Discovering Motifs in Real-World Social Networks

Lotte Romijn[1], Breanndán Ó Nualláin[1,2], and Leen Torenvliet[2]

[1] Amsterdam University College
lotteromijn@student.auc.nl;bon@science.uva.nl
[2] ILLC, University of Amsterdam
L.Torenvliet@uva.nl

Abstract. We built a framework for analyzing the contents of large social networks, based on the approximate counting technique developed by Gonen and Shavitt. Our toolbox was used on data from a large forum—boards.ie—the most prominent community website in Ireland. For the purpose of this experiment, we were granted access to 10 years of forum data. This is the first time the approximate counting technique is tested on real-world, social network data.

Keywords: approximate counting, software development, social networks, big data

1 Introduction

Many real-world systems are complex networks and consist of a large number of highly connected interacting components. Examples are the World Wide Web, Internet, neural and social networks. Complex networks can be represented as graphs. These graphs contain characteristic patterns and substructures, such as cycles or triangles. Such patterns are called network motifs, subgraphs or templates. There are several algorithmic procedures to count or detect network motifs of size $O(\log n)$. Counting and detecting motifs is a method of identifying functional properties of a network. The frequency of certain motifs indicates how nodes behave in the network. The term "motif" was coined by Milo et al., who subsequently found motifs in biochemical, neurobiological, ecological and engineering networks [11]. A problem with counting patterns in graphs is that the general problem is known to be #P-hard[13], and therefore no efficient (*i.e.,* polynomial time) algorithms are known. Attempts to count motifs in networks must either be limited to networks of a modest size, so exponential algorithms finish in reasonable time, or give an approximate answer. As networks of interest are emerging that are large by nature, the latter approach seems the way to go.

Approximate counting of motifs has not been attempted on large networks (more than a few thousand nodes) [16]. In the experiment which this paper reports, we set up and tested a framework that is capable of analyzing large, real-world, social-media networks, by transforming them into graphs and approximately counting motifs in these networks.

This paper is organized as follows. We start with an overview of related work, then we discuss the approximate counting algorithm and parameter settings that lead to acceptable results. Then, we discuss the dataset on which the experiments were run. Section 6 contains the results of our analysis and our findings based on this, and the final section contains directions of further results and expansion of the algorithms.

2 Related Work

Networks with similar global topology can have varying local structures. In fact, local motifs are increasingly considered to be the small building blocks which are responsible for local functions in a network. Milo *et al.* [11] found network motifs in biochemical, neurobiological, ecological and engineering networks. An example of functional properties of motifs was illustrated by Becchetti *et al.* [5]. They showed that the local number of triangles in large-scale Web graphs is an indication of spamming activity. Przulj *et al.* [12] uses the term graphlet to denote a connected network with a small number of nodes.

The search for motifs in networks focuses on either induced or non-induced motifs [8]. Induced motifs have an additional restriction: an induced motif is a subset of vertices that contains all the edges between those vertices that are present in the original network. In general, searching non-induced motifs is more informative because a vertex in a network could have functions not associated with all of its adjacent edges [8]. Motif *detection* is equivalent to the subgraph isomorphism problem, a well-known problem in Graph Theory, which is NP-complete [7]. The exact solution can be found by enumerating all possible combinations of vertices that together form the size of the motif, and checking whether the edges present correspond to the edges in the subgraph. Ullmann described an exponential algorithm for subgraph isomorphism which takes polynomial time for a fixed choice of a subgraph [14]. Counting the number of motifs of a particular vertex builds upon the subgraph isomorphism problem. Counting motifs amounts to enumerating how many subgraphs can be found in which a particular vertex is included. Finding non-induced motifs grows rapidly in computation time with input size, and has not been attempted on large-scale, real-world networks [16]. Reducing this computation time represents a major research challenge.

Brute-force search for a particular motif requires the enumeration of all possibilities. For instance, finding all triangles in a network requires finding every pair of edges with the same vertex as one of their end vertices, and checking whether there is an edge connecting the other end vertices of these two edges. Without any form of approximation, the most efficient way to solve this problem uses matrix multiplication, which is of order $O(n^3)$ if a textbook method is used. Without matrix multiplication, a naive algorithm takes computation time of $O(n^5)$. For larger networks testing such naive algorithms is problematic.

Counting the number of a certain motif is $\#P$-hard. #P-problems are of the form "compute the value of a function $f(x)$," where $f(x)$ is the number

of possible solutions to the corresponding NP-decision problem [13]. They are at least as hard as the NP-problem, since solving the decision problem entails finding out if this number is nonzero. There are a few existing algorithms for counting and detecting non-induced motifs. These techniques go back to Larry Stockmeyer's (1983) theorem for approximate counting. He proved that for every #P-problem there is a randomized approximate algorithm that determines the count, using an NP-oracle. [13] This means that for a particular instance a of P and $\epsilon > 0$, the algorithm returns the count C with a high probability such that $(1 - \epsilon)P(a) \leq C \leq (1 + \epsilon)P(a)$. The randomized algorithm is in principle an (ϵ, δ)-approximation method.

3 Approximate Counting Algorithm

3.1 Color Coding

The approximate counting algorithm makes use of the color coding technique introduced by Alon *et al.* [2], used there to detect simple paths, cycles and bounded treewidth subgraphs. (See, *e.g.,* Bodlaender and Koster [6]). Recently, the color coding technique has been used to detect signaling pathways in PPI-networks [1].

The color coding technique is based on random assignments of colors to the vertices of an input graph. It can detect specific subgraphs efficiently by only considering specific color assignments, in time proportional to a polynomial function of the input $n = \|V\|$. If the assignment of colors is repeated sufficiently many times the method will find a specific occurrence of the motif of size $O(\log(n))$ with high probability. Multiple algorithms use elements of, or are entirely based on, the color coding technique [3, 1]. Arvind and Raman [3] used color coding for counting the number of subgraphs isomorphic to a bounded treewidth graph. Alon *et al.* [1] described a polynomial time algorithm for approximating the number of non-induced occurrences of trees and bounded treewidth subgraphs with $k \in O(\log n)$ vertices. In 2007, Hüffner, Wernicke and Zichner [10] presented various algorithmic improvements for color coding that lead to savings in time and memory consumption.

Other methods have been explored to approximate the number of motifs, such as the exploitation of subgraph symmetries by Grochow and Kellis [9]. It could happen that a subgraph H can be mapped to a given subset G of a graph multiple times. Eliminating these subgraph symmetries significantly decreases computation time. However, the running time of the algorithm still increases exponentially with the size of the motif.

Zhao *et al.* [16] have recently shown that using color coding in addition to parallel programming can find motifs in networks with millions of nodes. They have combined parallelization of color coding with stream based partitioning. Their "ParSE" algorithm was tested on large-scale, synthetically generated, social contact networks for urban regions.

In this paper, color coding is also employed to count motifs. Gonen and Shavitt's algorithm for counting simple paths will be explained in detail, together

with its implementation in the Python programming language and performance on the forum data set. By testing the algorithm on the forum data, Gonen and Shavitt's simple path algorithm is applied for the first time to a real-world, social network.

3.2 Simple Path Algorithm

Gonen and Shavitt's algorithm to find simple paths uses the color coding technique by Alon, Yuster and Zwick [2]. It approximates the number of paths of length $k-1$, where k is the number of colors in the color set. The input is the graph G, a vertex $v \in V$, the path length $k-1$, fault tolerance ϵ, and error probability δ.

$t = \log(\frac{1}{\delta})$; $s = \frac{4k^k}{\epsilon^2 k!}$;
for $j = 1$ to t **do**
  for $i = 1$ to s **do**
    Color each vertex of F independently and uniformly
    at random with one of the k colors
    for all $u \in V$ **do**
      $C_i(u, \emptyset) = 1$
    end for
    for all $l \in [k]$ **do**
      $C_i(v, l) = \begin{cases} 1 \text{ if } col(v) = l \\ 0 \text{ otherwise} \end{cases}$
    end for
    for all $S \subseteq [k]$ s.t. $\|S\| > 1$ **do**
      $C_i(v, S) = \sum_{u \in N(v)} C_i(u, S \backslash col(v))$
    end for
    $P_i(v, [k]) = \sum_{l=1}^{k} \sum_{(S_1, S_2) \in A_{l,v}} \sum_{u \in N(v)} C_i(v, S_1) C_i(u, S_2)$,
    where $A_{l,v} = \{(S_i, S_j) \mid S_i \subseteq [k], S_j \subseteq [k],$
      $S_i \cap S_j = \emptyset, \|S_i\| = l, \|S_j\| = k - l\}$
    Let $X_i^v = P_i(v, [k])$
  end for
  Let $Y_j^v = \frac{\sum_{i=1}^{s} X_i^v}{s}$
end for
Let Z^v be the median of $Y_1^v \ldots, Y_t^v$
Return $Z^v . k^k / k!$

This algorithm is an (ϵ, δ)-approximation for counting simple paths of length $k-1$ containing vertex v, "simple" meaning that there are no repeated vertices in the path. $P_i(v, S)$ is the number of colorful paths (*i.e.,* paths on which all nodes have a distinct color) containing v using colors in S at the ith coloring. $C_i(v, S)$ is the number of colorful paths for which one of the endpoints is v using colors in S at the ith coloring. The algorithm finds an approximation of the number of paths within $[(1-\epsilon)r, (1+\epsilon)r]$, where r is the actual number of paths in the graph, with a probability of at least $1 - 2\delta$.

The estimator used in this algorithm is also called "median of means" and it can be shown, using Chebyshev's inequality and Chernoff bounds, that the expected value of the number of colorful paths (*i.e.,* the number of paths times $k!/k^k$) can be approximated arbitrarily closely using a limited number of iterations.

When experimenting with the algorithm on known graphs, two problems with the original pseudocode of Gonen and Shavitt became apparent. First of all, when v is a node on the path, the colorful paths containing v are counted "in both directions", in other words twice. Second, when v is an endpoint, one of the sets in the partition is the empty set and therefore $C_i(u, \{\}) = 1$ for all neighbors of v. To get results that are both theoretically correct and experimentally acceptable, we had to adapt the value of P_i as follows.

$$P_i(v, [k]) = C_i(v, k) + \frac{1}{2} \sum_{l=2}^{k-1} \sum_{(S_1.S_2) \in A_{l,v}} \sum_{u \in N(v)} C_i(v, S_1).C_i(u, S_2)$$

This adaptation in computing P_i influences the complexity of computing P_i only by a constant, leaving the complexity of the algorithm $O((2e)^k \|E\|. \log(\frac{1}{\delta})/\epsilon^2)$, as in the Gonen-Shavitt case.

4 About the Implementation

To implement and test the simple path algorithm in Python, we made use of the packages NetworkX (http://networkx.github.io/) for generating graphs of vertices and edges to which weights and labels can be associated, and Numpy (http://www.numpy.org/) that provides a library of mathematical functions for performing computations on large arrays.

To achieve a better performance we have made use an efficient bitwise representation of color sets. The sets of colors required are all "small" sets with cardinality certainly less than 32. Such sets can efficiently be represented as 32-bit integers using a bitwise representation where, for example, the set $\{6, 4, 3, 1\}$ is represented by the integer $2^6 + 2^4 + 2^3 + 2^1 = 90$. The counterparts to the necessary set operations can then be efficiently implemented as combinations of logical bit operations.

To test the implementation, we generated complete and random graphs with less than 20 nodes in which the number of paths of a given length can be computed exactly using built-in NexworkX functions and compared the results with the results of the color coding algorithm. An example of the measured results is in Figure 1

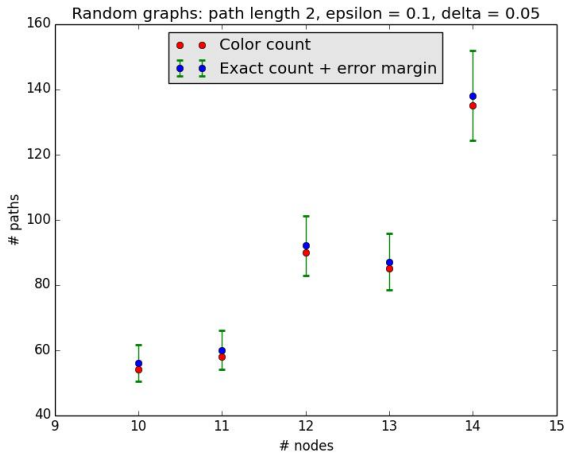

Fig. 1. Simple path counts of length 2 for random graphs with 2/3 possible edges

For obvious reasons, exact counting cannot be done on large graphs. However, both calculations and tests suggested that the number of iterations needed to achieve acceptable accuracy in the color coding case could be reduced significantly, resulting in a significant reduction in computation time (See Figure 2).

t	s	Number of iterations	Mean	Standard deviation
5	20	100	107.166375	6.722599292
5	90	450	107.85075	3.584042681
5	180	900	108.01275	2.277940394
5	270	1350	107.99725	1.998800875
5	360	1800	108.150938	1.535474752
5	540	2700	108.042375	1.245173181
5	720	3600	108.0345	1.199928865
5	900	4500	107.92755	1.023959715
5	1080	5400	108.06	0.980441546
5	1260	6300	107.981786	0.87689016
5	1440	7200	108.090234	0.878974341
5	1620	8100	108.073917	0.747840553
5	1800	9000	107.9613	0.64225233

Fig. 2. Accuracy of the algorithm for varying s and t

A series of further tests led to fine tuning of the parameters of the algorithm for the analysis of the real data set.

5 About The Data Set

The real-world, complex networks for which the numbers of simple paths have been counted are generated from an Irish forum data set. This data set was put online in 2008 for the "boards.ie SIOC Data Competition." (SIOC stands for Semantically-Interlinked Online Communities Project). The complete data

set contains ten years of Irish online life from Irelands largest community website "boards.ie" over the years 1998-2007. Since the foundation of the website in 1998, over 36 million posts have been made and the current posting rate is around 17,000 a day (retrieved from http://www.boards.ie/content/about-us). The data set is a large collection of RDF-files (Resource Description Framework), in which each file contains a post in a thread on a forum. The RDF-files have a tree-like structure, corresponding to the way in which the board's website contains various forums, each forum contains multiple threads and each thread contains board posts that are chronological replies to each other. In this project, useful information from the data set is extracted by parsing the RDF-files. Such information involves the topic of the post (title), the username of the person who posted, his FOAF-person profile, and the thread which contains the posts. The Python package that is used for this purpose is `rdflib` (https://github.com/RDFLib/rdflib/), which can query and extract certain elements from an RDF-file. From there, we generate graphs that represent the structure of the data set by using the NetworkX Python package. For the main analysis of this project, graphs are constructed in which nodes represent users with accounts on the `boards` website, and edges the connections between users if they posted in the same threads. For an initial analysis, the data of the years 1998-2000 were used. The distribution of simple paths in these graphs were compared to artificial data, such as randomly generated graphs, preferential attachment graphs, and small-world graphs. While testing the algorithm on the data sets, the algorithm was revised and further optimized. The results of the motif counts were analyzed and compared, and further analysis of these substructures was used to yield conclusions about this forum data set.

6 Results and Conclusions

6.1 Analysis

Having tested, corrected and fine-tuned our software, it was time to run tests on the SIOC data and compare characteristics of these data with artificially created networks of different sorts. Due to space limitation, only a small fraction of the results obtained can be presented here.

First we give a logarithmic representation of the means of all nodes per path length. We obtain a linear plot, which indicates the mean of the counts grows exponentially in path length. The plot shows a standard deviation close to the means, indicating a large tail of the distribution. Similar results were obtained for 1999 and 2000.

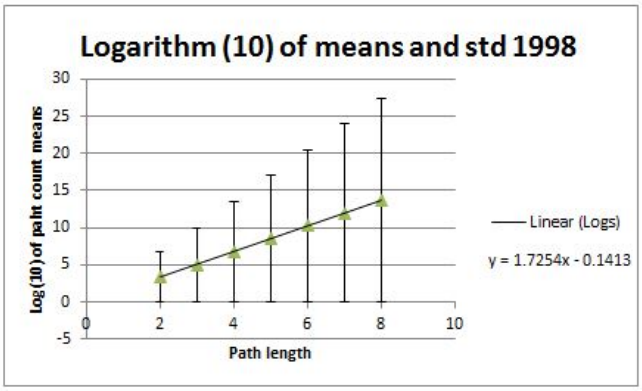

Fig. 3. Analysis of 1998 path counts. Logarithm of means and standard deviations

Figure 4 shows the relationship between the number of paths of length 3 vs length 2 and 4 vs 7 of the same node for the 1998 data set. The relationship is linear and was found for each pair of length up till length 9.

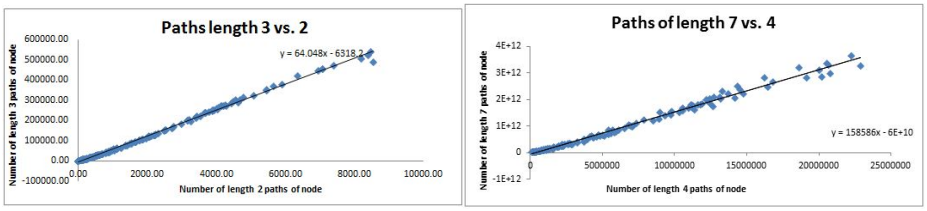

Fig. 4. Relationship between path lengths in 1998 data

The distribution of paths in the `boards` forum data was compared with random, preferential attachment and small-world graphs. For the graphs of 1998, 1999 and 2000 the distribution of the number of paths of a specific length is plotted. In Figures 5 and 6 the x-axis represents the number of paths of specific length that were counted of which a given node is a member. The y-axis shows the relative number of occurrences of this number of paths for all nodes in the graph in Figure 5. In Figure 6, the y-axis shows the relative occurrence of this number of paths for 150 graphs generated by graph models in NetworkX, in which each time the number of paths for a random node was determined. (These graphs have the same number of nodes and edges as the 1998 data.) Based on the minimum and maximum number of paths, the data is segmented into 50 bins. The data was fitted to all valid parametric probability distributions in Matlab, using the function `Allfitdist` (retrieved from http://www.mathworks.nl/matlabcentral/fileexchange/). The four best fitting probability density functions are displayed in the plot. Below the results are shown for the `boards` graph of 1998. Since the relationship between different path lengths is linear (Figure 4), the main analysis below is performed with path length 3. The graphs in Figure 6 are, respectively, generated randomly, then scale-free according to the preferential attachment model by Barabási and Albert [4] and then twice according to the small-world model of Watts and Stro-

gatz [15] with rewiring probabilities 0.5 and 0.03. Other probabilities were also tested and showed similar results.

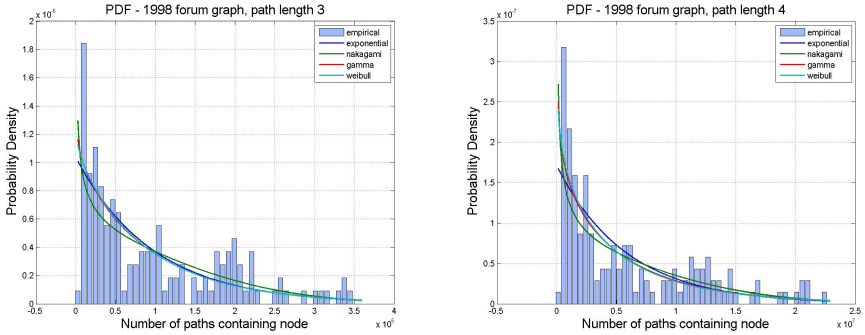

Fig. 5. Paths of length 3 and 4 in the 1998 data and fitted probability density functions

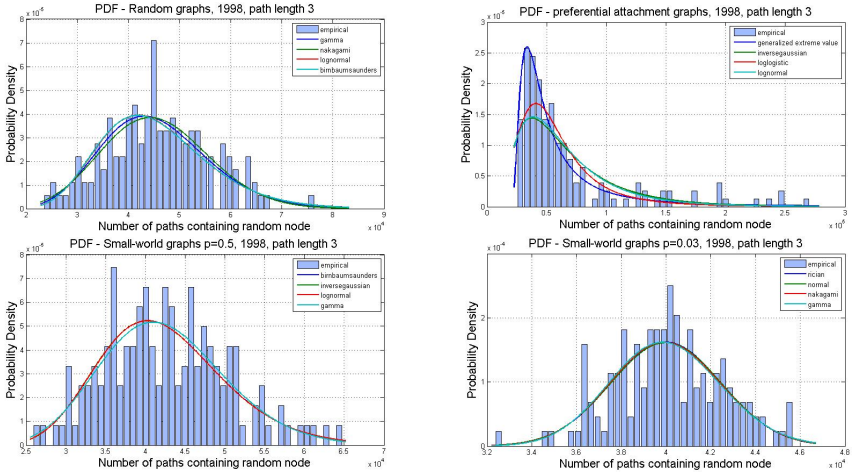

Fig. 6. Path counts in 150 randomly generated graphs according to different models

A final plot compares path counts to degree.

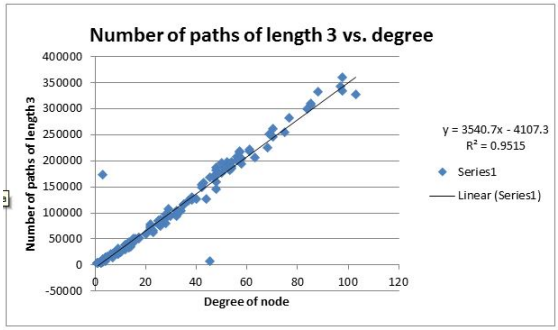

Fig. 7. paths of length 3 vs degree

6.2 Conclusions

Concerning the algorithms The most important conclusion to draw from the experiments is that the algorithms of Gonen & Shavitt and our implementation of them indeed work. Except from a minor glitch in the pseudocode, mentioned above, and the fact that in practice much fewer iterations of the randomized algorithm were necessary than were required by the theory everything went more or less smoothly. This again supports the practice of testing implementations against data for which the count is known. Nonetheless, there were some performance issues which support the consideration to port the implementation to a lower-level language, such as C, before making the tools more widely available. Meanwhile however, extensions as discussed in the next section, and tests on even larger networks seem in order.

Concerning the experiments The histograms of the number of paths (Figure 5) show that the distribution of the number of paths is broader than for random graphs with the same number of nodes and edges. The peak for path length 3 of the 1998 data is at 1.5×10^4 paths, while for random graphs with the same number of nodes the peak is at 4.5×10^4 (≈ 3 times larger than the SIOC graph). On the other hand, the tails of path distributions of the SIOC graphs are much larger for all path lengths. Again, for path length 3 the largest number of paths of the SIOC data is around 3.5×10^5, while for random graphs it is around 2.3×10^4 (≈ 15 times smaller than the SIOC graph). Similar results were obtained for larger path lengths. This suggests that there are a few members extremely active on the `boards` website and are connected with other nodes through many paths. For instance, when a node is a member of a large number of paths of length 3 (4 nodes), it suggests that the node itself posts prolifically and has many threads in common with other users (prolific poster) and it could have posted in a thread in which an extremely active user has posted (extremely active neighbor). However, from Figure 7 it can be concluded that counting the number of paths a node is a member of and its degree are highly correlated. This suggests that prolific users are generally in more paths. The two outliers in this correlation could be explained by considering neighbors; there is one node with low degree (degree 3) and a high number of paths (173000). This means that it is probably connected to a very prolific user, or even to two. There is one node with a higher degree (degree 45) and a low number of paths (8022).

Comparing the distribution of counts against those of artificially generated graphs revealed that the distribution of the `boards` data graph is similar to that of preferential attachment graphs with the same amount of nodes and edges. The small-world model [15] with varying rewiring probabilities does not show similar path-count distributions as the SIOC data. In the simple path counts it is seen that the distribution of the counts looks similar to preferential attachment networks. Each new individual on the `boards` website posting in a thread has a higher probability of posting in a thread in which a prolific poster has posted earlier. Hence, it has a higher probability of forming an edge with that person.

This mechanism is essentially how a preferential attachment network is dynamically generated. In a preferential attachment graph, every new node connects to already existing nodes and has a higher probability of connecting to "popular" nodes in the graph. However, the preferential attachment model differs from the real-world SIOC graph in several aspects. Firstly, in a preferential attachment graph, a new node has a fixed number of starting edges. Each new node connects to existing nodes with the same amount of edges, while in the SIOC data a new individual on a forum could actively engage in many threads, post new questions and/or be an observer. Also, the preferential attachment model does not take into account that new nodes can make new threads. In addition, the degree distribution of a scale-free network (that can be generated by the preferential attachment model) generally follows a power-law distribution. This was not observed for the SIOC data. In the SIOC data, the number of nodes with higher degree decreases less rapidly. However, it is likely that the preferential attachment process causes the similarity in distribution of path counts between the preferential attachment and the SIOC data graphs.

7 Further Research

In this project we created a toolbox for analyzing real-world, social networks of considerable size. Our approach was concentrated on simple paths, but the algorithms can easily be extended to researching other motifs and graph properties. The toolbox we created can be expanded in several directions:

- Algorithmically: the algorithms and code developed so far can still be optimized in several ways. For instance, the approach taken allows for numerous variations of parallelization. Not only can different parts of the graph be explored simultaneously, as was done on artifical data in [16] because of the limited size of the structures searched for, but also the iterations sketched above can be parallelized, with which a speedup is expected or, equivalently, the ability to tackle even larger graphs.
- Semantically: Though the motifs considered so far are interesting, they do not represent the only properties one would extract from social networks. For instance a question of interest is: which users form cliques interested in the same subset of subjects? Is the opinion of users about subjects related to their geographic location (IP-number)? Which (opposing) coalitions are formed? Simple paths are just a start. Yet, the triangles mentioned in Section 1 are simple paths of length two, of which the endpoints are neighbors. Circles are like paths of arbitrary length, and cliques are circles in which all points are neighbors.
- Dynamically: So far only static graphs have been considered. However, we have data spanning a long period of time. We would make the toolbox adequate to deal with development of structures over time. For example time decay could be added to the preferential attachment model to account for recency of threads.

We will continue to pursue this research.

Acknowledgements

We thank Tom Murphy of `boards.ie` and John G. Breslin, University College Galway, for providing access to the data from the SIOC project and are grateful to the anonymous referees for useful comments.

References

1. Alon, N., Dao, P., Hajirasouliha, I., Hormozdiari, F., Sahinalp, S.C.: Biomolecular network motif counting and discovery by color coding. Bioinformatics 24(13), 241–249 (2008)
2. Alon, N., Yuster, R., Zwick, U.: Color-coding. Journal of the ACM 42(4), 844–856 (1995)
3. Arvind, V., Raman, V.: Approximation algorithms for some parameterized counting problems. In: Bose, P., Morin, P. (eds.) ISAAC 2002. Lecture Notes in Computer Science, vol. 2518, pp. 453–464. Springer-Verlag, Heidelberg (2002)
4. Barabási, A.L.: Scale-free networks: A decade and beyond. Science 325(5935), 412–413 (2009)
5. Becchetti, L., Boldi, P., Castillio, C., Gionis, A.: Efficcient semi-streaming algorithms for local triangle counting in massive graphs. In: Proceedings of the 14th ACM SIGKDD International conference on Knowledge Discovery and Data Mining. pp. 16–24 (2008)
6. Bodlaender, H., Koster, A.: Combinatorial optimization on graphs of bounded treewidth. The Computer Journal 51(3), 255–269 (2008), doi:10.1093/comjnl/bxm037
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York (1979)
8. Gonen, M., Shavitt, Y.: Approximating the number of network motifs. In: Avrachenko, K., Donato, D., Litvak, N. (eds.) WAW 2009. LNCS, vol. 4453, pp. 13–24. Springer-Verlag, Heidelberg (2009)
9. Grochow, J., Kellis, M.: Network motif discovery using subgraph enumeration and symmetry-breaking. In: RECOMB 2007. LNCS, vol. 4453, pp. 92–106. Springer-Verlag, Berlin-Heidelberg (2007)
10. Hüffner, F., Wernicke, S., Zickner, T.: Algorithm engineering for color-coding with applications to signaling pathway detection. Algorithmica 52, 114–132 (2007)
11. Milo, R., Shen-Orr, S., Itzkovitz, S., Cashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. Science 298(5594), 824–827 (2002)
12. N.Przulj, Corneil, D., Jurisica, I.: Modelling interactome: Scale-free or geometric. Bioinformatics 150, 216–231 (2005)
13. Stockmeyer, L.: The complexity of approximate counting. In: Proceedings of the 15th annual ACM symposium on Theory of Computing. pp. 118–126 (1983)
14. Ullmann, J.: An algorithm for subgraph isomorphism. Journal of the Association for Computing Machinery 22(1), 31–42 (1976), doi:10.1145/321921.321925
15. Watts, D., Strogatz, S.: Collective dynamics of "small-world" networks. Nature 393, 440–442 (1998), doi:10.1038/30918
16. Zhao, Z., Khan, M., Kumar, V., Marathe, M.: Subgraph enumeration in large social contact networks using parallel color coding and streaming. In: Proceedings of the 39th conference on parallel processing. pp. 594–603 (2010)