2009 Fifth IEEE International Conference on e-Science

Strategies for Network Motifs Discovery

Pedro Ribeiro, Fernando Silva
CRACS & INESC-Porto LA
Faculdade de Ciências, Universidade do Porto
R. Campo Alegre, 1021/1055, 4169-007 Porto, Portugal
{*pribeiro, fds*}*@dcc.fc.up.pt*

Marcus Kaiser
School of Computing Science, Newcastle University, UK
Institute of Neuroscience, Newcastle University, UK
Dept. of Brain and Cognitive Sci., Seoul Nat. Univ., Korea
m.kaiser@newcastle.ac.uk

***Abstract*—Complex networks from domains like Biology or Sociology are present in many e-Science data sets. Dealing with networks can often form a workflow bottleneck as several related algorithms are computationally hard. One example is detecting characteristic patterns or "network motifs" – a problem involving subgraph mining and graph isomorphism. This paper provides a review and runtime comparison of current motif detection algorithms in the field. We present the strategies and the corresponding algorithms in pseudo-code yielding a framework for comparison. We categorize the algorithms outlining the main differences and advantages of each strategy. We finally implement all strategies in a common platform to allow a fair and objective efficiency comparison using a set of benchmark networks. We hope to inform the choice of strategy and critically discuss future improvements in motif detection.**

***Keywords*-Network Motifs; Graph Mining; Algorithms; Complex Networks**

I. Introduction

Complex networks have recently gained the attention of the research community due to the discovery that real-world networks have non-trivial topological features [1]–[5]. Applications of the concepts, models and techniques developed have been applied in many different disciplines like sociology, biology, physics, mathematics and computer science. This is because virtually almost every natural structure can be represented as a network. From this inherent interdisciplinarity there are many angles in which one can approach the networks. After knowing how to represent the structure that they want to study, scientists have available a paraphernalia of different measures to mine interesting characteristics and useful quantitative data [6]–[8].

From the beginning, typical properties studied were aggregated node measures intended to discover global features, like the degree distribution, diameter, average distance or clustering coefficient. However, this was not enough to bridge the gap between structure and function, and further study of more local and basic structures of the networks was needed. The focus passed from single and individual nodes to groups of nodes, forming subnetworks. It was with this in mind that the term "network motifs" first appeared [9]. Dubbed the "building blocks" of complex networks, they mimicked the concept of sequence motifs used in genomics, but extending it. In a sequence, a motif is a recurring subsequence, a pattern, that is conjectured to have some functional significance. In a network, a motif is therefore a recurring subnetwork conjectured to have some significance. In particular, it is a recurring subnetwork that appears with a higher frequency than it would be expected in similar random networks. A more concise and formal definition is given in section III. Note that this is substantially different from the problem of *discovering frequent subgraphs* [10], [11] (also named as motifs by some authors), which is normally applied to groups of graphs (instead of single graphs), counts the number of graphs in which a subgraph appears (but does not count the number of occurrences in each graph) and uses algorithmic techniques mostly based on the "a-priori" principle [12].

The seminal paper by Milo et al. [9] gave origin to a multitude of definitions and studies. Network motifs have since been used in the most varied areas. The concept has been applied to networks in domains like protein-protein interaction (PPI) [13], gene transcriptional regulation [14], [15], food webs [16], brain [17], electronic circuits [18] and software [19]. It should however be said that network motifs did not come without criticism [20]–[22]. We do not try to advocate any position; instead we focus our attention on studying efficient strategies for finding network motifs.

Like many other subgraph problems (such as finding maximal independent or bipartite sets), finding these motifs is a computationally hard task, because fundamentally we will be matching graph patterns with the desired motifs, which leads to the well known problem of graph isomorphism, with no polynomial time algorithm known [23]. As the size of the motifs increases, the time needed to calculate them grows exponentially. Hence, an exhaustive computation of all motifs of a network is typically reduced to very small sizes in order to obtain results in a reasonable amount of time. In this article, we survey the state-of-the-art strategies that have been proposed to find network motifs. Another survey [24] shares similar goals although more restricted to motif discovery in protein-protein networks. Our work differs from it in many ways: (i) we give a more formal definition of a network motif; (ii) we build a taxonomy for several concepts used; (iii) we use this taxonomy to build a table in which one can quickly identify the main differences between the strategies currently used; (iv) we write the main algorithms in pseudo-code thus enabling a clear understanding of their inner works; and (v) we compare

978-0-7695-3877-8/09 $26.00 © 2009 IEEE
DOI 10.1109/e-Science.2009.20

the main approaches in an integrated common framework that gives us the ability to isolate external factors and focus on the real practical differences between the various strategies.

The remainder of the article is organized as follows. In section II we introduce all needed terminology to approach the network as a mathematical object. Section III gives concrete informal and formal definitions for network motifs. Section IV details the different strategies and algorithms used for motif finding, giving pseudo-code and a table that quickly identifies the main differences between these methods. Section V compares the execution times in an integrated common framework that isolates the real differences between the different approaches. Section VI concludes the paper.

II. Network terminology

In order to have a well defined and coherent graph terminology throughout the paper, this section reviews the main concepts we will use. A network can be modeled as a *graph* G composed by the set $V(G)$ of *vertices* or *nodes* and the set $E(G)$ of *edges* or *connections*. The *size* of a graph is the number of vertices and is written as $|V(G)|$. A k-graph is a graph of size k. The *neighborhood* $N(u)$ of a vertex $u \in V(G)$ is composed by the set of vertices $v \in V(G) : (u, v) \in E(G)$. All vertices are assigned consecutive integer numbers starting from 0. The comparison $v < u$ means that the index of v is lower than that of u.

A *subgraph* G_k of a graph G is a graph of size k in which $V(G_k) \subseteq V(G)$ and $E(G_k) \subseteq E(G)$. This subgraph is said to be *induced* when $(u, v) \in E(G_k) \leftrightarrow (u, v) \in E(G)$. The neighborhood of a subgraph G_k is defined as $N(G_k) = \{v \in N(u) : v \notin V(G_k), u \in V(G_k)\}$. The *exclusive neighborhood* of a vertex u relative to a subgraph G_k is defined as $N_{excl}(u, G_k) = \{v \in N(u) : v \notin V(G_k) \cup N(G_k)\}$.

A *mapping* of a graph is a bijection where each vertex is assigned a value. Two graphs G and H are said to be *isomorphic*, denoted as $G \sim H$, if there is a one-to-one mapping between the vertices of both graphs where two vertices of G share an edge if and only if their corresponding vertices in H also share an edge. The set of isomorphisms of a graph into itself is called the set of *automorphisms* and is denoted as $Aut(G)$. Two vertices are said to be *equivalent* when there exists some automorphism that maps one vertex into the other. This equivalence relation partitions the vertices of a graph G into equivalence classes denoted as G_E.

III. The concept of network motifs

A. The original definition

Milo et al. [9] provided a basic informal definition for "network motifs":

Definition 1 (Network Motifs - informal definition): Network motifs are patterns of inter-connections occurring in complex networks in numbers that are significantly higher than those in similar randomized networks.

For the sake of simplicity, from now on we will refer to network motifs simply as motifs. The idea and motivation behind this definition was that the detection of the motifs could give new insight into their dynamical and functional behaviour. A possible interpretation was that the motifs appeared because of constraints in the way the network was developed [25], thus being related to the evolution of the whole complex system. Another possible use would be to distinguish classes of networks based on types of motifs found (this was done in [9], [26]), since it can be universally applied to all kinds of networks.

The definition above means that a motif is a subnetwork which is statistically over-represented. Next, we describe how the informal definition was put to practice in Milo et al. [9] supplementary material. The key aspect to ensure statistical meaning is to be able to generate the random networks as similar as possible to the original one. We want to be sure that the intrinsic global and local properties of the network do not determine the motif appearance and that it is specific to this particular network. The original proposal was therefore to maintain all single-node properties, namely the *in* and *out* degrees. Besides that, one should try to guarantee that when searching for k-motifs, the frequency of $(k-1)$-motifs would be the same, ensuring that the significance of a particular pattern does not simply derive from its subpatterns.

A motif is classified as such when three things occur. First, the probability that the number of times a motif appears on randomized network is greater than the number of times it appears in the real network should be smaller than a determined *probability* threshold P. This is determined using an ensemble of a large number of random networks as described before and will ensure that the motif is over-represented on the original network. P is estimated by assuming a random null hypothesis and z-scores (on a standard normal distribution). Let $f_{original}$ be the frequency in the original network and f_{random} be the frequency in a random network. We can then define the z-score as in equation 1 and use a pre-calculated table to infer the desired probability.

$$z\text{-}score(G_k) = \frac{f_{original} - \bar{f}_{random}}{std(f_{random})} \tag{1}$$

The second constraint is that $f_{original}$ should be higher than an *uniqueness* threshold U. This ensures a quantitative minimum to establish significance.

The third and last constraint is that $f_{original}$ is significantly larger than $\bar{f}_{random}$ in order to prevent the detection of motifs that have a small difference between these two values but have a narrow distribution in the random networks ($f_{original} - \bar{f}_{random} > D \times \bar{f}_{random}$ was used). D is the proportional threshold that ensures the minimum difference between $f_{original}$ and $\bar{f}_{random}$.

This leads us finally to a more formal version of motifs, expressed in the following definition:

Definition 2 (Network Motifs - formal definition): An induced subgraph G_K of a graph G is called a network

motif when for a given set of parameters $\{P, U, D, N\}$ and a random ensemble of N similar networks:

1) $Prob(\bar{f}_{random}(G_K) > f_{original}(G_K)) \leq P$ **(Over-representation)**
2) $f_{original}(G_K) \geq U$ **(Minimum frequency)**
3) $f_{original}(G_K) - \bar{f}_{random}(G_K) > D \times \bar{f}_{random}(G_K)$ **(Minimum deviation)**

Milo et al. [9] used $\{0.01, 4, 0.1, 1000\}$ as the set of parameters $\{P, U, D, N\}$, but other values can be used depending on what we want to accomplish with the motifs.

B. Variations

All of the aforementioned conditions constitute the original complete definition of motifs that is still probably regarded as the canon. However, several variations were introduced, varying different types of new details and constraints.

The first point to note is that the concept is very broad. It can be applied equally to *directed* or *undirected* networks without changing its behavior and method of calculation. One other possible application is to use the definition on coloured networks, giving origin to *coloured motifs* [27], [28].

A subtle variation is the notion of an *anti-motif* [26], which is a significantly under-represented subnetwork and may also be meaningful. Some authors take into account subnetworks which are not induced [29], [30], meaning that the total number of possible subnetworks will be much larger. Other authors only consider *maximal motifs* [31], which are motifs that do not contain as subgraphs other motifs, thus diminishing the total number of motifs found.

Another important variation is related to how we determine the frequency of a specific subgraph. Milo et al. [9] allow arbitrary overlapping of edges and nodes on different occurrences of a motif, which makes it the canon definition. This is typical in biological applications, since in these networks it is possible for several different overlapping subgraphs to be active and functioning at the same time, with the same motif assuming different functions on each occurrence, as for example is the case of proteins in PPI networks [24]. However, Schreiber et al. [32] introduces different concepts for the frequency, allowing more constraints, like no sharing of edges and nodes. This has the potential to drastically change the frequency of a motif, thus changing the tractability of the motif discovery problem.

The notion of the statistic significance can have a different formulation. Instead of using the z-score, some authors also use the so called *abundance* (Δ) [26], as defined in equation 2 (ϵ is a very small constant to ensure that when the frequency is small the abundance will not be misleadingly large).

$$\Delta(G_K) = \frac{f_{original} - \bar{f}_{random}}{f_{original} + \bar{f}_{random} + \epsilon} \tag{2}$$

Another approach is to sample some of the subgraphs and then estimate the concentrations of the studied motifs in the original network [33], as detailed in equation 3, where the denominator indicates the sum of the frequencies of all the k-subgraphs. Calculating the average and standard deviation of the concentration on the random ensemble of networks will then give an estimate on the z-score.

$$C(G_K) = \frac{f(G_K)}{\sum f(G_K)} \tag{3}$$

The generation of a random ensemble of similar networks is cumbersome and time consuming. Analytical methods to derive the desired probability for a subgraph to be over-represented would be most welcome and recent work pursues this line of research [34]–[37]. These theoretical statistical approaches have an enourmous potential, but they still require further development to reach the accuracy needed for a practical application.

IV. Strategies for motifs discovery

A. Overview

We surveyed the most currently used sequential strategies for motifs discovery, selecting those that rely on the canon definition of motif and are the basis of almost every motif application found in the literature. Table I summarizes the main differences among the strategies on a set of relevant distinguishable parameters.

Table I
CLASSIFICATION OF THE SELECTED STRATEGIES FOR NETWORK MOTIFS DISCOVERY.

Method	Bias	Symmetry Breaking	Network Centric	Public Tool	Motif Size
MFinder [9], [33]	yes	no	yes	yes	small
FanMod [35], [38]	no	yes	yes	yes	medium
Grochow [39]	yes	yes	no	no	large

Method indicates the name of the associated tool or the author name when no production tool name is given. **Bias** indicates whether the sampling method does not have a uniform probability of selecting a subgraph. **Symmetry Breaking** indicates if the method only finds once each occurrence of the motif, avoiding redundant calculations due to symmetry. **Network Centric** indicates whether the method must be applied to the whole network and all motifs or if it can be applied to a single specific motif. **Public Tool** indicates whether the method has available a production software tool, ready to be used by anyone wishing to do so. **Motif Size** gives a very broad overview of the size of the motifs that the respective algorithms can process in a reasonable amount of time. We considered `small` for subgraph sizes of around 4-7, `medium` for 6-9 and `large` for sizes >10.

B. The strategies

The main flow of all the strategies is basically the same. To look for motifs of size k, first enumerate all k-subgraphs of the original graph and then calculate a *subgraph census*. This results in a histogram with the frequency of the classes of isomorphic k-subgraphs. After

that an ensemble of similar random networks is generated. These random networks maintain the same global properties of the original graphs, namely the node degrees. A *subgraph census* is performed on each of the random networks. Finally, with all the gathered data, the statistical significance of the motifs on the original network is evaluated and the ones over-represented are reported.

Let us examine how the methods accomplish this flow of execution. Figures 1 and 2 illustrate the `MFinder` and `FanMod` algorithmic strategies for subgraph census. Both use a recursive backtracking algorithm.

Require: Graph G and a positive integer k
Ensure: k-subgraphs census of graph G
1: **for all** $(i, j) \in E(G)$ **do**
2: SEARCHSUBSET($\{i, j\}$)

3: **procedure** SEARCHSUBSET(S)
4: **if** $|S| = k$ **then**
5: **if** UNIQUE(S) **then**
6: INCREMENTCOUNT(subgraphId(S))
7: **else**
8: HASH.INSERT(S)
9: **for all** $i \in S$ **do**
10: **for all** $(i, k) \in E(G)$ **do**
11: **if** $k \notin S$ **then**
12: **if** HASH.NOTFOUND($S \cup \{k\}$) **then**
13: SEARCHSUBSET($S \cup \{k\}$)

Figure 1. `MFinder` enumeration algorithm.

`MFinder` starts by choosing an edge (line 1) and constructs the motif starting with its two constituent nodes (line 2). Each time it adds a new node that has an edge connected to the already partially constructed subgraph (lines 9 to 13). Whenever the desired subgraph size is achieved, the *id* of the corresponding isomorphism class is calculated and its frequency is updated in an hash table (line 6). To make the search more efficient, another hash table is maintained (in fact, one different hash table for each subgraph size smaller than k) to determine all sets of vertices (subgraphs) already found. This is used to avoid expanding again from a vertex set already explored (lines 8 and 12). Even with this, the same subgraph can be found several times (due to symmetries) and a test is made to certify that this is indeed a new uncounted motif (line 5). `MFinder` needs a lot of memory to maintain all the subgraphs explored, which hinders its capabilities to deal with large motifs.

`FanMod` appeared later and innovated precisely because it was able to avoid symmetries and to find all subgraphs exactly just once. Instead of starting with an edge, this method starts with a single "root" node and expands from there. Its core idea is that when expanding the set of nodes, only the ones with an index greater than the initial spawning node are allowed (lines 2 and 10). A list of possible vertices for extension is maintained (lines 2 and 11) and each time a vertex is chosen for expansion it is removed from the possible extensions (line 9) and its exclusive neighbours are added to the new possible extensions (line 10). The fact that they are exclusive guarantees that each subgraph is enumerated exactly only once, because the ones which are not exclusive will be added on another instance of the recursion. Figure 3 exemplifies in detail how the algorithm enumerates all 3-subgraphs of a graph with 5 vertices.

Require: Graph G and a positive integer k
Ensure: k-subgraphs census of graph G
1: **for all** $v \in V(G)$ **do**
2: $V_{Ext} \leftarrow \{u \in N(v) : u > v\}$
3: EXTENDSUBGRAPH($\{v\}, V_{Ext}, v$)

4: **procedure** EXTENDSUBGRAPH(V_{Subg}, V_{Ext}, v)
5: **if** $|V_{Subg}| = k$ **then**
6: INCREMENTCOUNT(canonicalLabeling(V_{Subg}))
7: **else**
8: **while** $V_{Ext} \neq \emptyset$ **do**
9: remove random chosen $w \in V_{Ext}$
10: $V'_{new} \leftarrow \{u \in N_{excl}(w, V_{subg}) : u > v\}$
11: $V'_{ext} \leftarrow V_{ext} \cup V'_{new}$
12: EXTENDSUBGRAPH($V_{subg} \cup \{w\}, V'_{ext}, v$)

Figure 2. `FaMmod` enumeration algorithm.

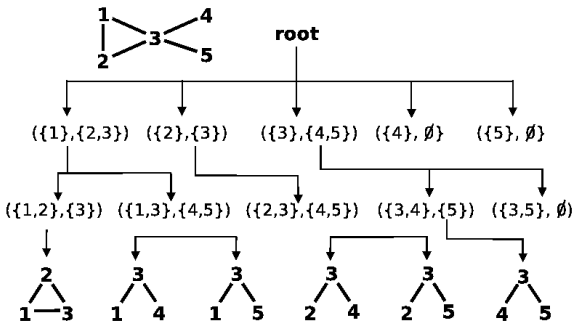

Figure 3. FanMod algorithm generates this recursive search tree for the given graph. Each internal node indicates the sets passed as parameters to the procedure `extendSubgraph`, respectively V_{Subg} and V_{Ext} (adapted from [35]).

Both algorithms also need to calculate isomorphism (line 6 of figure 1 and line 6 of figure 2) and both rely on finding a canonical form of the subgraph. While `MFinder` uses its own customized algorithm which is not well documented, `FanMod` uses a highly efficient third-party algorithm (*nauty* [23]).

The problem with doing the complete census is that the number of existing subgraphs grows exponentially as we increase the size of the network or the size of the subgraphs themselves. One way to cope with that growth is to sacrifice accuracy, using a probabilistic approximation algorithm: instead of fully enumerating all the subgraphs, we can sample a determined number of k-subgraphs on the original and on the random networks. We can then use their concentration to obtain an approximated z-score and therefore calculate an approximate significance, that will be more accurate as we increase the number of samples.

Figures 4 and 5 show how `MFinder` and `FanMod`, respectively, accomplish this.

Require: Graph G and positive integer k
Ensure: One sample k-subgraph
1: Pick random edge $(i, j) \in E(k)$
2: $S \leftarrow \{i, j\}$
3: **while** $|S| \neq k$ **do**
4: Pick random $v \in N(S) : v \notin S$
5: $S \leftarrow S \cup \{v\}$
6: Calculate probability to sample S
7: **return** S

Figure 4. `MFinder` sampling algorithm.

To sample one subgraph, `MFinder` chooses a random starting edge (line 1) and continues adding arbitrary new vertices that are on the neighbourhood of the partially constructed subgraph (lines 4 to 5) until the desired subgraph size is achieved (line 3). The problem is that this method is clearly biased because not all subgraphs have the same probability of being sampled [33]. To account for that, the sampling also calculates the probability P of this graph being chosen and then assigns the sample a weight of $W = P^{-1}$. With all this, sampling is a matter of calling the algorithm of figure 4 the desired number of times.

Require: Graph G, pos. integer k and set of prob. P_d
Ensure: Uniformly sampled k-subgraphs
1: lines 1 to 2 of algorithm in figure 2
2: with probability P_1
3: EXTENDSUBGRAPH($\{v\}, V_{Ext}, v$)
4: lines 4 to 11 of algorithm in figure 2
5: $V'_{Subg} \leftarrow V_{Subg} \cup \{w\}$
6: with probability $P_{|V'_{Subg}|}$
7: EXTENDSUBGRAPH(V'_{Subg}, V'_{Ext}, v)

Figure 5. `FanMod` sampling algorithm.

`FanMod` has a much more refined sampling algorithm. It uses the same base algorithm to enumerate all k-subgraphs but each recursive call is made only with a certain probability P_d, associated to the depth of the enumeration. Since each subgraph appears once and only once in the search subtree, all the subgraphs have the exact same probability of being called. More than that, we know that all subgraphs samples are different from each other, while in `MFinder` there is no such guarantee. On the other hand, we cannot exactly generate a fixed number of samples. Since we are dealing with probabilities, we can only choose the values of P_d in order to have an approximated number of sampled subgraphs: if we want to have a fraction $0 < q < 1$ of the subgraphs samples, then we must guarantee that $\prod P_d = q$. This still leaves the open question of how to choose the individual P_d values. The general rule is to have larger values for small d, since if we discard an entire search subtree near the root, many subgraphs will not have the possibility to be sampled. However, it must be noted that the larger these values are, then the more time the sampling will take, since we will have to branch into more subtrees.

`Grochow` takes a very different approach. While the other methods are network-centric in the sense that they discover motifs for the whole network, `Grochow` concentrates on counting the frequency of a specific isomorphic class. Figure 6 describes how it counts the number of occurrences of a single query graph.

Require: G, k and a query graph H
Ensure: All instances of H in G
1: $H_E \leftarrow$ EQUIVALENCEREPRESENTATIVES(H)
2: $C \leftarrow$ SYMMETRYBREAKINGCONDITIONS(H)
3: Sort $g \in V(G)$ by increasing degree and then by increasing neighbour degree sequence
4: **for all** $g \in V(G)$ **do**
5: **for all** $h \in H_E$ **do**
6: **if** SUPPORTS(g, h) **then**
7: $f \leftarrow$ partial map associating $f(h) = g$
8: ISOMORPHICEXTENSIONS(f, H, G, C)
9: Remove g from G.

10: **procedure** ISOMORPHICEXTENSIONS(f, H, G, C)
11: $D \leftarrow$ domain of f
12: **if** $D = H$ **then**
13: FOUNDOCCURRENCE(f)
14: $m \leftarrow$ most constrained neighbour of any $d \in D$
15: **for all** $n \in N(f(D))$, with $d \in D$ **do**
16: **if** $f(m) = n$ does not violate C **then**
17: **if** $\nexists d \in N(m) : n \notin N(f(d))$ **then**
18: **if** $\nexists d \notin N(m) : n \in N(f(d))$ **then**
19: $f' = f on D$ and $f'(m) = n$
20: ISOMORPHICEXTENSIONS(f', H, G, C)

Figure 6. `Grochow` single query subgraph algorithm.

The main idea is to progressively map the desired query subgraph H on the global graph G, instead of enumerating, and only after check for isomorphism. The algorithm starts by finding the equivalence classes of the query graph (line 1), in order to start the mapping in only one representative of each class, thus avoiding unnecessary and redundant searches. Then a series of symmetry conditions are found (line 2). The idea is to avoid symmetries by adding restrictions on the labelling of the vertices. This is done by going trough all equivalence classes and then imposing the condition that the label of one of its vertices is smaller than the minimum label of the others. Describing this procedure in more detail is beyond the scope of the this paper (more can be seen on [39]), but we should note that in terms of computational cost it consists on successive calculations of the automorphisms of the graph. Figure 7 gives an example of the conditions found in a graph with 6 vertices.

After doing all of this, the algorithm starts by trying to match every vertex g of the graph G into one of the vertices h representing each equivalence class of the query graph (lines 4 and 5). The vertices g are searched in order of their degree (line 3) to impose more constraints

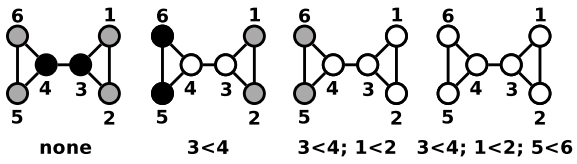

Figure 7. Finding symmetry conditions on a graph with 6 vertices. Vertices in white are fixed by any automorphism preserving the indicated conditions. Other vertices are coloured accordingly to their equivalence class (adapted from [39])

on the possible candidates. When one of the h vertices is a suitable candidate (line 6), that is, a possible match in what regards to its degree and neighbour degrees, the algorithm continues recursively mapping more vertices to see if the whole query graph can be mapped (line 8). In order to do this, the most constrained neighbour of g is tried (line 14), that is, the one with theoretically fewest possible candidates. If a candidate node does not violate the already calculated symmetry conditions and does not induce a contradictory neighbourhood (lines 16 to 18) then we add it to the mapping (line 19) and continue recursively (line 20) until the whole query graph is found (line 13). When we reach this stage, we have already addressed the isomorphism problem and we know that the subgraphs found correspond to the query subgraph.

In order to do an exhaustive census, `Grochow` uses McKay's `gtools` package [40] to generate all possible subgraphs of a determined size and then runs the single query search to determine their frequency. Other uses of this algorithm include a direct computation of whether a specific subgraph is a motif. This subgraph can be a larger than normal randomly sampled subgraph, which would have a prohibitive computational cost with the other methods.

One aspect remains to be explained, regarding the generation of similar random networks. The standard procedure is to use a Markov-Chain method. Starting with the original graph, a pair of edges $a \rightarrow b$, $c \rightarrow d$ is repeatedly swapped by $a \rightarrow d$, $b \rightarrow c$ in order to preserve vertex in and out degree. This is done until we achieve the desired degree of randomness. Another approach not so common is to directly generate the random network from scratch, which Milo et al. [9] adapted from Newman et al. [41]. Starting with an empty graph, we repeatedly add random connections with probabilities related to the number of connections that still must be made on each vertex. This continues until all vertices have the desired number of connections.

V. Comparison in an integrated framework

In order to compare the efficiency of the described algorithms we implemented them in a common framework using C++. We used the same underlying graph data structures and methods, the same isomorphism detection routines (*nauty*) and the same random subgraph generators. This ensures that the observed differences in the execution times of the implementations are due to the differences in the algorithms.

All tests were made on computers with an Intel Core 2 Duo T7500 processor at 2.2GHz and 2Gb of memory. All results obtained with `MFinder` and `FanMod` were double checked with the results obtained by using the publicly available tools with the same parameters to see if they produced equivalent results. In order to evaluate in different domains we used three different representative networks, summarized in Table II. `Circuit` and `Yeast` are directed graphs that were used as benchmarks in [9] for testing the original implementation of network motifs. `Social` is an artificial undirected graph used as a benchmark for community detection algorithms [42] and it was created with default parameters.

Table II
NETWORKS USED FOR EXPERIMENTAL TESTING OF THE ALGORITHMS.

Network	NrNodes	NrEdges	Short Description
`Circuit`	252	399	Electronic Circuit
`Yeast`	688	1079	Transcriptional network of *Saccharomyces cerevisiae*
`Social`	1000	15541	Social Network with heterogeneous communities

We run and compare the algorithms in full enumeration mode (no sampling). When computing network motifs, the number of random subgraphs generated has a dramatic impact on the time spent, since we have to do the census in all of the N random networks. What really matters is the time t to do a full census on one graph (the time spent on the generation of the random networks is negligible when compared to the time the census takes). The total time spent will be roughly equal to $(N+1) \times t$. Table III shows the time taken on average for one census of that size, calculated by applying an exhaustive census on the original graph. We do not show execution times greater then 4 hours.

Besides using the three main algorithms as they were imagined, we also consider two simple variations. If the graph is undirected, when applying `Grochow` we experiment to only generate and count all undirected subgraphs, which are much less than directed counterparts (for example, there are 1,530,843 different classes of isomorphic direct subgraphs with size 6 and only 112 when we consider just undirected connections). We call this approach `Grochow Undir`. We also experiment in applying `Grochow` only to the graphs that appear in the original subgraph previously found with the other enumeration method. We can only use this approach to count the occurrences in the random ensemble of networks after applying other method on the original network. Since there can be hundreds of random networks, the time for computing their respective census would dominate the time needed for the original network census. Note also that we would lose information like possible anti-motifs, but all possible motifs would be accounted for. We call this variation `Grochow Existent` and only show the time the census would take after we already know which subgraphs we need to count.

Table III
EXECUTION TIMES (IN SECONDS) OF AN EXHAUSTIVE CENSUS ON THE ORIGINAL GRAPH. k INDICATES THE MOTIF SIZE. MEM INDICATES THAT AS IT IS THE METHOD WOULD NEED TOO MUCH MEMORY. N/A MEANS THAT THE METHOD IS NOT AVAILABLE FOR THAT GRAPH.

	MFinder	FanMod	Grochow	Grochow Undir	Grochow Existent
Circuit					
k = 4	0.10	0.05	0.36	N/A	0.03
k = 5	0.36	0.14	15.79	N/A	0.15
k = 6	2.45	0.67	2519.3	N/A	0.81
k = 7	17.54	4.34	>4h	N/A	5.09
k = 8	130.66	27.49	>4h	N/A	33.83
k = 9	MEM	140.66	>4h	N/A	253.65
Yeast					
k = 4	2.42	0.77	2.25	N/A	0.55
k = 5	55.67	15.21	107.82	N/A	5.24
k = 6	MEM	260.38	>4h	N/A	70.91
k = 7	MEM	5023.91	>4h	N/A	1267.63
Social					
k = 4	40.44	11.22	16.98	1.48	1.45
k = 5	MEM	281.40	1122.74	33.30	30.50
k = 6	MEM	7951.47	>4h	888.75	872.30

As expected MFinder is slower than FanMod on all networks and sizes. More than that, it really uses large amounts of memory in order to hash all partial subgraphs found which quickly hinders its scalability to larger sizes. Even if we had implemented specialized compact notations for the subgraphs, thus handling one more unit of k, the growth of needed memory would still be exponential. Grochow is also very slow on its own, mostly due to the fact that it needs to consider all possible directed subgraphs of the motif size, thus meaning that a really huge number of different subgraphs that do not appear at all are considered. However, in the case of the only undirected network (Social), Grochow Undir manages to lower the number of different subgraphs and obtains results even better than FanMod, showcasing the efficiency of the method in terms of querying single subgraphs. When we know beforehand which subgraphs we must count, the behaviour is even better and Grochow Existent improves over FanMod on the time spent on the census. Note however that on small and not too dense networks (like Circuit) FanMod still has the upper-hand, but as we go to larger networks (Yeast and Social) Grochow Existent is several orders of magnitude faster.

VI. CONCLUSION

Complex networks are everywhere and as such are an important part of many e-Science data sets. There are many ways to analyze this type of data. One of them is to use network motifs, which we clearly defined, both informally and formally. We studied the state-of-art in strategies for finding network motifs and showcased them in a quick overview table that summarizes the main different and common properties that characterize the method. We also gave a detailed description of how these network motif discovery algorithms work, including pseudo-code describing their control flow.

We implemented the selected strategies on a common platform that allowed us to isolate external factors and compare execution times in order to understand which algorithm excels in what. When we need to enumerate and consider all subgraphs, FanMod is the strategy to follow, unless we have small motif sizes and undirected subgraphs, in which case Grochow can be the better option. If we are trying to find if a relatively small set of specific subgraphs is a motif, then Grochow is also the better option. Given this, an approach worth of further exploration is to initially use FanMod to enumerate all subgraphs of the original network, and then use Grochow to count the occurrences on the random ensemble of random networks. This would merge the positive aspects of both methods.

Adapting these algorithms for parallel execution is still an open research path that we intend to pursue. This is an area still in its early development stages but that could be of great benefit for e-Science applications.

ACKNOWLEDGMENTS

Pedro Ribeiro is funded by an FCT Research Grant (SFRH/BD/19753/2004) and Marcus Kaiser is funded by EPSRC (EP/E002331/1 and EP/G03950X/1).

REFERENCES

[1] R. Albert and A. L. Barabasi, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, no. 1, 2002.

[2] S. N. Dorogovtsev and J. F. F. Mendes, "Evolution of networks," *Advances in Physics*, vol. 51, no. 4, pp. 1079–1187, June 2002.

[3] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, 2003.

[4] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. Hwang, "Complex networks: Structure and dynamics," *Physics Reports*, vol. 424, no. 4-5, pp. 175–308, February 2006.

[5] M. Kaiser, "Brain architecture: a design for natural computation," *Philos Transact A Math Phys Eng Sci*, vol. 365, pp. 3033–3045, December 2007.

[6] E. Ziv, R. Koytcheff, M. Middendorf, and C. Wiggins, "Systematic identification of statistically significant network measures," *Physical Review E*, vol. 71, p. 016110, 2005.

[7] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM Computing Surveys*, vol. 38, p. 1, 2006.

[8] L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas, "Characterization of complex networks: A survey of measurements," *Advances In Phyiscs*, vol. 56, p. 167, 2007.

[9] 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, October 2002.

[10] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," *IEEE International Conference on Data Mining*, vol. 0, p. 313, 2001.

[11] J. Han and M. Kamber, *Data Mining, Second Edition: Concepts and Techniques*. Morgan Kaufmann, September 2006.

[12] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 1994, pp. 487–499.

[13] I. Albert and R. Albert, "Conserved network motifs allow protein-protein interaction prediction." *Bioinformatics*, vol. 20, no. 18, pp. 3346–3352, December 2004.

[14] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon, "Network motifs in the transcriptional regulation network of escherichia coli," *Nature Genetics*, vol. 31, no. 1, pp. 64–68, May 2002.

[15] R. Dobrin, Q. K. Beg, A. Barabasi, and Z. Oltvai, "Aggregation of topological motifs in the escherichia coli transcriptional regulatory network," *BMC Bioinformatics*, vol. 5, p. 10, 2004.

[16] M. Kondoh, "Building trophic modules into a persistent food web," *Proceedings of the National Academy of Sciences*, vol. 105, no. 43, pp. 16 631–16 635, 2008.

[17] O. Sporns and R. Kotter, "Motifs in brain networks," *PLoS Biology*, vol. 2, 2004.

[18] S. Itzkovitz, R. Levitt, N. Kashtan, R. Milo, M. Itzkovitz, and U. Alon, "Coarse-graining and self-dissimilarity of complex networks." *Physical Review E*, vol. 71, no. 1 Pt 2, January 2005.

[19] S. Valverde and R. V. Solé, "Network motifs in computational graphs: A case study in software architecture," *Physical Review E*, vol. 72, no. 2, 2005.

[20] A. Vazquez, R. Dobrin, D. Sergi, J. P. Eckmann, Z. N. Oltvai, and A. L. Barabasi, "The topological relationship between the large-scale attributes and local interaction patterns of complex networks," *Proceedings of the National Academy of Sciences*, vol. 101, p. 17945, 2004.

[21] J. Hallinan and P. Jackway, "Network motifs, feedback loops and the dynamics of genetic regulatory networks," in *Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2005.

[22] P. J. Ingram, M. P. Stumpf, and J. Stark, "Network motifs: structure does not determine function," *BMC Genomics*, vol. 7, p. 108, 2006.

[23] B. McKay, "Practical graph isomorphism," *Congressus Numerantium*, vol. 30, pp. 45–87, 1981.

[24] G. Ciriello and C. Guerra, "A review on models and algorithms for motif discovery in protein-protein interaction networks," *Briefings in Functional Genomics and Proteomics*, vol. 7, no. 2, pp. 147–156, 2008.

[25] D. S. Callaway, J. E. Hopcroft, J. M. Kleinberg, M. E. J. Newman, and S. H. Strogatz, "Are randomly grown graphs really random?" *Physical Review E*, vol. 64, no. 4, p. 041902, September 2001.

[26] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon, "Superfamilies of evolved and designed networks." *Science*, vol. 303, no. 5663, pp. 1538–1542, March 2004.

[27] V. Lacroix, C. G. Fernandes, and M. F. Sagot, "Motif search in graphs: Application to metabolic networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 3, no. 4, pp. 360–368, Oct.-Dec. 2006.

[28] M. Fellows, G. Fertin, D. Hermelin, and S. Vialette, "Sharp tractability borderlines for finding connected motifs in vertex-colored graphs," in *In Proc. 34th Int. Colloquium on Automata, Languages and Programming (ICALP)*, 2007.

[29] M. Middendorf, E. Ziv, and C. H. Wiggins, "Inferring network mechanisms: the drosophila melanogaster protein interaction network." *Proceedings of the National Academy of Sciences*, vol. 102, no. 9, pp. 3192–3197, March 2005.

[30] J. Chen, W. Hsu, M. L. Lee, and S.-K. Ng, "Nemofinder: dissecting genome-wide protein-protein interactions with meso-scale network motifs," in *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2006, pp. 106–115.

[31] L. Parida, "Discovering topological motifs using a compact notation," *Journal of Computational Biology*, vol. 14, no. 3, pp. 300–323, 2007.

[32] F. Schreiber and H. Schwobbermeyer, "Towards motif detection in networks: Frequency concepts and flexible search," in *Proc. of the Int. Workshop on Network Tools and Applications in Biology (NETTAB04)*, 2004, pp. 91–102.

[33] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon, "Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs," *Bioinformatics*, vol. 20, no. 11, pp. 1746–1758, 2004.

[34] C. Matias, S. Schbath, E. Birmel, J.-J. Daudin, and S. Robin, "Network motifs: mean and variance for the count," *REVSTAT*, vol. 4, pp. 31–35, 2006.

[35] S. Wernicke, "Efficient detection of network motifs," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 3, no. 4, pp. 347–359, 2006.

[36] F. Picard, J.-J. J. Daudin, M. Koskas, S. Schbath, and S. Robin, "Assessing the exceptionality of network motifs." *Journal of Computational Biology*, February 2008.

[37] S. Schbath, V. Lacroix, and M. Sagot, "Assessing the exceptionality of coloured motifs in networks," *EURASIP J. on Bioinformatics and Systems Biology*, 2008.

[38] S. Wernicke and F. Rasche, "Fanmod: a tool for fast network motif detection," *Bioinformatics*, vol. 22, no. 9, pp. 1152–1153, May 2006.

[39] J. Grochow and M. Kellis, "Network motif discovery using subgraph enumeration and symmetry-breaking," *Research in Computational Molecular Biology*, pp. 92–106, 2007.

[40] B. McKay, "Isomorph-free exhaustive generation," *Journal of Algorithms*, vol. 26, no. 2, pp. 306–324, 1998.

[41] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Physical Review E*, vol. 64, no. 2, p. 026118, July 2001.

[42] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, vol. 78, no. 4, 2008.