DATA STREAM COMPUTATION FOR MONITORING STATISTICS OF MASSIVE WEBGRAPHS

LUCIANA S. BURIOL*, DEBORA DONATO*, STEFANO LEONARDI*
AND TOBIAS MATZNER+

Abstract. We are interested in computing properties of large graphs, as the webgraph, using data stream algorihtms. In this work we report results on computing the indegree rarity distribution of a graph obtained as a stream of edges. We implement a rarity algorithm proposed in the literature and show experimentally that the results approximate very well the optimal value with very limited use of memory and time. Moreover, considering some structure in the stream, we present results for the algorithm adapted for maintaining the rarity distribution of the number of cliques of size three.

1. Introduction

Data stream algorithms aim to maintain the underlying information of a stream of data, using small memory space. The data is processed on the fly, as it is generated, or it can also be read from second memory devices. Typical applications of data stream algorithms are originated from massive datasets such as network traffic measurements, telephone call records, biological datasets and atmospheric observations. In these applications is unnecessary or impractical to read data multiple times. In many cases, the data is not even stored. This paper focuses on a "new" natural application for data streams. We are interested in using data stream algorithms for monitoring statistical and topological properties of large graphs such as the webgraph. By webgraph we mean the directed graph generated from the link structure of webpages: each webpage is a node and each hyperlink is an arc in this graph. The graph read in a streaming fashion considers each edge as an item and the stream is not required to be structured.

Several theorical results have been proposed in this new research field, some of them have not yet been implemented and experimented, some of them are not practical. In this paper we observe how a data stream algorithm behaves in practice for computing the indegree rarity distribution of a graph over the arc arrivals. More specifically, we maintain the distribution of the number of nodes that has a given indegree over the total number of different nodes seen in the stream so far. We use the algorithm proposed by Datar and Muthukrishnan [5] and show experimentally that the results are very close to the optima even when a low precision is requested. The original algorithm proposes the use of min-wise hash functions, whereas we

use universal hashing [5]. This decision is due to the fact that computing min-wise hashing consumes about two orders of magnitude more time than universal hashing without providing better results in practice for the graphs we have tested.

When considering a specific structure in the data stream, other properties can be computed. For example, reading the stream in an adjacency list fashion, the same rarity algorithm can be used for estimating the density of minors such as small bipartite cliques.

The indegree of webpages is an important measure of their popularity. The experimental observation of the indegree distribution has been the subject of seminal works aimed to characterize the structure of the web-graph [2, 3]. This study has also revealed a surprising number of dense subgraphs, specifically bipartite cliques, of moderately small size [8], considered as cores of hidden web communities.

In the next section we present the α-rarity algorithm of Datar and Muthukrishnan [5]. Section 3 describes the adaptations of the α-rarity algorithm for computing indegree rarity distribution, as well as for computing minors of small size. In section 4 we present experimental results for rarity of indegree e bipartite cliques of size three `(k3,3)` and for the minors `k(1,3)` and `k(2,3)`. We generalize the set of all minors mentioned above using the term `k(i,3)`, where i denotes the number of nodes in the graph that points to each node of a triple (set of three nodes). Comparison with the results of an optimal computation shows excellent practical results of our implementations.

2. Estimating rarity over data stream windows

We use the $\alpha - rare$ algorithm of Datar and Muthukrishnan [5] for driving our experiments. Consider a stream of items a_i generated in a universe `U`=[1,..,n]. A stream is a set of m elements $a_1, a_2, ..., a_m$ such that $a_i \in U$. An item i is called $\alpha - rare$ if it appears exactly α times in the stream. Let's call $\#\alpha - rare$ the number of elements that appear exactly α times in the stream. Likewise, $\#distinct$ denotes de number of distinct items in the stream. The α-rarity ρ_α is defined as the ratio $\rho_\alpha = \frac{\#\alpha - rare}{\#distinct}$. In other words, the α-rarity of a stream is the measure of number of items that repeat exactly α times in the stream.

The algorithm proposed by Datar and Muthukrishnan [5] for computing the α-rarity of a stream uses min-wise hash functions. Min-wise independent permutation families are defined in [4]. The referred algorithm uses only *O(log N + log u)* space, and *O(log log N)* per item processing time. It estimates ρ by $\hat{\rho} \in [1 \pm \epsilon]\rho + \epsilon\rho$ for a given fraction ϵ, with hight probability. The algorithm uses $h = 2\epsilon^{-3}p^{-1}log\tau^{-1}$ hash functions and two $|h|$-vectors, `min` and `count`, in main memory. Each position i of the vector `min` contains the minimum value found so far by the min-wise hash h_i, whereas `count` maintains, for each position i, the number of times that the current minimum

min-wise value was found. For each value of α, $\hat{\rho}$ is computed as the ration between the number of counters that have exactly value α and h.

A slightly different algorithm is proposed for computing the α-rarity of a windowed stream. E.g, Considering a fix window size equal to `W`, the algorithm maintains the α-rarity of the last `W` items seen in the stream.

3. Computing the rarity distribution of indegree and `k(i,3)`

In this sections we describe how the not-windowed algorithm described in the previous section is adapted to compute the α-rarity algorithm for computing the indegree and `k(i,3)` rarity distributions of a graph.

Consider an arbitrary scan of a digraph G=(V,E). The items of the stream, in this case, are the list of edges. The α-rarity of the stream can be understood as the percentage of nodes that has indegree α. With the underlying data stored for estimating α, we can compute the α_i- rarity for any value of i. The rarity distribution can be computed for a complete stream, or for the window of the last `W` items seen in the stream.

Reading the stream in an adjacency list fashion, the same rarity algorithm can be used for approximating the density of minors, such as small bipartite cliques. Such kind of structured data stream can be found naturally on some applications. For example, during a crawling process, each current fetched page is parsed and all outgoing links of this page identified. In this case G is also read in a streaming fashion, but considering all outgoing links of a node $i \in V$ in sequence. The lists of outgoing edges are not required to be in any specific order, as well as the edges intern to each list. So, for each node u, for each outgoing edge $(\overrightarrow{u,a}) \in OUT(u)$, triples are calculated considering node a and all combinations two by two of the head-node of the edges seen so far in $OUT(u)$. E.g, triples `(a,b,c)` are calculated for nodes $b, c \in OUT(u)$ considering edges $(\overrightarrow{u,b})$ and $(\overrightarrow{u,c})$ previously located in $OUT(u)$ than $(\overrightarrow{u,a})$. So, the overall number of triples (T) of the graph is the sum of the combination three by three of head-nodes of the outgoing list of each node $u \in V$, e.g., $T = \sum_{i=1}^{i} \frac{d_i*(d_i-1)*(d_i-2)}{6}$ where $d_i = |OUT(i)|$ is the outdegree of the node i. We require to store in main memory the whole outgoing adjacency list of the current node.

4. Experimental Results

In this section we describe the experimental results we performed using the α-rarity algorithm applied in maintaining the indegree and `k(i,3)` distributions.. The algorithms were coded in g++ version 3.3.2. The experiments were conducted in a Intel Pentium IV, with 1GB RAM, running Mandrake 9.0.

Due to the excessive computational time spent by min-wise hash functions, we use universal hash functions instead. We used the hash function (`hash31`) and the random number generator (`prng_int`) from the online available codes from the MassDAL group of Rutgers

(*http://www.cs.rutgers.edu/ muthu/massdal-code-index.html*). We use an optimized version of Jerry Zhao's implementation [10] of an approximate restricted min-wise independent permutation family proposed by Alon et al. [1].

We conducted our experiments on streams of Wikipedia graphs. A graph of this type is generated from the link structure of the online and free-content encyclopedia Wikipedia (*www.wikipedia.org*). Following the definition of a webgraph, each article is a node, and each hyperlink is a link in the graph. One graph is extracted for each language. We generate streams of edges of the wikipedia graphs following their generation on time. Due to space restrictions, we limited the presentation of experimental results in this extended abstract to the `wikiEN` and `wikiPT` graphs. The graphs were obtained from an old dump of July 2004. Some comments are added about the experimental results on the other three graphs. Graph `wikiPT` contains 8,131 nodes and 48,168 edges, while graph `wikiEN` is two orders of magnitude larger containing 286,754 nodes and 4,065,530 edges.

Figure 1 presents results for the rarity for the unbounded case, using 100 (right) and 1000 (left) hash functions. The lines are plot for a logarithmic number of indegree values. For a good approximation, a larger number of hash functions are required. But we observed, that even with a small number of hash functions, the results are close to the optima. The plot omits results for indegree higher than 63 for the sake of clarity of the figure, but a complete plot would present additional lines on the bottom of the figure, appearing on increasing order of the number of edges processed.

For the windowed case, similar quality of results were found, but spending more time, as it was expected.

We also found good approximation when using the α-rarity algorithm for computing the rarity distribution of `k(i,3)` on the graph. Results for i=1,2,3 are plot in Figure 2. The plot is in log scale to be able to visualize all three distributions clearly on the same plot. Usually the number of `k(1,3)` $\gg$ `k(2,3)` $\gg$ `k(3,3)`. The difference between this values decrease with the increase of i. Observe, for example, the precision on results between the estimated and exact computation of `k(1,3)` and `k(2,3)`. Since `k(1,3)` is found many more times than `k(2,3)`, the results are more accurate. For values of $i > 4$ we did not plot for the sake of clarity of the plot, but the precision on the results decrease with the increase of i. As expected, we have less precision for computing $\hat{\rho}$ of α-rare elements that occur less frequently.

5. Concluding remarks

In this paper we use in practice data stream algorithms for computing statistical and topological properties of large graphs. We presented experimental results for the α-rarity algorithm applied on webgraphs for computing the rarity distribution of indegree and `k(i,3)` and obtained very good approximations. For the windowed case, applied for the indegree distribution,

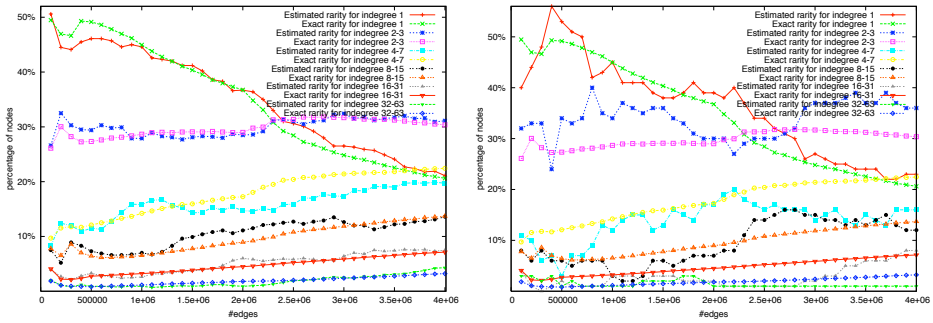

FIGURE 1. Estimated and exact indegree rarity distribution computed for edges arrivals of graph `wikiEN`. The estimation makes use of 1000 (graph on the left) and 100 (graph on the right) universal hashing functions. Values are presented to α up to 63, presented as $\log_2$ plot. This plot presents the percentage of nodes with a given indegree (y-basis) considering the amount of edges processed so far (x-basis). Results are plot every 100,000 items processed.

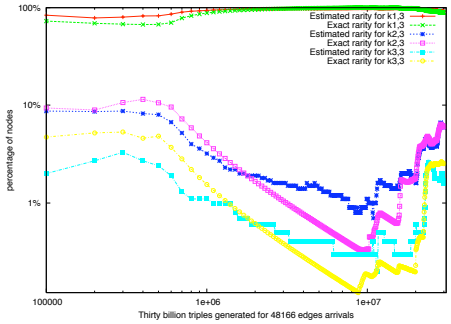

FIGURE 2. Plot in log scale of the estimated and exact `ki,3` rarity distribution, for i=1,2 and 3, computed for edges arrivals of graph `wikiPT`. The estimation makes use of 1000 universal hash functions. This plot presents the percentage of triples pointed by exactly i nodes (y-basis) considering the amount of triples seen so far (x-basis). The triples are computed accordingly with the edges arrivals. Results are plot every 10,000 triples processed.

we observed again good approximation in a reasonable time. For the `k(i,3)` estimation we obtained good approximations, but spending long time. That happens because, in this case, all triples obtained are hashed by the `#h` hash functions. For the `wikiPT` graph, we observe a total of 624 triples generated for each edge processed.

We conclude that using universal hashing by this algorithm speed up a lot the codes, maintaining good approximations.

As further work, we would like to test other algorithms that estimates interesting statistical and topological properties of webgraphs. Moreover, dynamic aspects of webgraphs also could be explored, as edges being inserted and removed over time. The α-rarity algorithm does not have solution for deletions. But a recent publications [6, 7] support also deletions.

6. Acknowledgements

We are very thankful to Jerry Zhao for providing the first version of the min-wise hash functions and for the suggestions for its optimization. We also thanks S. Muthukrishnan for several helpful discussions.

References

[1] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Journal of Random structures and Algorithms*, 3(3):289–304, 1992.

[2] A.L. Barabasi and A. Albert. Emergence of scaling in random networks. *Science*, (286):509, 1999.

[3] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, S. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks*, 33:309–320, June 2000.

[4] A.Z. Broder, M. Charikar, A.M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Proc. of STOC*, pages 327–336, 1998.

[5] M. Datar and S. Muthukrishnan. Estimating rarity and similarity over data stream windows. *LNCS*, 2461:323–334, 2002.

[6] Irina Rozembaum G. Cormogode, S. Muthukrishnan. Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. *Proccedings of the 31st VLDB Conferenct*, 2005.

[7] C. Sohler G. Frahling, P. Indyk. Sampling in dynamic data streams and applications. *21st Annual Symposium on Computational Geometry*, 2005.

[8] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the web for emerging cyber communities. pages 403–416, 1999.

[9] D. Sivakumar Z. Bar-Yosseff, R. Kumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 623–632, 2002.

[10] J. Zhao. An implementation of min-wise independent permutation family. 2005. http://www.icsi.berkeley.edu/ zhao/minwise/.

(*) Dipartimento di Informatica e Sistemistica, Universitá di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy

E-mail address: {buriol,donato,Stefano.Leonardi}@dis.uniroma1.it

(+) Fakultät für Informatik, Universität Karlsruhe, Karlsruhe, Germany

E-mail address: tobias.matzner@rechnerpost.de