2016 IEEE 16th International Conference on Data Mining

Waddling Random Walk: Fast and Accurate Mining of Motif Statistics in Large Graphs

Guyue Han and Harish Sethu
Department of ECE, Drexel University
Email: {guyue.han, sethu}@drexel.edu

***Abstract*—Algorithms for mining very large graphs, such as those representing online social networks, to discover the relative frequency of small subgraphs within them are of high interest to sociologists, computer scientists and marketeers alike. However, the computation of these network motif statistics via naive enumeration is infeasible for either its prohibitive computational costs or access restrictions on the full graph data. Methods to estimate the motif statistics based on random walks by sampling only a small fraction of the subgraphs in the large graph address both of these challenges. In this paper, we present a new algorithm, called the *Waddling Random Walk (WRW)*, which estimates the concentration of motifs of any size. It derives its name from the fact that it sways a little to the left and to the right, thus also sampling nodes not directly on the path of the random walk. The WRW algorithm achieves its computational efficiency by not trying to enumerate subgraphs around the random walk but instead using a randomized protocol to sample subgraphs in the neighborhood of the nodes visited by the walk. In addition, WRW achieves *significantly* higher accuracy (measured by the closeness of its estimate to the correct value) and higher precision (measured by the low variance in its estimations) than the current state-of-the-art algorithms for mining subgraph statistics. We illustrate these advantages in speed, accuracy and precision using simulations on well-known and widely used graph datasets representing real networks.**

I. Introduction

The analysis of large graphs, such as those representing online social networks, is of increasing scholarly interest to sociologists, mathematicians, economists, computer scientists and marketeers [1]. In particular, mining of large graphs for their microstructure describing patterns of relationships between neighboring vertices, is of significant interest to researchers in data mining [2]–[5]. This microstructure is best captured by motif or graphlet statistics, i.e., the relative frequencies with which different small subgraphs of a certain size appear in the large graph [6]–[13]. For example, the clustering coefficient (the number of triangles in relation to the number of wedges) has long served as an important metric in sociometry and social network analysis [14], [15]. In fact, the relative frequencies of network motifs are indicative of important properties of graphs such as modularity, the tendency of nodes in a network to form tightly interconnected communities, and even play a role in the organization and evolution of networks [6]. Knowledge of these motif statistics combined with homophily, the tendency of similar nodes to connect to one another, add to the ability of businesses such as Facebook to better mine their graphs and monetize their social platforms through targeted advertisements [16].

Computing motif statistics, however, is rendered difficult by two challenges: one computational and the other having to do with restricted access to the full graph data. The computational challenge arises because accurate computation of the relative frequencies of different motifs requires enumeration of all the induced subgraphs and checking each for isomorphism to known motif types. The time complexity of enumerating all induced subgraphs of size k in a graph with V vertices and E edges is exponential in k with an upper bound of $O(E^k)$ and a lower bound of $O(Vc^{k-1})$ [17]. Even when k is as small as 4, in a graph with only millions of edges, the number of motifs can reach hundreds of billions. The other problem is one of restricted access because the data on many large graphs, especially online social networks, can only be obtained piecemeal via the platform's public interface encapsulated in its API for developers on the platform. One common query allowed by most social network APIs is one that returns the list of neighbors of a node — a feature that allows random walks on these large graphs even when the full graph data is unavailable [18].

The computational and the access challenges above motivate the need for an approach to estimating motif statistics via sampling the graph using a random walk and checking only a small fraction of all the induced subgraphs for isomorphism [12], [15], [18], [19].

A. Problem statement

Consider a connected, undirected graph $G = (V, E)$ with vertex set V and edge set E. We assume that information about the graph can only be ascertained through querying each node separately for a list of its neighbors.

For convenience and clarity, we denote each motif by a unique 2-tuple, $M(k, m)$, where k is the number of vertices in the motif and m is the motif id which uniquely identifies a motif given k. Fig. 1 illustrates all motifs with $k \leq 5$.

Let $\mathbf{S}(k)$ denote the set of all connected induced subgraphs with k vertices in G. Similarly, let $\mathbf{S}(k, m)$ denote the set of all connected induced subgraphs which are isomorphic to motif $M(k, m)$. Now, the motif statistics or motif concentrations are given by the relative frequencies of each of the motif types:

$$C(k, m) = \frac{|\mathbf{S}(k, m)|}{|\mathbf{S}(k)|}$$

Given a large graph G, the problem considered in this paper is one of determining $C(k, m)$ for any k and m by visiting

2374-8486/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDM.2016.120

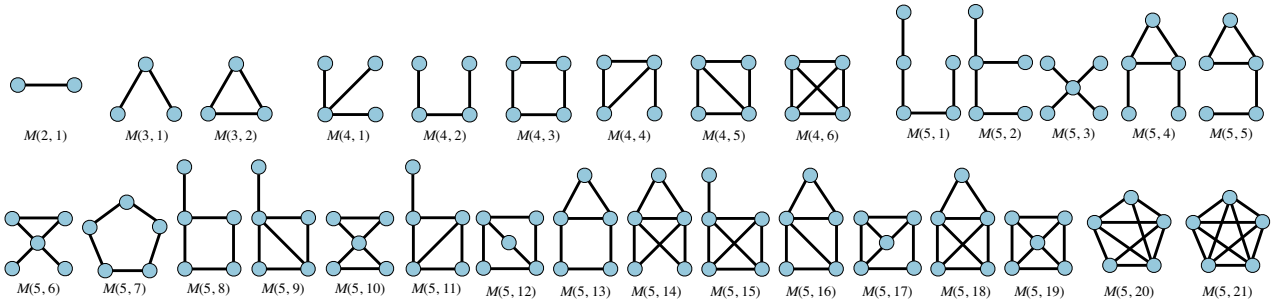


Fig. 1. All 2, 3, 4 and 5 vertices undirected motifs.

nodes in the graph only through its public interface via a random walk. The goal is to make an estimate that is accurate and precise while visiting as few nodes as possible.

B. Related Work

The earliest work on motifs in large graphs began with studies of triadic properties such as triangle counts and the global clustering coefficient [14], [20]. Since then, a large body of work has focused on understanding and estimating the properties of graphs related to 3-node motifs. Yet, computing the accurate statistics of even these smallest of motifs (wedges and triangles) is prohibitively expensive for large graphs, inspiring multiple efforts based on making estimates using edge sampling [21]–[23]. The class of approaches based on random walks, however, solve not only the computational challenge but also the typical restrictions imposed on full access to the graph — they allow piecemeal collection of data by walking the graph querying a node at a time for its list of neighbors [15], [18].

A smaller but increasing body of work has tried to develop graph sampling methods that apply to motifs of size larger than three [7], [9], [13], [24], [25]. Applications in bioinformatics, in particular, have inspired these efforts due to the need for motif detection and motif-related computations in biology [26], [27]. There have been at least two classes of approaches in the estimation of the statistics of larger-size motifs: one based on edge sampling and the other based on random walks. Edge sampling approaches are able to reduce the computational complexity of making an estimation, but they usually require knowledge of global properties of the graph (such as the total number of edges in the graph) or they require access to the full graph. Only methods based on random walks rely entirely on the public interface of live networks and are able to address both the computational and the access challenges mentioned in the previous section.

One approach based on random walks uses the Metropolis-Hastings method [28] which can collect uniformly random nodes to infer motif statistics. However, since nodes selected uniformly randomly may not necessarily induce connected subgraphs, a better approach is to build a graph of connected induced subgraphs (CIS) and conduct a random walk on this CIS graph [7], [11], [12]. Two subgraphs in this CIS graph are directly connected by an edge if they differ in only one node in the original graph. Starting from one subgraph, one can move to a neighboring subgraph by dropping and adding a node without having pre-computed the entire graph of subgraphs. This approach to a subgraph random walk is improved in [7] using a Metropolis-Hastings based sampling method to perform a uniform sampling of CISs in the large graph, leading to a Markov Chain Monte Carlo sampling method for estimating the motif frequency distribution of 3-node, 4-node and 5-node motifs. The use of Metropolis-Hastings for walking the CIS graph is further refined in [11] to collect motif statistics, in an algorithm called the Metropolis-Hastings Random Walk (MHRW).

An alternative approach, also based in random walks on CIS graphs, is one that avoids the Metropolis-Hastings method for its inefficiency involving randomized selections and the consequent rejections of nodes in determining the next step in the walk. Instead, in this approach, the unbiased sampling of Metropolis-Hastings method is replaced with the use of the Horvitz-Thompson construction to unbias the estimation [29]. Such a method is used in [12] which develops the Pairwise Subgraph Random Walk (PSRW), which cleverly samples a set of CISs with a smaller number of $k-1$ nodes to estimate the concentrations of motifs with k nodes.

Both PSRW and MHRW are capable of estimating motif concentrations of any size. As presented in [11], [12], these two algorithms are significantly better than the existing methods in terms of accuracy and speed. However, both of these algorithms rely on some subgraph enumeration which adds significantly to the runtime. The Waddling Random Walk (WRW), proposed in this paper, however, avoids such enumeration and instead uses a randomized approach to sample subgraph and reduce computational costs. WRW achieves a significant improvement in speed as well as in the accuracy and the precision of its estimates.

C. Contributions

We present a new random walk algorithm, called *Waddling Random Walk (WRW)*, named so because it sways left and right and also samples nodes not directly in the path of the random walk. In Section II, we develop the theoretical foundation for the algorithm and show that motif statistics can be inferred

from the probability with which we sample sets of nodes and whether or not the subgraphs induced by those nodes are isomorphic to the motifs of interest.

Section III presents the WRW algorithm to sample k-node motif statistics for any k along with a pseudocode description of it. The algorithm relies on a randomized waddling protocol to sample nodes in the neighborhood of the random walk — a key strength of the algorithm is that the waddling protocol can be customized for specific access or other constraints, with the only requirement being that it be a *randomized* protocol so that the probabilities of sets of nodes selected by the protocol can be computed. Section III also describes the specific version of the algorithm for collecting 4-node and 5-node motif statistics.

Section IV describes a thorough performance analysis of WRW in comparison to the best two algorithms which address the same problem: PSRW introduced in 2014 [12] and MHRW introduced in 2015 [11]. We show that WRW achieves a significantly improved running time. Most importantly, we show, using graph datasets representing real networks, that the WRW algorithm achieves significantly higher accuracy (in terms of the closeness of its answers to the actual values) and higher precision (in terms of the variance in its estimations).

Section V concludes the paper.

II. The Rationale

In this section, we build the theoretical rationale for the Waddling Random Walk. In particular, we illustrate the need for waddling during the random walk by first considering a simpler algorithm without waddling.

A. Preliminaries and Notation

Given a graph $G = (V, E)$, let $v \in V$ denote a vertex in G and let $N(v)$ denote the set of neighbors of vertex v in G. Let $d(v)$ denote the degree of vertex v and let $D = \sum_{v \in V} d(v)$ denote the sum of the degrees of all the vertices in G.

Consider the k-node motif $M(k, m)$. Let $l(k, m)$ denote the number of vertices in the shortest path (allowing repeated vertices) in motif $M(k, m)$ that includes all of the motif's k vertices. For example, $l(4, 1)$ is 5 while $l(4, 2)$ is 4.

A path is called simple if it does not have any repeated vertices. Let $L(k, m)$ denote the number of vertices in the longest simple path of motif $M(k, m)$. For example, $L(4, 1)$ is 3 while $L(4, 2)$ is 4.

Let T_k denote the number of different k-node motifs. For example, $T_3 = 2$, $T_4 = 6$ and $T_5 = 21$.

Let $P_r(k, m, s)$ denote the number of different paths with s vertices (allowing repeats) in motif $M(k, m)$ which include all of the k nodes. For example, $P_r(3, 1, 3)$ is 2 while $P_r(3, 2, 3)$ is 6. Similarly, $P_r(4, 1, 5)$ is 6, $P_r(4, 2, 4)$ is 2 while $P_r(4, 6, 4)$ is 24.

Consider a random walk on G, $(r_1, r_2, \dots)$, where r_1 denotes the starting node and r_i denotes the node visited in step i. Let t denote the number of steps in the random walk required to reach the mixing time [30], i.e., when the probability of visiting a given node in a given step reaches a stationary distribution and is largely independent of the initial node r_1 chosen to begin the random walk. In many real networks, including social networks in particular, t is small and usually of the order of a few hundreds of nodes [15], [28].

Let $\phi_i(v_j)$ denote the probability that the random walk visits node v_j in step i. For $i > t$, the mixing time, we can drop i from the notation and denote by $\phi(v_j)$ the probability that the random walk visits node v_j in any given step. In the rest of this paper, we assume that all the computations are based on observations made in the random walk after the mixing time is reached. As shown in [31], in a random walk, $\phi(v_j)$ is given by:

$$\phi(v_j) = \frac{d(v_j)}{D} \tag{1}$$

Let $\mathbf{R}^{(s)}$ denote the set of all sequences of s nodes which may appear in a random walk in G; it is the set of all s-node paths (allowing revisits to nodes) in G. Let $X^{(s)} = (x_1, x_2, \dots, x_s)$ represent a sequence of s nodes such that $X^{(s)} \in \mathbf{R}^{(s)}$. At any given point in the random walk, let $\phi(X^{(s)})$ denote the probability that it steps through exactly the sequence of nodes $X^{(s)}$. Then, $\phi(X^{(s)})$ is given by:

$$\begin{aligned} \phi(X^{(s)}) &= \frac{d(x_1)}{D}\frac{1}{d(x_1)}\frac{1}{d(x_2)}\cdots\frac{1}{d(x_{s-1})} \\ &= \frac{1}{D}\frac{1}{d(x_2)}\cdots\frac{1}{d(x_{s-1})} \end{aligned} \tag{2}$$

Let $H(X^{(s)})$ denote the subgraph in G induced by the set of vertices in the random walk sequence $X^{(s)}$. If the number of distinct nodes in $X^{(s)}$ is k, then $H(X^{(s)})$ is isomorphic to one of the k-node motifs. Define the function $\omega(X^{(s)}, k, m)$ as follows to indicate if $H(X^{(s)})$ is isomorphic to motif $M(k, m)$:

$$\omega(X^{(s)}, k, m) = \begin{cases} 1 & \text{if } H(X^{(s)}) \text{ is isomorphic to } M(k, m), \\ 0 & \text{otherwise.} \end{cases}$$

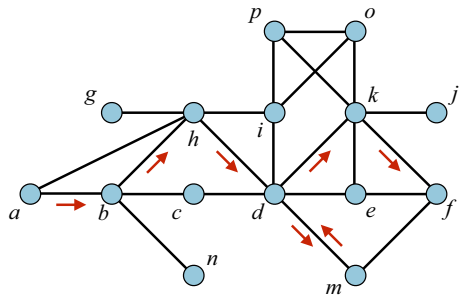
Note that the function $\omega(X^{(s)}, k, m)$ does not depend on the order of the nodes in the sequence $X^{(s)}$.

As we traverse nodes in the random walk, we can observe the sequences of nodes visited and compute the probability that those sequences are encountered using the expression in Eqn. (2) above. Then, we can evaluate $\omega(X^{(s)}, k, m)$ for those sequences to check if they induce a subgraph isomorphic to a certain motif. The rest of this section explains how we can infer motif statistics from these quantities.

B. Motif statistics without waddling

Given a motif $M(k, m)$ which appears in G, there are $P_r(k, m, s)$ ways in which an s-node path may traverse this motif visiting all its nodes. Therefore, if we sum up the function $\omega(X^{(s)}, k, m)$ for every path $X^{(s)} \in \mathbf{R}^{(s)}$, we should get the total number of motifs of type $M(k, m)$ multiplied by $P_r(k, m, s)$. More formally,

$$\sum_{X^{(s)} \in \mathbf{R}^{(s)}} \omega(X^{(s)}, k, m) = P_r(k, m, s)|\mathbf{S}(k, m)| \tag{3}$$



Step	Node visited	4-node path examined	Motif recognized	5-node path examined	Motif recognized
4	*d*	*abhd*	*M*(4,4)	—	None
5	*m*	*bhdm*	*M*(4,2)	*abhdm*	None
6	*d*	*hdmd*	None	*bhdmd*	None
7	*k*	*dmdk*	None	*hdmdk*	*M*(4,1)
8	*f*	*mdkf*	*M*(4,3)	*dmdkf*	None

Fig. 2. An example to illustrate the collection of 4-node motif statistics in a random walk (shown by red arrows) without a waddle. 5-node paths are examined only to check for motif $M(4,1)$.

For any sequence of nodes on the random walk, $X^{(s)}$, define $f(X^{(s)})$ as follows using Eqn. (2):

$$f(X^{(s)}) = \frac{1}{\phi(X^{(s)})D} = d(x_2)d(x_3)\cdots d(x_{s-1}) \quad (4)$$

Let $R_i^{(s)} \in \mathbf{R}^{(s)}$ denote the sequence of s nodes visited during steps $i-s+1$ through i, i.e., $(r_{i-s+1}, r_{i-s+2}, \ldots, r_i)$. Consider the expected value of $\omega(R_i^{(s)},k,m)f(R_i^{(s)})$ over the random walk:

$$\begin{aligned} &E\left[\omega(R_i^{(s)},k,m)f(R_i^{(s)})\right] \\ &\quad= \sum_{X^{(s)}\in\mathbf{R}^{(s)}} \phi(X^{(s)})\left(\omega(X^{(s)},k,m)f(X^{(s)})\right) \\ &\quad= \left(\frac{1}{D}\right)\sum_{X^{(s)}\in\mathbf{R}^{(s)}} \omega(X^{(s)},k,m) \end{aligned} \quad (5)$$

Using Eqn. (4) for $f(R_i^{(s)})$ on the LHS and substituting for the above summation using Eqn. (3) on the RHS, we get:

$$\begin{aligned} &E\left[\omega(R_i^{(s)},k,m)\prod_{j=1}^{s-2} d(r_{i-j})\right] \\ &\quad= \left(\frac{1}{D}\right) P_r(k,m,s)|\mathbf{S}(k,m)| \end{aligned} \quad (6)$$

Eqn. (6) suggests a simple algorithm for sampling motif statistics via a random walk. For each motif type $M(k,m)$, let $s = l(k,m)$, the number of vertices in the shortest path in the motif that includes all of its k vertices. As we visit nodes in the random walk, we can check if the previous s nodes induce a subgraph isomorphic to $M(k,m)$ to evaluate $\omega(R_i^{(s)},k,m)$. Fig. 2 illustrates motifs recognized by such an algorithm during a random walk.

At each step, if the isomorphism test passes, we can compute the product of the degrees of the middle $s-2$ nodes in $R_i^{(s)}$ and obtain the average of these results to get the LHS in Eqn. (6) for $M(k,m)$, which we denote by LHS(k,m). In the RHS of Eqn. (6), since $P_r(k,m,s)$ is known for all the motifs and D is a constant, we can compute the fraction of k-node motifs in a graph which are of a certain type as follows:

$$C(k,m) = \left(\frac{\text{LHS}(k,m)}{P_r(k,m,s)}\right) \Bigg/ \left(\sum_{j=1}^{T_k} \frac{\text{LHS}(k,j)}{P_r(k,j,s)}\right)$$

C. Why waddle?

The algorithm suggested by Eqn. (6) in the previous section works well when $l(k,m)$ is equal to k but can become less accurate when $l(k,m)$ is larger than k. For example, in the case of the 4-star motif or $M(4,1)$, $l(4,1)$ is 5 and so the random walk has to take 5 steps within the motif to encounter and recognize the motif; this means that motifs with larger $l(k,m)$ would be encountered and recognized with lower probability, especially so in large social network graphs with high average degree.

A significant improvement is possible if we allow our random walk to waddle a little (sway left and right) and query random nodes to the right and the left of the random walk as well. For example, consider the random walk illustrated in Fig. 2 to collect 4-node motif statistics. Suppose, in addition to the nodes visited on the random walk, we also query a random neighboring node of each node visited directly on the random walk. Suppose we query node *g* at the step in which the walk visits node *h*. Then, when the walk visits node *d* for the first time, we can recognize the 4-star motif $M(4,1)$ induced by nodes *h*, *b*, *g* and *d* in addition to recognizing motif $M(4,4)$ in the same step induced by nodes *a*, *b*, *h* and *d*.

Waddling, since it also examines nodes not in the direct path of the random walk, allows us to restrict the number of previously visited nodes along the walk that we examine for isomorphism to a motif $M(k,m)$ to no more than the length in the number of nodes, $L(k,m)$, of the longest simple path on the motif. Since $L(k,m) \leq k$, we will sample motifs with a higher probability during every step of the walk. Waddling helps count more motifs and thus improves the accuracy of the motif statistics collected. As we will show in the next section, for best efficiency, how we waddle (i.e., which other nodes we query along the random walk and how deep a chain of nodes we query) depends on the motif for which we are seeking to collect statistics. But, as long as we can correctly compute the probability of choosing the set of nodes for which we examine the induced subgraphs, the methodology detailed in this section can be transferred to the waddling algorithm to estimate the motif statistics.

III. Waddling Random Walk

Algorithm 1 presents the pseudocode of the Waddling Random Walk (WRW) to compute the concentrations of k-node motifs for any k. In our algorithm, we use T_k different

Algorithm 1 Waddling Random Walk

Input: Graph $G = (V, E)$, motif size k, motif id m, random walk length n.
Output: Motif concentration $C(k, m)$

1: $c_m \leftarrow 0$, $1 \leq m \leq T_k$
2: Start and continue random walk until after the mixing time, reaching node r_{i-1} at step $i-1$
3: **while** $i < n$ **do**
4: $\quad r_i \leftarrow$ Random node in $N(r_{i-1})$
5: $\quad$ **for** $m : 1, \ldots, T_k$ **do**
6: $\quad\quad s = L(k, m)$
7: $\quad\quad R_i^{(s)} \leftarrow (r_{i-s+1}, \ldots, r_i)$
8: $\quad\quad$ **if** Nodes in $R_i^{(s)}$ are all distinct **then**
9: $\quad\quad\quad$ **if** $s = k$ **then**
10: $\quad\quad\quad\quad$ **if** $H(R_i^{(s)})$ is isomorphic to $M(k, m)$ **then**
11: $\quad\quad\quad\quad\quad c_m \leftarrow c_m + \left(\prod_{j=1}^{s-2} d(r_{i-j}) \right) \Big/ P_r(k, m, s)$
12: $\quad\quad\quad\quad$ **end if**
13: $\quad\quad\quad$ **else**
14: $\quad\quad\quad\quad$ Choose a random s-node path of $M(k, m)$ and map it on the nodes in $R_i^{(s)}$
15: $\quad\quad\quad\quad W_i^{(k-s)} \leftarrow$ Set of $(k-s)$ nodes chosen using the randomized waddle protocol
16: $\quad\quad\quad\quad$ **if** $H(R_i^{(s)} \cup W_i^{(k-s)})$ is isomorphic to $M(k, m)$ **then**
17: $\quad\quad\quad\quad\quad c_m \leftarrow c_m + \left(\dfrac{Z(k, m) \prod_{j=1}^{s-2} d(r_{i-j})}{\phi(W_i^{(k-s)} | R_i^{(s)}) P_r(k, m, s) P_w(k, m, s)} \right)$
18: $\quad\quad\quad\quad$ **end if**
19: $\quad\quad\quad$ **end if**
20: $\quad\quad$ **end if**
21: $\quad$ **end for**
22: $\quad i \leftarrow i + 1$
23: **end while**
24: $c_t \leftarrow \sum_{j=1}^{T_k} c_j$
25: **return** c_m / c_t

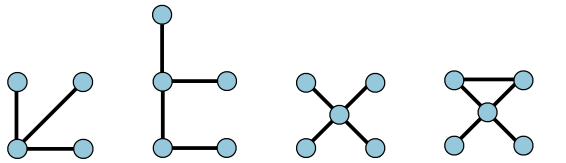
temporary variables, c_m for $1 \leq m \leq T_k$, in which we record the T_k motif concentrations, one for each type. Lines 1–2 in the pseudocode perform necessary initializations, begin the random walk and proceed until the mixing time is reached.

Lines 3–23 describe the walk after the mixing time, during which period we collect the motif statistics. Line 4 takes the next step in the random walk to reach node r_i. At each step of the walk, the `for` loop in lines 5–21 loops through the processing required for each motif type — the loop can be further optimized for computational efficiency; we present the pseudocode as such for clarity at the expense of some efficiency. Consider a motif $M(k, m)$ and let $s = L(k, m)$, the number of vertices in its longest simple path. Note that $s \leq k$. Let $R_i^{(s)} = (r_{i-s+1}, \ldots, r_i)$ denote the sequence of s nodes visited during steps $i - s + 1$ through i.

Lines 8–20, expressed in generalized form, is the heart of the algorithm and we will describe these at length. If the nodes in $R_i^{(s)}$ are *not* all distinct, we will not recognize any motifs of the type being considered and we will move forward to either check for the next motif type or take the next step in the random walk if all motif types at the current step have already been considered.

If the nodes in $R_i^{(s)}$ are all distinct, there are two cases to consider depending on the motif type: $s = k$ and $s < k$. If $s = k$, then there is no need to waddle and we can use the approach in the previous section to check for the isomorphism of $H(R_i^{(s)})$ and $M(k, m)$, and add to the motif count toward estimation of the motif concentration $C(k, m)$. Lines 9–12 handle this case when $s = k$.

In the other case when $s < k$, we can map the longest simple path of the motif $M(k, m)$ onto these $R_i^{(s)}$ nodes (line 14). Note that there are $P_r(k, m, s)$ different assignments that can accomplish the mapping and we randomly choose one of them. To test for isomorphism to $M(k, m)$, we now need



$M(4,1)$	$M(5,2)$	$M(5,3)$	$M(5,6)$
$Z(4,1)=1$	$Z(5,2)=2$	$Z(5,3)=1$	$Z(5,6)=2$
$P_r(4,1,3)=6$	$P_r(5,2,4)=4$	$P_r(5,3,3)=12$	$P_r(5,6,4)=8$
$P_\omega(4,1,3)=1$	$P_\omega(5,2,4)=1$	$P_\omega(5,3,3)=2$	$P_\omega(5,6,4)=1$

Fig. 3. $Z(k,m)$, $P_r(k,m,s)$ and $P_w(k,m,s)$ values for some motifs.

at least an additional $k-s$ nodes. This is accomplished by waddling in line 15 described in greater detail below.

Consider a randomized *waddle* protocol which queries an additional set of $k-s$ choices of nodes, $W_i^{(k-s)}$, such that $R_i^{(s)} \cup W_i^{(k-s)}$ induces a connected subgraph. The querying of these nodes in $W_i^{(k-s)}$, which may be to the right or the left as we take the random walk, produces the waddle for which the algorithm is named. Note that the waddle protocol chooses the nodes $W_i^{(k-s)}$ randomly and cannot guarantee if the induced subgraph will be isomorphic to $M(k,m)$ for some m or even if it has exactly $k-s$ distinct nodes (due to the randomization, it may choose the same node more than once). A powerful feature of our algorithm is that it does not actually prescribe a specific waddle protocol — for the Waddling Random Walk to work, we only need the waddle protocol to be randomized. In fact, the waddle protocol may be customized and optimized for different motifs; we will provide the waddle protocol optimized for 4-node and 5-node motifs later in this section.

Let $\phi(W_i^{(k-s)}|R_i^{(s)})$ denote the probability that the randomized waddle protocol chooses exactly the nodes in $W_i^{(k-s)}$ when the random walk visits the nodes in the sequence $R_i^{(s)}$. In the 4-node motif sampling example illustrated in Fig. 2, if $R_i^{(3)}=(b,h,d)$ and if the waddle protocol randomly chooses a neighbor of h to include in $W_i^{(1)}$, then the probability $\phi(W_i^{(1)}|R_i^{(3)})$ is $1/d(h)$.

Consider $\phi(R_i^{(s)}, W_i^{(k-s)})$, the probability that the random walk visits the sequence of nodes $R_i^{(s)}$ and then the waddle protocol chooses the set of nodes $W_i^{(k-s)}$ at step i of the random walk. Using Eqn. (2), it is given by:

$$\phi(R_i^{(s)}, W_i^{(k-s)}) = \frac{\phi(W_i^{(k-s)}|R_i^{(s)})}{D\prod_{j=1}^{s-2} d(r_{i-j})} \tag{7}$$

Let $H(R_i^{(s)} \cup W_i^{(k-s)})$ denote the subgraph induced by the set of nodes in $R_i^{(s)} \cup W_i^{(k-s)}$. As in the previous section, define the function $\omega(R_i^{(s)} \cup W_i^{(k-s)}, k, m)$ as follows to indicate if $H(R_i^{(s)} \cup W_i^{(k-s)})$ is isomorphic to motif $M(k,m)$:

$$\omega(R_i^{(s)} \cup W_i^{(k-s)}, k, m) = \begin{cases} 1 & \text{if } H(R_i^{(s)} \cup W_i^{(k-s)}) \text{ is} \\ & \quad \text{isomorphic to } M(k,m), \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathbf{W}^{(k-s)}$ be the set of all collections of $k-s$ nodes (allowing repeated nodes) such that if $Y^{(k-s)} \in \mathbf{W}^{(k-s)}$, then for some $X^{(s)} \in \mathbf{R}^{(s)}$, $X^{(s)} \cup Y^{(k-s)}$ induces a connected subgraph in G.

Depending on the motif type, note that the node-to-node mapping of $R_i^{(s)} \cup W_i^{(k-s)}$ to the nodes in the motif may not be possible if the mapping of nodes in $R_i^{(s)}$ to the longest simple path of the motif is reversed, i.e., if the node mapped to r_i is now mapped to r_{i-s+1} and vice-versa and so on. Define $Z(k,m)$ as 1 if the mapping is possible under such a reversal and 2 otherwise; note that $Z(k,m)$ is a property of the motif indicating if the motif is lengthwise symmetric around the longest simple path.

Similarly as in the case of Eqn. (3),

$$\sum_{X^{(s)} \in \mathbf{R}^{(s)}} \sum_{Y^{(k-s)} \in \mathbf{W}^{(k-s)}} \omega(X^{(s)} \cup Y^{(k-s)}, k, m) = \left(\frac{1}{D}\right)\left(\frac{P_r(k,m,s)P_w(k,m,s)|\mathbf{S}(k,m)|}{Z(k,m)}\right) \tag{8}$$

where $P_r(k,m,s)$ is the number of different paths of length s in motif $M(k,m)$, and $P_w(k,m,s)$ is the number of different ways in which nodes in $Y^{(k-s)}$ can then be mapped on to the nodes not on the s-node path in motif $M(k,m)$. While $P_r(k,m,s)$ captures the number of different ways in which an s-node path can be mapped on to the longest simple path of the motif, $P_w(k,m,s)$ captures the number of different ways one can map the additional $k-s$ nodes chosen by the waddle protocol on to the remaining $k-s$ nodes of the motif. Fig. 3 shows examples of some k-node motifs whose longest simple path is of length $s<k$ and the corresponding values of $Z(k,m)$, $P_r(k,m,s)$ and $P_w(k,m,s)$.

Using Eqn. (7), define a function $f(.)$ such that:

$$f(R_i^{(s)}, W_i^{(k-s)}) = \frac{1}{\phi(R_i^{(s)}, W_i^{(k-s)})D} = \frac{\prod_{j=1}^{s-2} d(r_{i-j})}{\phi(W_i^{(k-s)}|R_i^{(s)})}$$

Let $\Omega(k,m,X^{(s)},Y^{(k-s)})$ denote the function given by the product:

$$\phi(X^{(s)},Y^{(k-s)})\omega(X^{(s)} \cup Y^{(k-s)},k,m)f(X^{(s)},Y^{(k-s)})$$

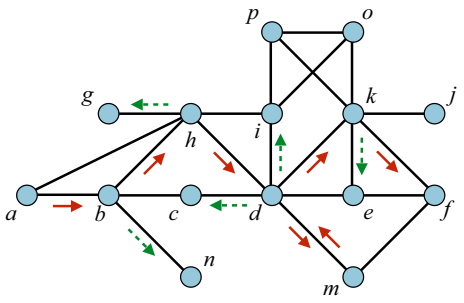
As in Eqn. (5), we have the following expected value:

$$E[\omega(R_i^{(s)} \cup W_i^{(k-s)},k,m)f(R_i^{(s)},W_i^{(k-s)})] = \sum_{X^{(s)} \in \mathbf{R}^{(s)}} \sum_{Y^{(k-s)} \in \mathbf{W}^{(k-s)}} \Omega(k,m,X^{(s)},Y^{(k-s)}) \tag{9}$$

Using Eqns. (7) and (8), we get:

$$E[\omega(R_i^{(s)} \cup W_i^{(s)},k,m)\frac{\prod_{j=1}^{s-2} d(r_{i-j})}{\phi(W_i^{(k-s)}|R_i^{(s)})}] = \left(\frac{1}{D}\right)\left(\frac{P_r(k,m,s)P_w(k,m,s)|\mathbf{S}(k,m)|}{Z(k,m)}\right) \tag{10}$$

Fig. 4 shows an example of how 4-node motif statistics are estimated using a combination of a random walk and a randomized waddle.



Step	Node visited	$R_i^{(4)}$	Motif recognized	$R_i^{(3)}$	$W_i^{(1)}$	Motif recognized
3	*h*	—	None	*abh*	*n*	None
4	*d*	*abhd*	*M*(4,4)	*bhd*	*g*	*M*(4,1)
5	*m*	*bhdm*	*M*(4,2)	*hdm*	*c*	*M*(4,1)
6	*d*	*hdmd*	None	*dmd*	None	None
7	*k*	*dmdk*	None	*mdk*	*i*	*M*(4,1)
8	*f*	*mdkf*	*M*(4,3)	*dkf*	*e*	None

Fig. 4. An example to illustrate the collection of 4-node motif statistics in Waddling Random Walk. The random walk is shown by red arrows and the waddles are shown by dotted-line green arrows.

Lines 16–17 of the pseudocode use Eqn. (10) to compute the sum $|\mathbf{S}(k,m)|$ by computing the LHS of the above equation and using known values of $Z(k,m)$, $P_r(k,m,s)$ and $P_w(k,m,s)$ for each motif. Lines 24–25 finally compute and return the motif concentration.

A. Example: 4 and 5-node motif statistics

While Algorithm 1 shows the pseudocode for the Waddling Random Walk in the general case for k-node motifs, it is illustrative to show how the waddle works in the case of 4 and 5-node motifs. Algorithm 2 and Algorithm 3 replace the lines 3–23 in Algorithm 1 for the 4-node case and 5-node case, respectively.

Algorithm 2 Snippet of WRW for 4-node motifs

1: **while** $i \leq n$ **do**
2: $r_i \leftarrow$ Random node in $N(r_{i-1})$
3: **if** Nodes in $R_i^{(4)}$ are distinct **then**
4: $m \leftarrow$ id of the motif induced by $R_i^{(4)}$
5: $c_m \leftarrow c_m + \dfrac{d(r_{i-1})d(r_{i-2})}{P_r(4,m,4)}$
6: **end if**
7: **if** Nodes in $R_i^{(3)}$ are distinct **then**
8: $w \leftarrow$ Random node in $N(r_{i-1})$
9: **if** $H(R_i^{(3)} \cup w)$ is isomorphic to $M(4,1)$ **then**
10: $c_1 \leftarrow c_1 + \dfrac{d(r_{i-1})^2}{6}$
11: **end if**
12: **end if**
13: $i \leftarrow i+1$
14: **end while**

Lines 5 and 10 in the 4-node case are surprisingly simple compared to the generalized case represented in line 17 of Algorithm 1. This is because $\phi(W_i^{(k-s)}|R_i^{(s)})$ is simply $1/d(r_{i-1})$. The motif type $M(4,1)$ is the only case in which $s = 3$, for which $P_r(4,1,3) = 6$, $P_w(4,1,3) = 1$ and $Z(k,m) = 1$.

In the 5-node case, only motif types $M(5,2)$, $M(5,3)$ and $M(5,6)$ have the number of vertices in their longest simple path less than 5. The corresponding values of P_r, P_w and Z are presented in Fig. 3.

Algorithm 3 Snippet of WRW for 5-node motifs

1: **while** $i \leq n$ **do**
2: $r_i \leftarrow$ Random node in $N(r_{i-1})$
3: **if** Nodes in $R_i^{(5)}$ are distinct **then**
4: $m \leftarrow$ id of the motif induced by $R_i^{(5)}$
5: $c_m \leftarrow c_m + \dfrac{d(r_{i-1})d(r_{i-2})d(r_{i-3})}{P_r(5,m,5)}$
6: **end if**
7: **if** Nodes in $R_i^{(4)}$ are distinct **then**
8: $w \leftarrow$ Random node in $N(r_{i-2})$
9: **if** $H(R_i^{(4)} \cup w)$ is isomorphic to $M(5,2)$ **then**
10: $c_2 \leftarrow c_2 + \dfrac{d(r_{i-1})d(r_{i-2})^2}{2}$
11: **else if** $H(R_i^{(4)} \cup w)$ is isomorphic to $M(5,6)$ **then**
12: $c_6 \leftarrow c_6 + \dfrac{d(r_{i-1})d(r_{i-2})^2}{4}$
13: **end if**
14: **end if**
15: **if** Nodes in $R_i^{(3)}$ are distinct **then**
16: $w_1 \leftarrow$ Random node in $N(r_{i-1})$
17: $w_2 \leftarrow$ Random node in $N(r_{i-1})$
18: $R_{\text{temp}} \leftarrow R_i^{(3)} \cup \{w_1, w_2\}$
19: **if** $H(R_{\text{temp}})$ is isomorphic to $M(5,3)$ **then**
20: $c_3 \leftarrow c_3 + \dfrac{d(r_{i-1})^3}{24}$
21: **end if**
22: **end if**
23: $i \leftarrow i+1$
24: **end while**

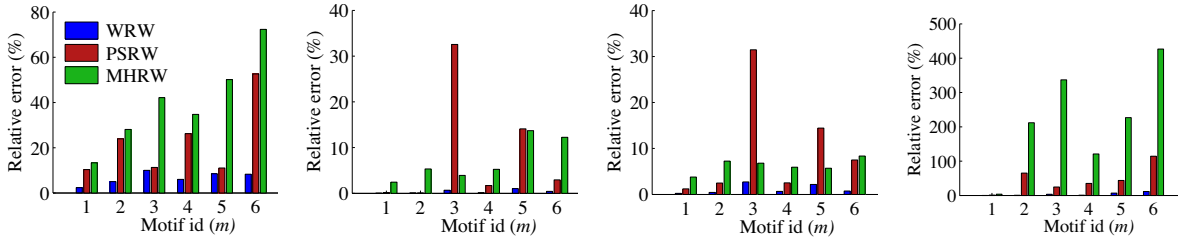
IV. Performance Analysis

We conduct a comparative performance analysis of Waddling Random Walk (WRW)[1] against the best two, both recently proposed, graph sampling algorithms which address the same problem under the same constraints on access to the full graph. The Metropolis-Hastings Random Walk (MHRW) estimates the concentrations of k-node motifs by adopting the Metropolis-Hastings method to perform a uniform random

[1]Source code of WRW and the links to the raw data can be found at `http://www.pages.drexel.edu/%7egh97/WRW/`

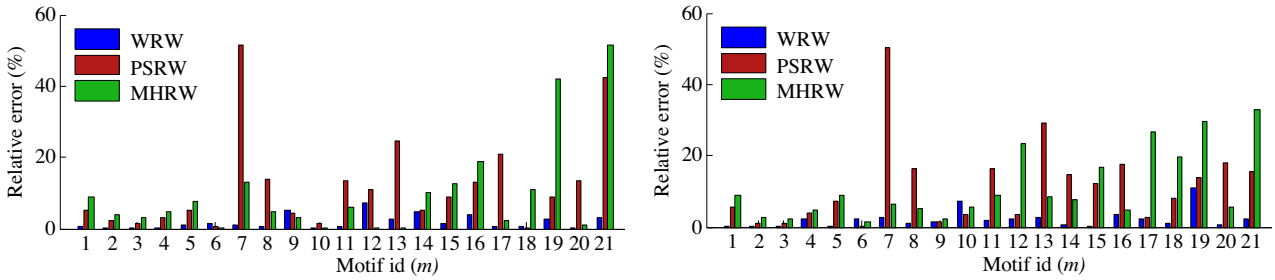
TABLE I
GRAPH DATASETS USED IN THE ANALYSIS.

Graph (LCC)	Nodes $\lvert V\rvert$	Edges $\lvert E\rvert$	$C(4,1)$	$C(4,2)$	$C(4,3)$	$C(4,4)$	$C(4,5)$	$C(4,6)$	$C(5,3)$	$C(5,21)$
com-Amazon	334,863	925,872	6.99e-01	2.10e-01	2.37e-03	7.69e-02	1.05e-02	1.55e-03	7.45e-01	7.24e-06
soc-Slashdot	77,360	469,180	6.86e-01	2.93e-01	1.25e-03	1.86e-02	7.77e-04	9.19e-05	6.15e-01	1.15e-06
socfb-Penn94	41,536	1,362,220	6.52e-01	3.04e-01	1.30e-03	3.93e-02	2.16e-03	3.59e-04	6.18e-01	2.30e-06
com-Youtube	1,134,890	2,987,624	9.82e-01	1.57e-02	3.98e-05	2.12e-03	3.80e-05	8.55e-07	—	—



(a) com-Amazon($Q = 8K$) (b) soc-Slashdot($Q = 8K$) (c) socfb-Penn94($Q = 8K$) (d) com-Youtube($Q = 20K$)

Fig. 5. Comparison of the relative errors in the estimates of 4-node motif concentrations.



(a) soc-Slashdot($Q = 18K$) (b) socfb-Penn94($Q = 18K$)

Fig. 6. Comparison of the relative errors in the estimates of 5-node motif concentrations.

sampling of connected induced subgraphs (CIS) with k nodes in the large graph [11]. The Pairwise Subgraph Random Walk (PSRW), samples a set of CISs with $k-1$ nodes by walking on the graph of these CISs to estimate the k-node motif statistics [12]. Both PSRW and MHRW, like WRW, are capable of estimating motif concentrations of any size.

We performed our experiments on real graphs from the Stanford Network Analysis Project (SNAP) [32] and the koblenz network collection [33]. For each graph dataset used, we run the random walk algorithms on the largest connected component (LCC) of it. The name, the number of vertices and the number of edges in these graphs are listed in Table I along with the actual values of 4-node and selected 5-node motif concentrations.

We conduct our comparative analysis based on four key metrics: the number of queries, the run time, the accuracy (how close is the estimate to the correct answer?), and the precision (how low is the variance in the estimates?). The first two address the speed of the algorithms and the latter two address the confidence we should have in the estimates. If the amount of processing done per query by the algorithms are different, the number of queries is not directly indicative of the speed of the algorithm — therefore, we also use the actual run time of the algorithms in our analysis. The number of queries, however, is still meaningful as a metric since it indicates the amount of information collected by the algorithm.

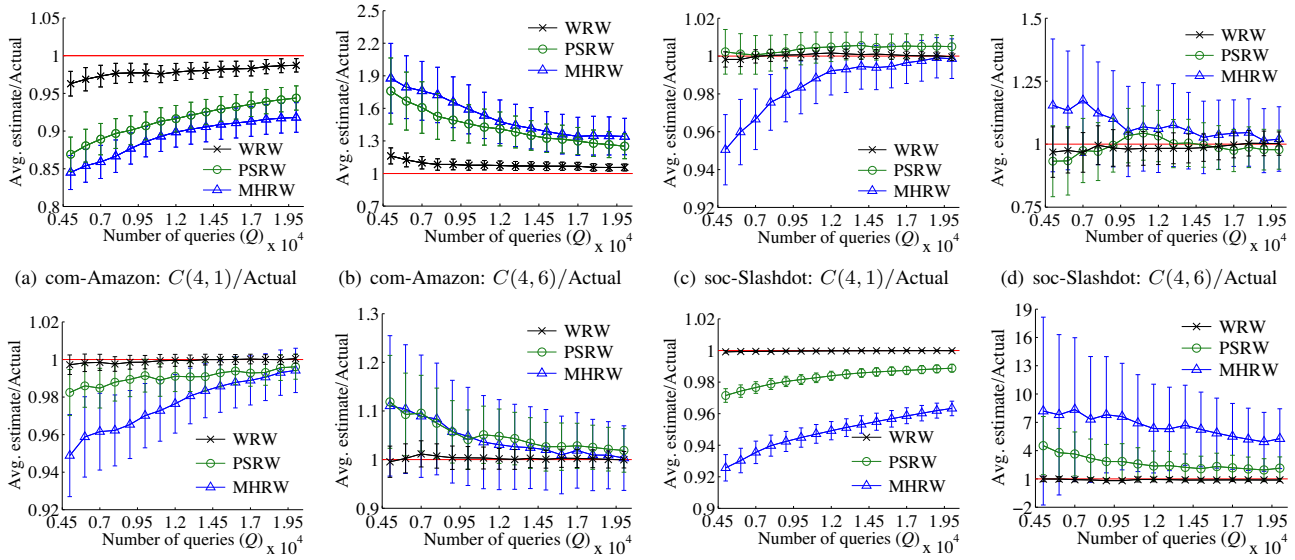
A. Accuracy and precision

Figs. 5 and 6 show relative errors in estimating the concentrations of each of the 4-node and 5-node motifs for each of the three algorithms. We measure the relative error as:

$$\text{Relative error} = \frac{\text{Average estimate} - \text{Actual value}}{\text{Actual value}}$$

The average estimate is calculated as the mean of the estimated value over 200 independent runs.

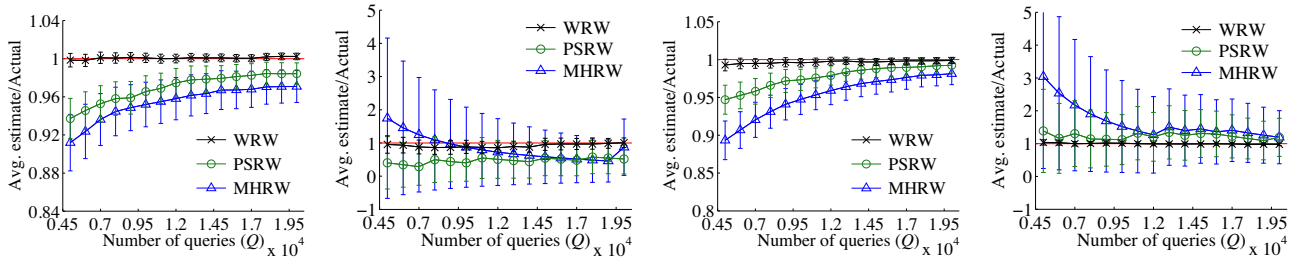
For each graph, we fixed Q, the number of queries. For PSRW and MHRW, since the CIS sampled in the next step differs from the CIS of the current step in only one node, we assume only one query per CIS considered. As shown in Figs. 5 and 6, for almost all motif types, the WRW algorithm proposed in this paper yields a smaller relative error, i.e., higher accuracy, for the same number of queries than the other two algorithms, and especially so when the actual concentration of the motif is low.



(a) com-Amazon: $C(4,1)$/Actual (b) com-Amazon: $C(4,6)$/Actual (c) soc-Slashdot: $C(4,1)$/Actual (d) soc-Slashdot: $C(4,6)$/Actual

(e) socfb-Penn94: $C(4,1)$/Actual (f) socfb-Penn94: $C(4,6)$/Actual (g) com-Youtube: $C(4,1)$/Actual (h) com-Youtube: $C(4,6)$/Actual

Fig. 7. Comparison of the ratio of the average estimated values of $C(4,1)$ and $C(4,6)$ and their actual values. Red line indicates 1. The error bars indicate 95% confidence intervals over 200 independent runs.



(a) soc-Slashdot: $C(5,3)$/Actual (b) soc-Slashdot: $C(5,21)$/Actual (c) socfb-Penn94: $C(5,3)$/Actual (d) socfb-Penn94: $C(5,21)$/Actual

Fig. 8. Comparison of the ratio of the average estimated values of $C(5,2)$ and $C(5,3)$ and their actual values. Red line indicates 1. The error bars indicate 95% confidence intervals over 200 independent runs.

Figs. 7 and 8 help evaluate both the accuracy and the precision of the WRW algorithm in comparison to PSRW and MHRW in 4-node case and 5-node case. They show the ratio of the average estimated motif concentration to the actual value for each of the four graphs with increasing number of queries.

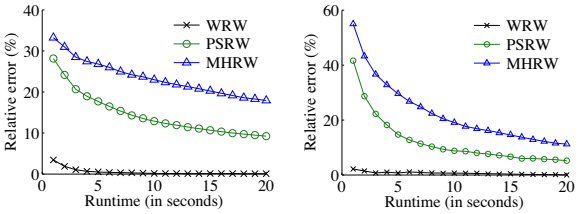
In large graphs, the star motifs, e.g., $M(4,1)$ and $M(5,3)$, typically have the largest concentration. The clique motifs, such as $M(4,6)$ and $M(5,21)$, have the smallest concentration in most cases. In order to demonstrate accuracy and precision for the spanning the full range of actual motif concentrations, we choose to by plotting the estimates for $C(4,1)$ and $C(4,6)$ in case of 4-node motifs (in Fig. 7) and $C(5,3)$ and $C(5,21)$ in case of 5-node motifs (in Fig. 8). When the concentration is high, it is easier to reach good accuracy and precision since the random walk will encounter more samples. But, note that, as also shown in Figs. 5 and 6, the WRW algorithm achieves especially good accuracy and precision when doing so is harder, i.e., when the motif concentration is very low.

The closeness of the WRW plot to the red line indicates its significantly better accuracy than the other algorithms. Also, the smaller error bars on the WRW plot show that, besides being more accurate, the estimates made by WRW are also more precise compared to the other algorithms.

B. Runtime

The WRW algorithm achieves an improvement in the runtime by avoiding the enumeration of subgraphs but instead simply picking a random set of nodes and checking for isomorphism. This makes a particular difference in the case of graphs with high average node-degree.

We implemented the three algorithms, WRW, PSRW and MHRW, in python using *iGraph* routines. Since part of the point of graph sampling is to make Big Data analysis feasible on ordinary desktops, we ran all of the simulations on an iMac with 8GB 1600MHz DDR3 memory and 2.7GHz Intel Core i5 processor. We assume that the graph datasets are stored on the local machine and, so, the time lost to querying corresponds to the time involved in accessing the memory.



(a) com-Youtube: $C(4,1)$ (b) socfb-Penn94: $C(5,3)$

Fig. 9. Comparisons of the relative error in estimating $C(4,1)$ and $C(5,3)$ against the runtime in seconds.

Fig. 9 plots the relative error of the estimates of $C(4,1)$ and $C(5,3)$ made by the three algorithms against the total runtime, averaged over 100 independent runs on com-Youtube and socfb-Penn94 graphs. The figure shows that WRW achieves significantly better accuracy for the same runtime than other algorithms in both 4-node and 5-node motif cases.

V. Conclusions

This paper demonstrates a simple approach, based on a random walk, to collect and estimate the motif statistics of a large graph by sampling only a small fraction of the motifs in the graph. The algorithm, called Waddling Random Walk, is significantly faster than other known algorithms that address both the computational and access challenges in the subgraph mining of large graphs. Besides the improvement in speed, the algorithm is also more accurate (its estimates are closer to the correct answer) and more precise (its estimates have low variance) than the best of previously known algorithms.

A powerful feature of our methodology is that it also offers a generalized approach which can be customized to optimize for specific motifs of interest. In fact, the theoretical rationale used for our approach only requires that our method of waddling used in the algorithm be a randomized protocol. Within this approach, there is much room for improving the waddling protocol and we hope this paper will form the foundation for new approaches leading to newer and better algorithms in the computationally feasible analysis of large graphs.

Acknowledgment

This work was partially funded by the National Science Foundation Award #1250786.

References

[1] T. Tiropanis, W. Hall, J. Crowcroft, N. Contractor, and L. Tassiulas, "Network science, web science, and Internet science," *Commun. ACM*, vol. 58, no. 8, pp. 76–82, Jul. 2015.

[2] S. Chu and J. Cheng, "Triangle listing in massive networks and its applications," in *ACM KDD*, 2011, pp. 672–680.

[3] E. R. Elenberg, K. Shanmugam, M. Borokhovich, and A. G. Dimakis, "Beyond triangles: A distributed framework for estimating 3-profiles of large graphs," in *ACM KDD*, 2015, pp. 229–238.

[4] R. Zou and L. B. Holder, "Frequent subgraph mining on a single large graph using sampling techniques," in *Proc. ACM Workshop on Mining and Learning with Graphs*, 2010, pp. 171–178.

[5] A. Silva, W. Meira, Jr., and M. J. Zaki, "Structural correlation pattern mining for large graphs," in *ACM Workshop on Mining and Learning with Graphs*, 2010, pp. 119–126.

[6] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[7] M. Bhuiyan, M. Rahman, M. Al Hasan *et al.*, "Guise: Uniform sampling of graphlets for large graph analysis," in *IEEE Int'l Conf. Data Mining (ICDM)*, 2012, pp. 91–100.

[8] J. Ugander, L. Backstrom, and J. Kleinberg, "Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections," in *WWW*, 2013, pp. 1307–1318.

[9] M. Jha, C. Seshadhri, and A. Pinar, "Path sampling: A fast and provable method for estimating 4-vertex subgraph counts," in *WWW*, 2015, pp. 495–505.

[10] P. Wang, J. Tao, J. Zhao, and X. Guan, "Moss: A scalable tool for efficiently sampling and counting 4-and 5-node graphlets," *arXiv preprint arXiv:1509.08089*, 2015.

[11] T. K. Saha and M. Al Hasan, "Finding network motifs using MCMC sampling," in *Complex Networks VI*. Springer, 2015, pp. 13–24.

[12] P. Wang, J. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan, "Efficiently estimating motif statistics of large networks," *ACM TKDD*, vol. 9, no. 2, p. 8, 2014.

[13] P. Wang, J. Lui, and D. Towsley, "Minfer: Inferring motif statistics from sampled edges," *arXiv preprint arXiv:1502.06671*, 2015.

[14] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM Computing Surveys (CSUR)*, vol. 38, no. 1, p. 2, 2006.

[15] S. J. Hardiman and L. Katzir, "Estimating clustering coefficients and size of social networks via random walk," in *WWW*, 2013, pp. 539–550.

[16] J.-C. Wang and C.-H. Chang, "How online social ties and product-related risks influence purchase intentions: A Facebook experiment," *Electronic Commerce Res. and Appl.*, vol. 12, no. 5, pp. 337–346, 2013.

[17] R. Itzhack, Y. Mogilevski, and Y. Louzoun, "An optimal algorithm for counting network motifs," *Physica A: Statistical Mechanics and its Applications*, vol. 381, pp. 482–490, 2007.

[18] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Walking in Facebook: A case study of unbiased sampling of OSNs," in *IEEE INFOCOM*, 2010, pp. 1–9.

[19] B. Ribeiro and D. Towsley, "Estimating and sampling graphs with multidimensional random walks," in *ACM Conf. Internet Measurement*, 2010, pp. 390–403.

[20] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*. Cambridge University Press, 1994.

[21] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos, "Doulion: Counting triangles in massive graphs with a coin," in *ACM KDD*, 2009, pp. 837–846.

[22] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella, "Graph sample and hold: A framework for big-graph analytics," in *ACM KDD*, 2014, pp. 1446–1455.

[23] M. Jha, C. Seshadhri, and A. Pinar, "A space-efficient streaming algorithm for estimating transitivity and triangle counts using the birthday paradox," *ACM TKDD*, vol. 9, no. 3, p. 15, 2015.

[24] S. Wernicke, "Efficient detection of network motifs," *IEEE/ACM Trans. Computational Biology and Bioinformatics (TCBB)*, vol. 3, no. 4, pp. 347–359, 2006.

[25] M. Rahman, M. Bhuiyan, and M. A. Hasan, "Graft: an approximate graphlet counting algorithm for large graph analysis," in *ACM Int'l Conf. Information and Knowledge Management*, 2012, pp. 1467–1471.

[26] W. Kim, M. Li, J. Wang, and Y. Pan, "Biological network motif detection and evaluation," *BMC Systems Biology*, vol. 5, no. 3, p. 1, 2011.

[27] S. Panni and S. E. Rombo, "Searching for repetitions in biological networks: methods, resources and tools," *Briefings in bioinformatics*, vol. 16, no. 1, pp. 118–136, 2015.

[28] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On unbiased sampling for unstructured peer-to-peer networks," *IEEE/ACM Trans. on Netw.*, vol. 17, no. 2, pp. 377–390, 2009.

[29] D. G. Horvitz and D. J. Thompson, "A generalization of sampling without replacement from a finite universe," *J. American Statistical Association*, vol. 47, no. 260, pp. 663–685, 1952.

[30] L. Lovász and P. Winkler, "Mixing times," *Microsurveys in discrete probability*, vol. 41, pp. 85–134, 1998.

[31] L. Lovász, "Random walks on graphs," *Combinatorics, Paul Erdos is eighty*, vol. 2, pp. 1–46, 1993.

[32] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[33] "Konect datasets: The koblenz network collection," http://konect.uni-koblenz.de/networks, May 2015.