Mining large networks with subgraph counting[*]

Ilaria Bordino
Sapienza Università di Roma
Rome, Italy
bordino@dis.uniroma1.it

Debora Donato
Yahoo! Research
Barcelona, Spain
debora@yahoo-inc.com

Aristides Gionis
Yahoo! Research
Barcelona, Spain
gionis@yahoo-inc.com

Stefano Leonardi
Sapienza Università di Roma
Rome, Italy
Stefano.Leonardi@dis.uniroma1.it

Abstract

The problem of mining frequent patterns in networks has many applications, including analysis of complex networks, clustering of graphs, finding communities in social networks, and indexing of graphical and biological databases. Despite this wealth of applications, the current state of the art lacks algorithmic tools for counting the number of subgraphs contained in a large network.

In this paper we develop data-stream algorithms that approximate the number of all subgraphs of three and four vertices in directed and undirected networks. We use the frequency of occurrence of all subgraphs to prove their significance in order to characterize different kinds of networks: we achieve very good precision in clustering networks with similar structure. The significance of our method is supported by the fact that such high precision cannot be achieved when performing clustering based on simpler topological properties, such as degree, assortativity, and eigenvector distributions. We have also tested our techniques using swap randomization.

1 Introduction

Graphs are ubiquitous data representations that are used to model complex relations in a wide variety of applications, including biochemistry, neurobiology, ecology, social sciences, and information systems. One of the most basic tools for analysing graph structures and revealing the properties of the underlying data is finding frequent patterns in graphs. This task has numerous applications, like network characterization [24, 25], modeling complex networks[22], detecting anomalies [8], and indexing graph databases [34].

Despite the fact that many interesting graph datasets have really large scale (a prominent example is the Web graph), a lot of the existing work has been restricted to the analysis of small networks [25]. Counting subgraphs on large datasets is a challenging computational task, due to the explosion of the number of candidate subgraphs involved. A natural approach that can be used to perform computations on huge data sets is the one based on the adoption of the data-stream model [26]. The most basic subgraph-counting problem, counting the number of triangles in an undirected graph, has indeed been studied in this model [6, 10, 20]. However, to the best of our knowledge, the problem of designing efficient data stream algorithms for counting other patterns in large-scale graphs has not been studied before.

Our contributions in this paper are summarized as follows:

We extend the techniques of Buriol et al. [10] for counting triangles in data streams, and we develop data-stream algorithms that approximate the number of all graph minors of three and four vertices in directed and undirected graphs.

We demonstrate the practical applicability of our algorithms by developing an optimized implementation and evaluating their performance on real networks of size up to one billion edges.

We perform extensive experiments that demonstrate the relevance and usefulness of our graph-minor counting algorithm for the task of recognizing families of networks.

We show that the precision obtained by clustering algorithms that use as features the distributions of the graph minors in the networks cannot be achieved with simpler topological features. We also assess the statistical significance of our method by swap randomization [15].

The rest of this paper is organized as follows. Section 2 describes previous work on mining frequent subgraphs and on data stream computation. In section 3 we present our

[*]This work was partially supported by the EU within the 6th Framework Programme under contract 001907 "Dynamically Evolving, Large Scale Information Systems" (DELIS)

general algorithm for counting graph minors. In Section 4 we present the experimental results on the quality of the approximations and our network clustering algorithm. Finally, Section 5 is a short conclusion.

2 Related work

The problem of discovering frequent subgraphs has been studied extensively in the area of data mining [13, 18, 23, 33]. The algorithmic techniques for this problem are mostly based on the a-priori principle [5]. A key difference between our paper and the above line of research is that we focus on counting the occurrences of subgraphs in one single large graph.

Our algorithm for clustering networks is based on the distribution of minors and it is, to a large extent, inspired by the work of Milo et al. [24, 25], who search for those minors whose frequency is significantly higher than the frequency that one would expect in random networks with the same degree distribution. A similar idea of testing data mining results against random networks with a given degree distribution was also proposed in [15]. Milo et al. [21] have also proposed an algorithm that uses a randomly sampled set of subgraphs to estimate subgraphs concentrations and to detect network motifs. However, this algorithm has been shown to have a bias for sampling certain subgraphs more often than others [32].

The problem of finding graphlets of three and four nodes in protein interaction networks was studied recently in [30]. Differently from our algorithms, the heuristics proposed in [30] do not have any performance guarantee on the processing time and storage requirements. Moreover, their algorithm is not scalable for large graphs.

The large body of work on data stream algorithms [17] contrasts with a lack of efficient solutions for many natural graph problems. Previous to this work, algorithms for counting triangles have been presented in [6, 20, 10, 11]. In this paper we extend the techniques of counting triangles to counting all minors of size 3 and 4 for directed and undirected graphs.

3 Algorithms for counting graph minors

Let $G = (V, E)$ be a graph, which can be either directed or undirected. Our basic model for the representation of G is the "incidence" stream model, in which we assume that all edges incident to the same vertex appear subsequently in the stream. In the case of undirected graphs, every edge appears twice—in the incidence list of both incident nodes. The ordering $v_1, \ldots, v_n$ of the vertices can be arbitrary. We denote by d_i the number of vertices incident to vertex v_i.

We present a general *three-pass* sampling algorithm for counting the number of occurrences of a specific minor M of c vertices in the graph G. We denote by $M = (X_M, Y_M)$

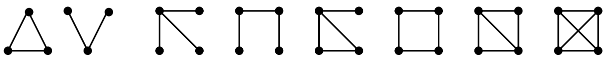

Figure 1. All minors of size 3 and 4 for undirected graphs

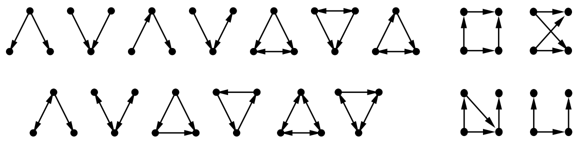

Figure 2. Minors of size 3 and 4 for directed graphs

a prototype minor of G, which is a simply connected subgraph of G having no multiple edges and no vertex loops. Here $X_M = \{\bar{x}_1, \ldots, \bar{x}_c\}$ and Y_M are the vertices and edges of M, respectively. The minors of size 3 and 4 for undirected and directed graphs are shown in Figures 1 and 2. In the figures all the possible minors are shown, except for the case of directed minors of four nodes, where only four minors are shown out of 199 possible ones. We now describe in detail the algorithm for counting minors. First we fix a minor $M = (X_M, Y_M)$, whose number of occurrences we want to count in the graph G. The algorithm uses two basic concepts: (i) the concept of a *prototype subgraph* S_M for M, and (ii) the concept of the *sample space* $\mathcal{S}$ for M with respect to S_M.

The prototype subgraph $S_M = (X_M, Y'_M)$, $Y'_M \subseteq Y_M$ is simply a subgraph of M defined on the same set X_M of c vertices as M. For example, if M is the minor representing an undirected triangle, then S_M can be a path of length 2. The sample space $\mathcal{S}$ is defined to be the set of all distinct subgraphs S of G that are *isomorphic* to S_M.

At a very intuitive level, $\mathcal{S}$ defines the set of "candidate places" in which M can potentially appear in G. The algorithm samples such candidate places from $\mathcal{S}$, checks if M actually appears in those places, and then uses the count of the occurrences of M in the sample to estimate the number of occurrences of M in G.

More formally, we define $\mathcal{S}$ to be the set of subgraphs $S = (X, Y)$ of G for which there is a bijection $f : X \to X_M$, such that for each $x_1, x_2 \in X$ it is $(x_1, x_2) \in Y$ if and only if $(f(x_1), f(x_2)) \in Y'_M$. Given a subgraph $S = (X, Y)$ in $\mathcal{S}$, we define $\bar{Y}(X)$ to be the edges that are needed to extend S to an occurrence of M, that is, $\bar{Y}(X) = \{(f^{-1}(\bar{x}_1), f^{-1}(\bar{x}_2)) : (\bar{x}_1, \bar{x}_2) \in Y_M / Y'_M\}$. Finally, we denote by $|\mathcal{S}|$ the size of the sample space $\mathcal{S}$.

The general three-pass algorithm is the following:

```
SAMPLEMINOR
  1st Pass:
    Compute the size $|\mathcal{S}|$ of the sample space.
  2nd Pass:
    Uniformly choose a member $S = (X, Y)$ of
    the sample space.
3rd Pass:
    Run the following test:
    if all edges in $\bar{Y}(X)$ are in the graph
    then $\beta = 1$
    else $\beta = 0$
return $\beta$
```

The accuracy and the performance of the algorithms we propose rely crucially on the structure of the sample space and, consequently, on the choice of the prototype subgraph S_M. We observe that in order to provide a uniform sampling of all occurrences of M, we need to ensure that every single occurrence of M in the graph can be detected by extending the same number of distinct subgraphs in the sample space. This number, which we denote by n_M, is defined to be the number of isomorphic mappings of the subgraph S_M to the minor M. The specified requirement is clearly guaranteed by our method since we check if all the edges needed to extend a sampled subgraph to an occurrence of the minor M exist in the graph.

We make a few remarks on the SAMPLEMINOR algorithm. The first pass requires to design a sample space whose size can be easily determined in a streaming pass over the graph. Thus, the choice of the prototype subgraph S_M should be made in a way that ensures that the size of the sample space depends only on simple parameters, like the number of vertices, the number of edges, and the degree of every vertex. The second pass of the algorithm requires to list all samples of the sample space in linear order, and to efficiently identify the sample corresponding to some position in the order. This task is easy if, say, the sample space is formed by the set of all edges of the graph. In general, more complex enumeration schemes might be needed.

Now, let us denote by T_M the number of occurrences of minor M in the graph G. We recall that the objective of the algorithm is to provide a good estimation of T_M.

Lemma 3.1. *The algorithm* SAMPLEMINOR *outputs a value β, which has expected value*

$$\mathbf{E}[\beta] = \frac{n_M \cdot T_M}{|\mathcal{S}|}$$

Proof. The algorithm chooses a random element $S(X)$ of the sample space defined on a subset X of c vertices. Each of the T_M minors can be obtained in n_M different ways, i.e., starting from n_M different samples. Since each choice of a $S(X)$ has the same probability, the probability of choosing a sample that can be extended to a minor is $\frac{n_M \cdot T_M}{|\mathcal{S}|}$. □

A single run of the SAMPLEMINOR algorithm returns a binary value. To obtain an estimation of the expectation of the parameter β, we perform multiple runs of the SAMPLEMINOR algorithm. In particular, we start

$$s \geq \frac{3}{\epsilon^2} \cdot \frac{|\mathcal{S}|}{n_M \cdot T_M} \cdot \ln(\frac{2}{\delta}) \tag{1}$$

parallel instances of SAMPLEMINOR and return the value

$$\widetilde{T_M} := \left(\frac{1}{s} \cdot \sum_{i=1}^{s} \beta_i\right) \cdot \frac{|\mathcal{S}|}{n_M}$$

as an estimate of T_M, the number of occurrences of the minor M in the graph. For the quality of approximation provided by $\widetilde{T_M}$ we can show the following.

Lemma 3.2. *With probability $1 - \delta$ the following statement holds:*

$$(1 - \epsilon) \cdot T_M < \widetilde{T_M} < (1 + \epsilon) \cdot T_M.$$

Proof. (Sketch) We use the Chernoff bounds

$$\Pr[\frac{1}{s} \sum_{i=1}^{s} \beta_i \geq (1 + \epsilon)\mathbf{E}[\beta]] < e^{-\epsilon^2 \cdot \mathbf{E}[\beta] \cdot s/3}$$

and

$$\Pr[\frac{1}{s} \sum_{i=1}^{s} \beta_i \leq (1 - \epsilon)\mathbf{E}[\beta]] < e^{-\epsilon^2 \cdot \mathbf{E}[\beta] \cdot s/2},$$

and the definition of s from Equation (1) for the number of repetitions of the algorithm. □

We complete this section with the analysis of the running time of the algorithm. The time complexity of the first two passes of the algorithm cannot be stated in general, since they depend on the specific sampling strategy. However, if there is a constant-time method to access the i-th element of the sample space, the uniform selection of a sample can be implemented in constant time per edge of the graph by reservoir sampling [31], as we will discuss in the next section. For the third pass of the algorithm, if we implement the different instances of the SAMPLEMINOR algorithm independently of each other, we require $O(\frac{1}{\epsilon^2} \cdot \log(\frac{1}{\delta}) \cdot (\frac{|\mathcal{S}|}{n_M T_M}))$ time to process each edge. However, as we will also describe in the next section, the cost of checking for M in all instances of S_M can be reduced to expected constant time per edge of the graph G via hashing. We therefore conclude with the following theorem.

Theorem 1. *There is a three-pass streaming algorithm to count the number of minors in incidence streams up to a multiplicative error of $1 \pm \epsilon$, with probability at least $1 - \delta$, which needs $O(s)$ memory cells and amortized expected update time $O(1 + s \cdot \frac{|V|}{|E|})$, where*

$$s \geq \frac{3}{\epsilon^2} \cdot \frac{|\mathcal{S}|}{n_M \cdot T_M} \cdot \ln(\frac{2}{\delta}).$$

Table 1. The datasets used for experimental evaluation.

Graph class	Type	# Instances	Max $\lvert V\rvert$ (thousands)	Max $\lvert E\rvert$ (thousands)	Graph class	Type	# Instances	Max $\lvert V\rvert$ (thousands)	Max $\lvert E\rvert$ (thousands)
Synthetic [22, 7]	un/directed	39	850	30	Word adjacency [24]	directed	4	30	30
Wikipedia [9]	un/directed	7	350	5 000	Author Collaboration [7, 28, 27]	undirected	5	400	7 000
Webgraphs [3]	un/directed	5	40 000	900 000	Autonomous Systems (AS) [1]	undirected	12	12	43
Cellular [19]	directed	43	3	7	Protein Interaction [2]	undirected	3	4	16
Citation [14]	directed	3	4 000	16 000	US Road [4]	undirected	12	24 000	58 000
Food webs [1]	directed	6	0.3	0.3					

4 Experimental evaluation

We provide an optimized implementation of our algorithm for counting all the subgraphs of three and four nodes. Figure 3 shows the prototype subgraphs that we have used for sampling. In the implementation, we merge the first two

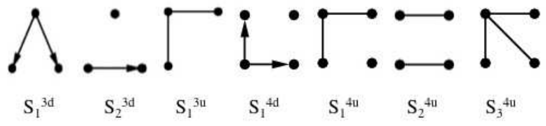

Figure 3. Prototype sampling subgraphs

passes of our algorithm by applying *reservoir sampling* [31] to choose elements at random for a sample set whose size is not known in advance. We also implement efficiently the third pass using a uniform hash function that requires linear space, as proposed in [29]. We evaluate our algorithms on an extensive collection of real and synthetic datasets, which we summarize in Table 1. All the datasets are made available.

4.1 Quality of approximation

We have performed extensive experiments to evaluate the accuracy of the counts produced by our methods. For the sake of concreteness and clarity of presentation, we present approximation results on counting the occurrences of one particular directed 3-node minor, which we call M11 and it is shown in Table 2. We have verified that the quality of approximation is similar for many other minors.

Table 2 reports the results on the quality of approximation of counting the occurrences of M11 in the 5 largest Webgraphs in our dataset. We run the algorithm three times using $10K$, $100K$ and $1M$ samples. For each run, Table 2 reports: (i) the estimated number $\widetilde{N}$ of occurrences of M11; (ii) the quality $Qlt(\%)$ of the result, expressed in terms of percentage deviation from the exact count: a positive value indicates an overestimation and a negative value indicates an underestimation; (iii) the running time in seconds.

Our algorithm obtains an approximation as good as 5% even when using only 10,000 samples. For the two largest datasets, uk-2002 and uk-2005, which have 200 and 940 million edges respectively, we do not indicate the quality of the approximation since we are not able to compute the exact value.

Alternative sampling strategies for undirected 4-node minors: For each undirected minor of 4 nodes we implement two of the three sampling strategies S_1^{4u}, S_2^{4u}, S_3^{4u}, presented in Figure 3. Strategy S_1^{4u} is used for all minors. We then implement S_3^{4u} as alternative strategy for the minors that contain S_3^{4u}, while we use S_2^{4u} for the rest of minors.

None of the above sampling strategies can be a-priori considered as the best one for all cases. The reason is that, according to our analysis, the smaller the size of the sample space, the smaller is the variance of the estimation, the better the strategy, since we need fewer samples to obtain a good approximation. We compare the accuracy of the alternative sampling strategies for 6 Wikipedia graphs and 6 AS graphs. The experimental results match the theoretical analysis: for all graphs the winning strategy is the one that uses the smallest sample space. We do not present the results due to lack of space.

4.2 Clustering networks

Inspired by previous work [25, 24], we use the distribution of subgraph counts for characterizing different types of networks. We use clustering to investigate whether features based on minors can be used to partition the input networks into meaningful families (e.g., web graphs, food-chain networks, protein interaction networks, etc.) and if they outperform simpler features based on classical measures used in literature to characterize complex networks [12].

Our approach is to represent graphs by vectors, where each coordinate corresponds to the number of occurrences of one particular minor. Let G be a graph and let t_n be the number of distinct minors of n nodes. Let $|M_j|$ be the number of instances of the j-th minor with $j = 0, .., t_n$. We then represent G by the vector $m(G) =< m_1, \ldots, m_{t_n} >$, where the coordinate m_j is the normalized number of occurrences of minor M_j. Then, given a set of graphs from different families we cluster the graphs by clustering the vectors by which the graphs are represented.
For the clustering task we use Weka.[1] We use two clustering algorithms, the Expectation-maximization (EM) and the k-means algorithms [16], varying the number of clusters

[1]http://www.cs.waikato.ac.nz/ml/weka

Table 2. Results for minor M_{11} in the graphs extracted from 5 crawls of the Web domains .cnr, .eu, .in, .uk

Graph	Nodes	Edges	$\mid S \mid = 10,000$			$\mid S \mid = 100,000$			$\mid S \mid = 1,000,000$		
			$\widetilde{N}$△	Qlt(%)	Time	$\widetilde{N}$△	Qlt(%)	Time	$\widetilde{N}$△	Qlt(%)	Time
cnr-2000	$3.2 \cdot 10^5$	$3.2 \cdot 10^6$	8 910 124	-5.43	1.02	9 261 635	-1.70	5.71	9 349 349	-0.77	80.23
eu-2005	$8.6 \cdot 10^5$	$1.9 \cdot 10^7$	137 742 793	-4.50	4.85	140 576 158	-2.54	9.82	142 990 124	-0.87	34.86
in-2004	$1.38 \cdot 10^6$	$1.69 \cdot 10^7$	820 098 732	4.84	4.00	797 908 977	2.01	9.70	780 542 774	-0.21	69-92
uk-2002	$1.8 \cdot 10^7$	$1.9 \cdot 10^8$	$4.3 \cdot 10^9$	-	77.43	$4.4 \cdot 10^9$	-	117.45	$4.3 \cdot 10^9$	-	206.49
uk-2005	$3.9 \cdot 10^7$	$9.36 \cdot 10^8$	$3 \cdot 10^{10}$	-	302.64	$2.9 \cdot 10^{10}$	-	354.23	$2.9 \cdot 10^{10}$	-	649.93

from 4 to 8. We compare how well the computed clusters match with the original classes, adopting the *classes to cluster evaluation* method implemented in Weka. We label each instance with the type of network it belongs to. During the clustering, this label is ignored. In a second phase, the majority class in each cluster is determined.

Undirected graphs. The best result is achieved with the EM method for $k = 7$ clusters: more than 75% of the instances are correctly classified. Due to lack of space, results are deferred to an extended version of this paper.

Directed graphs. For the directed graphs we compare the following 5 groups of features:

1. *Standard topological properties:* We consider 16 different measures for every node, including indegree, outdegree, average indegree of successors, average outdegree of predecessors, assortativity, edge reciprocity, PageRank, and local number of triangles. In order to assign the considered microscopic measures to each graph, we compute the mean, the variance, the median, the 10-percentile and 90-percentile of them, for a total number of 81 features.
2. *Minors with* 3 *nodes*, for a total number of 13 features.
3. *Minors with* 4 *nodes containing a* K_{12} *clique*, for a total number of 190 features.
4. *All the minors of size* 3 *and* 4, for a total number of 203 features.
5. *All the properties listed above*, for a total number of $81 + 203 = 284$ features.

For all the listed cases, we run the k-means algorithm imposing $k = 7$ clusters. The matching matrices for the first 4 cases are shown in Table 3. In all the cases at least 5 out of 7 classes are correctly identified. We number from 0 to 6 the clusters that match the classes cellular, food-web, word, citation, wikipedia, Webgraph and synthetic and we use the numbers greater than 7 for the clusters matching no class.

In the first case, i.e., using standard topological properties, the k-means algorithm is able to correctly classify the $\mathbf{74}\%$ of the instances. When using only minors of size 3, $\mathbf{77.78}\%$ of the instances are correctly classified. In the third case, i.e., using only minors of size 4, we correctly classify $\mathbf{84.26}\%$ of the instances. Using all the minors, $\mathbf{90.74}\%$ of the instances are correctly classified. It is worth observing that using all the 284 features together (classical + minors) does not improve the result achieved using only the minors-counting methodology.

The results show that minors outperform standard topological features and play a fundamental role for the characterization of complex networks.

4.3 Swap randomization

We assess the statistical significance of our method by comparing the number of minors found in a network with the number of minors observed in random networks with the same degree sequence, generated through a sequence of swaps between pairs of edges of the graph [15].

We generate 5 randomized networks for each of the 43 real directed cellular graphs. For each graph we perform $1000 \cdot \#\{\text{edges}\}$ swaps, a number much larger than what has been empirically considered sufficient to obtain a random graph [15]. The application of the EM algorithm with two clusters correctly separates all but one of the randomized networks from all the real ones. We then conclude that 3- and 4-node minors contain valuable information about the structure of the real networks.

5 Conclusions

We have proposed a suite of methods for approximating the number of 3- and 4-node minors for directed and undirected graphs. Our algorithms are based on random sampling. They can be used to estimate with high precision the number of minors in a graph using limited storage and constant per unit processing time, and making three passes on the input graph.

We provide an optimized implementation of the algorithms and test them on networks extracted from more than 10 application domains. We then propose a network-clustering algorithm based on the frequency of occurrence of all minors, which achieves a precision by far higher than performing clustering based on simpler topological properties. The quality of our data mining techniques has also been tested using swap randomization.

Table 3. Matching Matrix for the clustering using k-means method with k=7 clusters using (1) classical topological properties; (2) only 3 nodes minors; (3) only 4 nodes minors; (4) 3 and 4 nodes minors

	(1)									(2)								(3)									(4)							
assigned to	0	1	2	3	4	5	6	7	8	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	8	0	1	2	3	4	5	6	7
cellular	43	0	0	-	0	-	0	0	0	31	0	0	0	0	-	0	12	43	0	-	-	0	0	0	0	0	43	0	0	-	0	0	0	0
food-web	0	5	0	-	0	-	1	0	0	0	6	0	0	0	-	0	0	2	3	-	-	0	0	0	0	1	0	4	0	-	0	0	0	2
word	0	0	4	-	0	-	0	0	0	0	0	4	0	0	-	0	0	4	0	-	-	0	0	0	0	0	0	0	4	-	0	0	0	0
citation	0	0	0	-	3	-	0	0	0	0	1	0	2	0	-	0	0	0	0	-	-	3	0	0	0	0	0	0	0	-	3	0	0	0
wikipedia	0	0	0	-	7	-	0	0	0	0	0	0	0	6	-	1	0	0	0	-	-	7	0	0	0	0	0	0	0	-	7	0	0	0
Webgraph	0	0	0	-	6	-	0	0	0	0	0	0	0	2	-	4	0	0	0	-	-	0	2	3	0	1	0	1	0	-	2	1	2	0
synthetic	0	0	0	-	9	-	21	5	4	0	0	0	4	0	-	35	0	0	0	-	-	0	0	36	3	0	0	0	0	-	0	0	39	0
Correct	74%									77.78%								$84,26\%$									90.74%							

References

[1] COSIN Project. http://www.cosin.org.

[2] DIP database at the University of California. http://dip.doe-mbi.ucla.edu.

[3] Laboratory of Web Algorithmics, Università degli Studi di Milano. http://law.dsi.unimi.it.

[4] The 9th DIMACS Implementation Challenge on Shortest Paths. http://www.dis.uniroma1.it/c̃hallenge9.

[5] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.

[6] Z. Bar-Yosseff, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA*, 2002.

[7] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.

[8] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Link-based characterization and detection of Web Spam. In *AIRWeb*, 2006.

[9] L. Buriol, C. Castillo, D. Donato, S. Leonardi, and S. Millozzi. Temporal evolution of the wikigraph. In *Web Intelligence*, 2006.

[10] L. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler. Counting triangles in data streams. In *PODS*, 2006.

[11] L. Buriol, G. Frahling, S. Leonardi, and C. Sohler. Estimating clustering indexes in data streams. In *ESA*, 2007.

[12] Da, F. A. Rodrigues, G. Travieso, and Villas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1), 2007.

[13] M. Deshpande, M. Kuramochi, and N. Wale. Frequent substructure-based approaches for classifying chemical compounds. *TKDE*, 17(8), 2005.

[14] J. Gehrke, P. Ginsparg, and J. Kleinberg. Overview of the 2003 KDD Cup. *SIGKDD Explorarion Newsletters*, 2003.

[15] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. In *KDD*, 2006.

[16] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[17] M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *External memory algorithms*, 1999.

[18] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *ICDM*, 2003.

[19] H. Jeong, B. Tombor, R. Albert, Z. Oltvai, and A. Barabasi. The large-scale organization of metabolic networks. *Nature*, 407, 2000.

[20] H. Jowhari and M. Ghodsi. New streaming algorithms for counting triangles in graphs. In *COCOON*, 2005.

[21] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, July 2004.

[22] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *FOCS*, 2000.

[23] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. *TKDE*, 16(9), 2004.

[24] 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*, 303, 2004.

[25] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298, 2002.

[26] S. Muthukrishnan. Data streams: algorithms and applications. *FTTCS*, 1(2), 2005.

[27] M. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 2006.

[28] M. E. J. Newman. The structure of scientific collaboration networks. *Proc Natl Acad Sci USA*, 98(2), 2001.

[29] A. Ostlin and R. Pagh. Uniform hashing in constant time and linear space. In *STOC*, 2003.

[30] N. Pržulj, D. G. Corneil, and I. Jurisica. Efficient estimation of graphlet frequency distributions in protein–protein interaction networks. *Bioinformatics*, 22(8), 2006.

[31] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions Mathematical Softwware*, 11(1), 1985.

[32] S. Wernicke. Efficient detection of network motifs. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(4), 2006.

[33] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *ICDM*, 2002.

[34] X. Yan, P. Yu, and J. Han. Graph indexing: A frequent structure based approach. In *SIGMOD*, 2004.