4}$$

3 Methods

Given a graph G (which we refer as input graph) and an integer p, a sampling based method samples a small set of p-subgraphs of G. From this set, it approximates the concentration of each topology in Λ_p as shown in section 2.3. To measure the exact concentration, one must perform unbiased sampling, where each of the p-subgraphs has an uniform probability to be sampled. This is not an easy task, as the sample space is very large. Besides, a direct sampling method is not applicable because for that we need to enumerate all the p-subgraphs (to obtain the size of the sample space), which we want to avoid. So, an indirect sampling strategy must be followed. Both MFinder [8] and RAND-ESU [16] adopt indirect sampling; however, they differ in the sampling



(a) Left: A graph G with the current state of random walk; Right: Neighborhood information of the current state (1,2,3,4)

(b) Left: The state of random walk on G (Figure 2(a)) after one transition; Right: Updated Neighborhood information

Fig. 2. Neighbor generation mechanism

methodologies. MFinder's sampling is biased which requires post-adjustment of concentration for correcting the bias; on the other hand, RAND-ESU guaranty a uniform sampling which requires no correction. For large p, both MFinder and RAND-ESU are costly.

In this paper, we propose MHRW, and SRW-RW for sampling p-subgraphs of a graph using Markov chain Monte Carlo (MCMC) sampling. As a Metropolis-Hasting based method (discussed in sec: 2.4), they perform a random walk over the state space so that the stationary distribution of the random walk converges to a desired target distribution. For our task, the state space are the set of p-subgraphs. Since, we want to approximate the concentration of each of the topologies in Λ_p, our target distribution is *uniform*, i.e., we want to sample each of the p-subgraphs with an identical probability. If $\mathcal{P}$ is the set of the p-subgraphs in the input graph G, and π is the target distribution, we want $\pi(g) = 1/|\mathcal{P}|, \forall g \in \mathcal{P}$.

```
MHRW(G, p, N)
    G: Input Graph
    p: Subgraph Size
    N: Size of the sample set, |S|

 1  g = Starting State
 2  M = φ
 3  i = 0
 4  d_g = Neighbor count of g
 5  while i < N
 6      h = Any neighbor of g chosen
            uniformly at random from (1, |d_g|)
 7      d_h = Neighbor count of h
 8      accp_val = d_g / d_h
 9      accp_probablility = min(1, accp_val)
10      if uniform(0,1) ≤ accp_probability
11          g = h
12          d_g = d_h
13      i = i + 1
14      Generate the Canonical code of g
15      Insert the code into the set M
        and update the count
16  Normalize the frequency using equation 5,
    ∀_i ω_{p,i} ∈ M
17  return M.
```

Fig. 3. MHRW Pseudocode

For the random walk of both MHRW and SRW-RW, a neighbor of a p-subgraphs (say, g) is obtained by simply replacing one of its existing vertices of g with another vertex which is not part of g and find the subgraph induced by the new vertex-set.

While replacement, the methods ensure that the new set of vertices induce a connected p-subgraph. At every iteration, all possible neighbors are populated using the above strategy. For a state, the number of neighboring states are called its *degree*.

Example: Suppose our sampling method (MHRW or SRW-RW) is sampling a 4-subgraph from the graph G shown in Figure 2(a)(Left). Let, the 4-subgraph $\langle 1,2,3,4\rangle$ (shown in bold lines) is the existing state of this random walk. One of it's neighbor state is $\langle 1,2,3,8\rangle$, which can be obtained by replacing the vertex 4 by the vertex 8. In Figure 2(a)(Right) we show the information of all its neighbors. Box labeled by x contains all the vertices that can be used as a replacement of vertex x to get a neighbor. If the random walk transition chooses to go to the neighbor state $\langle 1,2,3,8\rangle$, it can do so simply by adding the vertex 8 (a vertex in the box labeled by 4) and deleting the vertex 4. The updated state of the random walk along with the updated neighbor-list is shown in Figure 2(b). The degree of a state is the number of neighbors, which is simply the sum of the entries in each of the boxes; thus the degree of state $\langle 1,2,3,4\rangle$ is 21, and the degree of the state $\langle 1,2,3,8\rangle$ is 13. ■

To apply MH algorithm, we also need to decide on a proposal distribution, q. For MHRW random walk, we choose the proposal distribution to be uniform, i.e., in the proposal step MHRW chooses one of g's neighbors uniformly. If $h \in \mathcal{P}$ and h is a neighbor of g based on our neighborhood definition, using proposal distribution, the probability of choosing h from g, $q(g,h) = 1/d_g$, where d_g is the degree of the state g. Also note, if $m \in \mathcal{P}$, but m is not a neighbor of g, $q(g,m) = 0$, i.e., transitions are allowed among neighboring states only.

Using the proposal (q) and target (π) distributions, MHRW method is simply an implementation of the algorithm that we discussed in Section 2.4. A pseudo-code of MHRW is given in Figure 3. At the beginning of the sampling for each topology in Λ_p, we assign a counter which is initialized to 0. As the sampling progress, for each state we identify the specific topology that the state represents, and increment its counter by 1. Thus, if $\mathcal{S}$ is the sample set, the concentration equation defined in 1 for g where $g \in \Lambda_p$ becomes:

$$\widehat{C}(g) = \frac{1}{|\mathcal{S}|} \sum_{x \in \mathcal{S}} 1_{(x==g)} \tag{5}$$

At any iteration from the current stage g, the method chooses one of its neighbors, (say, h) using the proposal distribution (uniform), and either accept or reject the proposed move using Equation 4 i.e. MHRW adjusts the transition probability by accepting or rejecting the proposed transition so that the target distribution is guaranteed to be uniform.

On the other hand, an iteration of SRW-RW (simple random walk with re-weighting) simply chooses one of the neighbors uniformly and make this transition. Thus the difference between MHRW and SRW-RW is that the latter chooses the proposed transition with 100% probability. This does not guarantee uniform sampling of the states (p-subgraphs); rather the states are sampled in proportional to their degree values. In other words, the target distribution of simple random walk is directly proportional to the degree value of the p-subgraphs. So, the concentration of the topologies in Λ_p is also biased in proportional amount. To obtain an unbiased estimate of concentration, the estimated concentration should be re-weighted, which gives the name simple

random walk with re-weighting or in short SRW-RW. After re-weighting the concentration equation (Equation 1) of SRW-RW takes the following form:

$$\widehat{C}(g) = \frac{1}{W} \sum_{x \in \mathcal{S}} (1/d_x)_{(x==g)} \tag{6}$$

where, W is the sum of the total weights, i.e., $W = \sum_{x \in \mathcal{S}} (1/d_x)$. Such an idea of re-weighting has been used in [2] for approximating degree distribution of a large network by sampling.

Pseudo-code of SRW-RW is similar to the pseudo-code of Figure 3, the only difference is that, there is no acceptance rejection step and in Line 12, instead of incrementing the frequency count by 1, we increment the concentration by $1/d_g$. Finally, we normalize in Line 13 using equation 6 instead of equation 5.

Claim: For a given p and an input graph G, both MHRW and SRW-RW returns an unbiased estimate of the concentration of a topology in Λ_p.

Proof: Assume $g \in \Lambda_p$ is an arbitrary topology and $\mathcal{S}$ is a set of induced subgraph sampled from G. The expectation of g's concentration in G is $E\left[\widehat{C}(g)\right] = E\left[\frac{1}{|\mathcal{S}|}\sum_{x \in \mathcal{S}} 1_{(x \cong g)}\right] = E\left[P_u(x \cong g)\right]$. Here, $P_u(x \cong g)$ is the probability that a graph x in the sample set $\mathcal{S}$ is isomorphic to the topology g when it is sampled under uniform distribution. But, this value is the exact concentration value of g. So, $E\left[\widehat{C}(g)\right] = E\left[C_g\right] = C_G$. So, MHRW returns an unbiased estimate of the concentration of a topology in Λ_p.

By construction, the stationary distribution π for SRW-RW's random walk is proportional to the degree of a p-subgraph. Thus, for an arbitrary p-subgraph, w, its stationary probability $\pi(w) = d_w/K$ where K is a normalizing constant. For a topology $g \in \Lambda_p$, before re-weighting the expected value of its concentration is equal to $\sum_{w \in \mathcal{P}} \pi(w) \cdot 1_{(w \cong g)} = \sum_{w \in \mathcal{P}} \frac{d_w}{K} \cdot 1_{(w \cong g)}$. However if each sample w of type g contributes only $1/d_w$ instead of 1 in the counter of g, the expected value of concentration becomes $\sum_{w \in \mathcal{P}} \frac{d_w}{K} \cdot (\frac{1}{d_w})_{(w \cong g)} = \frac{1}{K}\sum_{w \in \mathcal{P}} 1_{(w \cong g)} = \frac{1}{K}C(g)$, which is the unbiased concentration scaled by a multiplicative constant. Since the concentration of all the topologies in Λ_p sums to 1, the expected value of the concentration returned by equation 6 after normalization is an unbiased estimate of the true concentration. ■

3.1 Implementation issues

Starting State. When we start the random walk on G, both MHRW, and SRW-RW starts from an arbitrary p-subgraph. To find it, the methods randomly choose an edge (of G) and include other adjacent edges to form an induced subgraph of desired size. As the input graph is connected, this process returns a p-subgraph of G.

Canonical Label of a Graph. We use *min-dfs-code* [17] for canonical labeling of the graph to unify different isomorphic forms of the same graph.

4 Results and Discussion

We implement MHRW and SRW-RW in C++ language and perform a set of experiments for evaluating their performance. We run all the experiments in a computer with 2.60 GHz processor and 4 GB RAM running Linux operating system. For experiments, we use graphs of different sizes from different domains. Table 1 lists the graphs along with the vertex count, the edge count and the average degree. Since the existing implementation of our methods only consider undirected graphs, all the input graphs are made undirected if necessary. The graphs are available from the following two web sites[1].

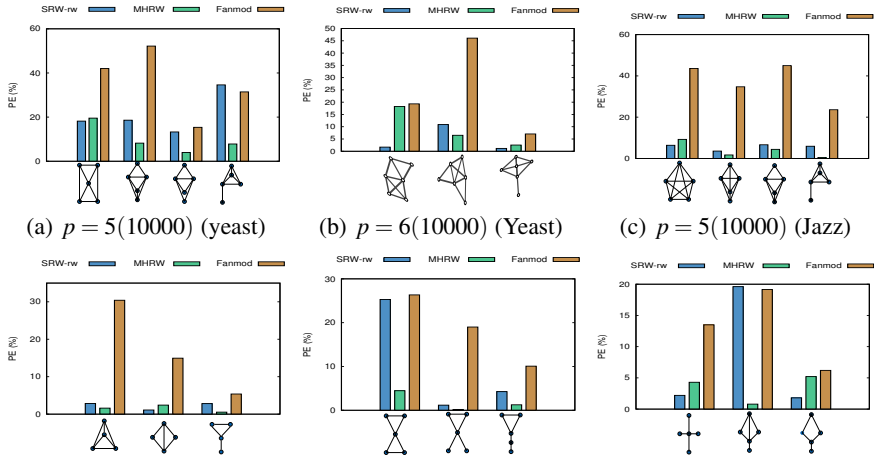
Table 1. Dataset Statistics

Graph	Vertex	Edge	Average Degree
Yeast	2,224	6,609	5.94
Jazz	198	2,742	27.49
ca-GrQc	4,158	13,422	6.43
ca-HepTh	8,638	24,806	5.74
ca-AstroPh	17,903	196,972	22.0

Experimental results in the earlier works show that RAND-ESU is the best among these three methods. In [16], Wernicke have shown that RAND-ESU is significantly faster than MFinder with a better accuracy. Another recent work [12] shows that RAND-ESU is the fastest among a set of methods including MODA. In this paper, we compare the performance of our methods with RAND-ESU to show that our methods are better than RAND-ESU in different performance metrics. We also considered MODA [12] for a comparison, but we found that its available implementation is unstable; the same fact was also reported by the authors of [9]. Note that we do not compare our methods with existing exact algorithm as they do not scale with the size of motif and also with the size of the input graph. For comparison with RAND-ESU, we use the implementation by authors that is available in the FANMOD library. Note that, in this implementation, the algorithm supports subgraph size up to 8. Besides a user need to set some probability values, which we set using the recommendation in FANMOD's documentation. In the result section, we will refer RAND-ESU as FANMOD following the convention in the earlier works.

We use three performance metrics: runtime, error, and convergence to compare our method with others. To compute the error value for a topology g, we first find the exact concentration of g using an exact method, then we find the approximate concentration using the sampling based method; the absolute difference between the above two concentration normalized by the actual concentration is the error for the topology g. However, since the sampling method is a randomized process, instead of using the approximate concentration of a single run, we take the average of the approximate concentration of 10 different runs. We represent the error as percentage and use the symbol $PE(g)$ (percentage error of g) for this metric.

[1] `http://snap.stanford.edu/data/index.html` and `http://www-personal.umich.edu/~mejn/netdata`



(a) $p = 5(10000)$ (yeast) (b) $p = 6(10000)$ (Yeast) (c) $p = 5(10000)$ (Jazz)

(d) $p = 4(40000)$ (Ca-AstroPh) (e) $p = 5(10000)$ (Ca-Hepth) (f) $p = 5(10000)$ (Ca-GrQC)

Fig. 4. Comparison of Percentage Error value for various methods. The dataset name, motif size, and the number of samples (in parenthesis) are given in figure sub-title.

4.1 Error Comparison

We compare the error percentage (PE) of various topologies using SRW-RW, MHRW, and FANMOD algorithms on all the datasets for different size values (p). Instead of showing the PE for all the topologies, we only show it for the topologies that are likely to be motifs, i.e., for these topologies, the $\widehat{z}_G(g)$ value in Equation 3 is the highest among all the topologies. For this experiment, we fixed the number of samples to 10000 for all of the experiments except for the experiment of Ca-AstroPh dataset, where we use 40000 samples.

For all the datasets, we see that our methods are significantly better than the FANMOD method based on the PE metric. Specifically, the performance gap between our method and FANMOD is very high for the Ca-AstroPh dataset, which is the largest among all our datasets. The performance of SRW-RW and MHRW are comparable. However, we observe that for topologies for which the concentration is high, MHRW's approximation is better than SRW-RW. On the other hand for graphs for which the concentration is small (see the dense topologies in Figure 4(b)), SRW-RW's approximation is better than MHRW. There are a few occasions where the PE of SRW-RW are as bad as FANMOD; nevertheless, the plots clearly demonstrate the superiority of Markov Chain based techniques over FANMOD in terms of percentage error.

4.2 Runtime Comparison

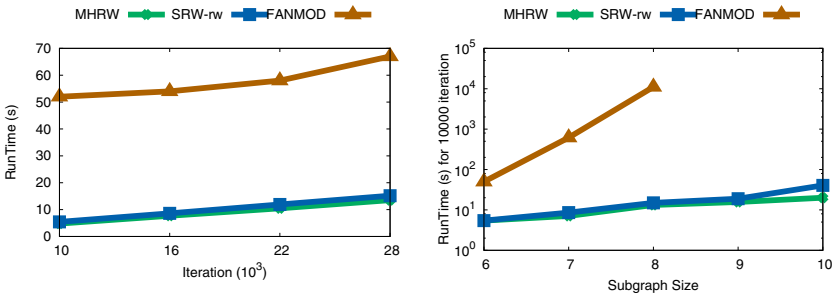
The runtime performance comparison of our methods with FANMOD is shown in Table 2. Here, we have fixed the sample count to 10000 for all the methods. To highlight the poor scalability of FANMOD with the size of the motif, we show some of the numbers in bold font. If we carefully observe the table we can see that as the size increases

by unity the runtime of FANMOD increases more than 10 times. For the Ca-AstroPh dataset which is the densest, for generating 10000 samples, FANMOD takes 180*s*, on the other hand both of our methods take about 5 seconds only. For this metric also, the performance gap between our methods and FANMOD increases as the dataset or the motif size increases.

We also show the runtime performance of the algorithms with the increasing number of samples in Figure 5(a) for yeast dataset and for subgraph size 5. The time increases mostly linearly for all the datasets; however, both of our methods have much smaller runtime than FANMOD. We also compare the runtime performance of the algorithms for motif sizes from 6 to 10. The result is shown in Figure 5(b) (note that *y*-axis is in logarithm scale). It is clear from the plot that our methods scale well with the increasing subgraph size. But, for FANMOD the runtime grows exponentially with the subgraph size; for example, to sample 10000 graphs from the yeast dataset, for subgraph size 7 and 8, it takes 616 seconds and 3 hours respectively. On the other hand, for size 8 our methods sample identical number of graphs in only 50 seconds. Also note that, FANMOD runs only for subgraph size up to 8.

Table 2. Runtime comparison of our methods with FANMOD

Dataset	Motif Size	MHRW (s)	SRW-rw (s)	FANMOD (s)
Yeast	5	2.73	3.13	2.73
	6	4.78	5.43	**50**
Jazz	5	5.08	5.71	3.45
	6	9.68	10.92	**52**
Ca-GrQC	3	0.79	1.06	0.026
	4	2.11	2.79	0.275
	5	7.03	10.53	2.79
	6	25.36	32.30	**34**
Ca-Hepth	3	0.60	0.75	0.43
	4	1.43	1.72	0.413
	5	3.03	3.30	5.37
	6	4.98	5.13	**70.41**
Ca-Astroph	3	3.20	4.48	3.35
	4	7.90	9.80	**180.38**



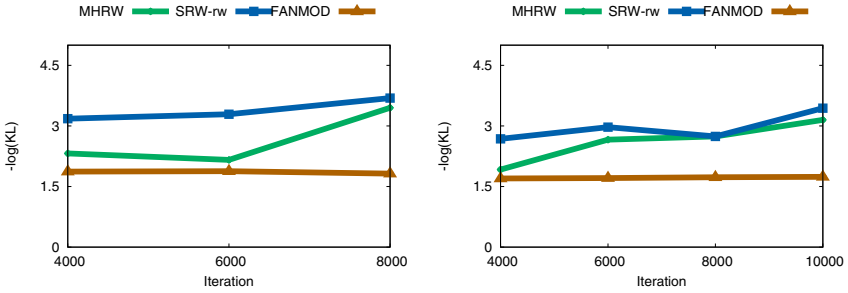
(a) Sample size vs Runtime (b) Subgraph Size vs Runtime

Fig. 5. Runtime performance for different sample sizes and for different subgraph sizes

4.3 Convergence Comparison

In this experiment, we study the convergence using the negative log (KL) metric by varying the number of samples. Figure 6(a) and 6(b) show that as we increase the

number of samples both the Markov chain based techniques approximate the concentration distribution more accurately (increasing value of $-\log(KL)$), on the other hand, for FANMOD the curve is almost flat, i.e. with an increasing number of samples FANMOD does not converge to the true concentration.



(a) Iteration vs KL (Size 5) (ca-Hepth) (b) Iteration vs KL (Size 6) (ca-Hepth)

Fig. 6. Comparison of convergence trend of our methods with FANMOD using KL Divergence

5 Conclusion

In this paper, we propose two methods MHRW, and SRW-RW for approximating the concentration of p-subgraphs in a host network for any given value of p. Our experimental results demonstrates that both of our proposed methods are significantly faster than the best of the existing methods. Moreover, our methods do not require full access over the networks. This makes our method useful for very large network (such as, Web) which can only be crawled.

References

1. Albert, I., Albert, R.: Conserved network motifs allow protein–protein interaction prediction. Bioinformatics 20(18), 3346–3352 (2004)
2. Gjoka, M., Kurant, M., Butts, C.T., Markopoulou, A.: Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. In: Proc. of IEEE INFOCOM, pp. 1–9 (2010)
3. Goodman, L.A.: Snowball sampling. Ann. Math. Statist. 32, 148–170 (1961)
4. Grochow, J.A., Kellis, M.: Network motif discovery using subgraph enumeration and symmetry-breaking. In: Speed, T., Huang, H. (eds.) RECOMB 2007. LNCS (LNBI), vol. 4453, pp. 92–106. Springer, Heidelberg (2007)
5. Itzkovitz, S., Alon, U.: Subgraphs and network motifs in geometric networks. Physical Review E, Statistical, Nonlinear, and Soft Matter Physics
6. Juszczyszyn, K., Kazienko, P., Musiał, K.: Local topology of social network based on motif analysis. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part II. LNCS (LNAI), vol. 5178, pp. 97–105. Springer, Heidelberg (2008)
7. Kashani, Z., Ahrabian, H., Elahi, E., Nowzari-Dalini, A., Ansari, E., Asadi, S., Mohammadi, S., Schreiber, F., Masoudi-Nejad, A.: Kavosh: a new algorithm for finding network motifs. BMC Bioinformatics 10(1), 318 (2009)

8. Kashtan, N., Itzkovitz, S., Milo, R., Alon, U.: Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. J. Bioinformatics 20(11), 1746–1758 (2004)
9. Li, X., Stones, D.S., Wang, H., Deng, H., Liu, X., Wang, G.: Netmode: Network motif detection without nauty. PLoS One 7(12) (December 2012)
10. Milo, R., Kashtan, N., Itzkovitz, S., Newman, M.E.J., Alon, U.: On the uniform generation of random graphs with prescribed degree sequences (May 2004)
11. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. Science 298, 824–827 (2002)
12. Omidi, S., Schreiber, F., Masoudi-Nejad, A.: MODA: an efficient algorithm for network motif discovery in biological networks. Genes and Genetic Systems 84(5), 385–395 (2009)
13. Ribeiro, P., Silva, F.: G-tries: an efficient data structure for discovering network motifs. In: Proc. ACM Symp. on Applied Computing, pp. 1559–1566 (2010)
14. Shen-Orr, S.S., Milo, R., Mangan, S., Alon, U.: Network motifs in the transcriptional regulation network of escherichia coli. Nature Genetics 31, 1061–4036 (2002)
15. Wang, P., Lui, J., Ribeiro, B., Towsley, D., Zhao, J., Guan, X.: Efficiently estimating motif statistics of large networks. ACM Trans. Knowl. Discov. Data 9(2) (2014)
16. Wernicke, S.: Efficient detection of network motifs. IEEE/ACM Transactions on Computational Biology and Bioinformatics 3(4), 347–359 (2006)
17. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: Proc. of 2nd International Conference on Data Mining, pp. 721–724. IEEE Computer Society (2002)