1. **Graph Synopses, Sketches, and Streams: A Survey**

**Author**: Sudipto Guha University of Pennsylvania, Philadelphia, PA

Andrew McGregor University of Massachusetts Amherst, MA (马萨诸塞大学阿默斯特分校)

**Published** in: Journal Proceedings of the VLDB Endowment archive Volume 5 Issue 12, August 2012 Pages 2030-2031

 August 2012

**Content:**

1. **Graph Stream Summarization: From Big Bang to Big Crunch(从宇宙大爆炸到宇宙大收缩)**

**Author**: Nan Tang Qatar Computing Research Institute, HBKU, Doha, Qatar (卡塔尔计算研究所, 哈法利大学)

Qing Chen Qatar Computing Research Institute, HBKU, Doha, Qatar

Prasenjit Mitra Qatar Computing Research Institute, HBKU, Doha, Qatar

**Published** in: Proceeding SIGMOD '16 Proceedings of the 2016 International Conference on Management of Data Pages 1481-1496

June 26 - July 01, 2016

**Overview:**

* present TCM, a novel graph stream summary
* discuss a wide range of supported queries and establish some error bounds
* experimentally compare TCM with existing summarizations

**Future Work:**

Since evaluating queries over multiple sketches of TCM is naturally parallelizable, we plan to implement it on a distributed platform e.g., GraphX, so as to handle big graph streams in practice, in a more scalable way.

**Content:**

1. **Introduction to the graph sketch and summarization**

It presents TCM, a novel graph stream summary. Given an incoming edge, it summarizes both node and edge information in constant time. Consequently, the summary forms a graphical sketch where edges capture the connections inside elements, and nodes maintain relationships across elements.

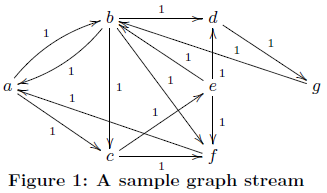
Compared with The widely used practice of summarizing data streams is to treat each stream element independently by e.g., hash- or sample-based methods, without maintaining the connections (or relationships) between elements. Hence, existing methods can only solve ad-hoc problems, without supporting diversified and complicated analytics over graph streams.

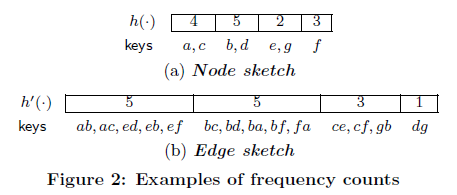
(TCM保持了图的顶点和边的信息，目前广泛应用的summarization 独立的对待流中每个元素)

Summarization: Given a graph stream G, directed or undirected, the problem of graph stream summarization is to summarize G as SG with a much smaller (sublinear) space, linear construction time and constant maintenance cost for each edge update, such that SG allows many queries over G to be approximately conducted efficiently. Some properties about SG are :

* |SG||G|: the size of SG is far less than G, preferably in sublinear space.
* The time to construct SG from G is in linear time.
* The update cost of SG for each edge insertion/deletion is in constant time.
* SG is a graph. (TCM is, the others not)

Example1. Approximate frequency counts with sketches of widely used summarization





First, put the node or edge into one bucket, use hash function;

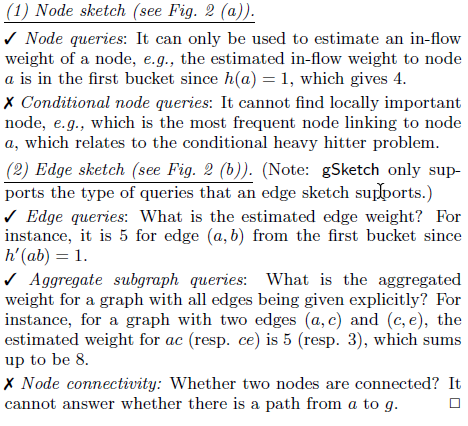
Second, compute the value of bucket;

The value of bucket in Node sketch : sums up the in-flow weights of key nodes

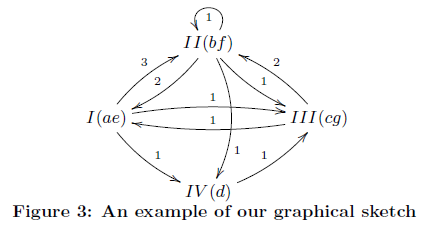
The value of bucket in Edge sketch : sums up the weights for edges key edges

The power of frequency counts based approaches is limited inherently by its structure: It is a vector of (hashed key, value) pairs and each hashed key is independent of each other. That is, one cannot reason for any relationships between two hashed keys. In fact, CountMin or gSketch cannot support any more queries than their frequency counts counterparts, since they are proposed to improve the accuracy.

See the types of queries a sketch supports, and not supports.



Example2 : Approximate frequency counts with graphical sketch of TCM summarization



In contrast to previous sketches that have one-dimensional data structures, TCM is a graphical sketch with a two-dimensional data structure. TCM keeps all structural connectivities of the original graph stream, not only edges, but also paths. This salient property provides rich structural information to support various graph analytics (see Section 4 for a detailed discussion).

The queries supported and not supported by existing node or edge sketches are all supported.

eg. Conditional node queries and Node connectivity.

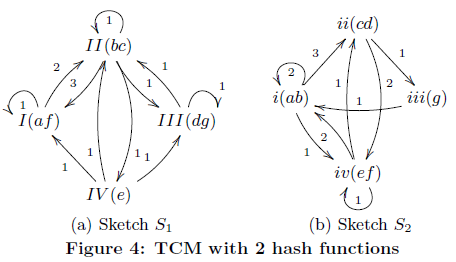
1. **The TCM Model**

Graph stream: A graph stream is a sequence of elements e = (x, y; t) where x, y are node identifiers (labels) and edge (x, y) is encountered at time-stamp t. Such a stream,

G = <e1, e2, …, em>

naturally defines a graph G = (V,E) where V is a set of nodes and E is a set of edges as { e1, …, em }. We write  the weight for the edge ei, and  the aggregated edge weight from node x to node y. We call m the size of the graph stream, denoted by |G|. In this work, we assume that edge weight is non-negative (i.e., ≥ 0).

The TCM model: A TCM sketch is a set of graph sketches. Here, we use d hash functions h1, · · · , hd, where  is used to generate Si. Also, h1, · · · , hd are pairwise independent hash functions. Eg.



Using multiple hash functions can indeed improve the accuracy of estimation

Q1: estimate the aggregated edge weight from b to c 1

Q2: compute the aggregated weight from g to b 1

1. **TCM powered graph analytics**

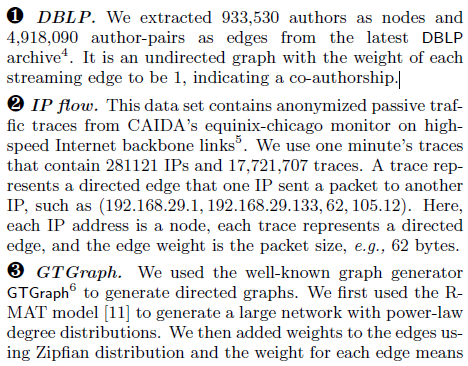
* Edge Queries (estimate the communication frequency between two specific friends.)
* Node Queries (find heavy hitters, DoS (Denial-of-service) attacks for cyber security)
* Path Queries(verify the availability of network service, IP routing)
* Subgraph Queries

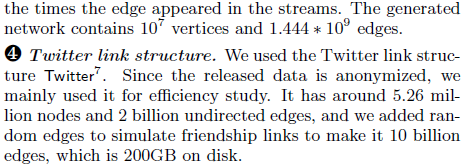
1. **TCM internals**
2. **Experiments**

Purpose:

* prove that TCM should serve as the new backbone for graph stream management
* prove that TCM is as good as ad-hoc sketches for specific problems but is much more general in supporting a wide range of analytics not supported by existing sketches

DataSet:





Environment:

All algorithms were implemented in C++. We used the same hash functions as adopted in an open source Count-Min code8. All our experiments were conducted on an Intel PC with a 3.4 GHz CPU and 32GB RAM, running Ubuntu.

Method: Comparison with State-of-the-Art

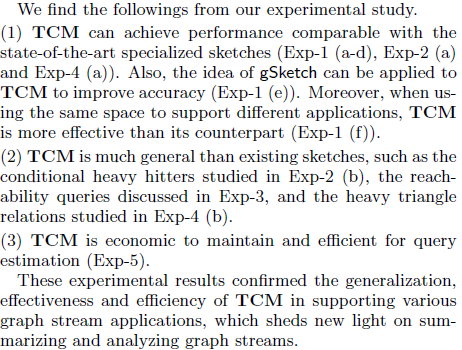
* In sketches for data streams, there are two main types: frequency counts based (e.g., CountMin and gSketch) and sample-based [29].
* Compare TCM , CountMin and gSketch on frequency counts.

Metrics:

* Average relative error
* Intersection accuracy (交点精度).

Results:

Summary:



1. **Counting triangles in data streams**

**Author**: Luciana S. Buriol Universidade Federal de Santa Maria, Brazil

Gereon Frahling University of Paderborn, Germany

Stefano Leonardi Universitá di Roma "La Sapienza", Italy

Alberto Marchetti-Spaccamela Universitá di Roma "La Sapienza", Italy

Christian Sohler University of Paderborn, Germany

**Published** in: Proceeding PODS '06 Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. Pages 253-262

June 26 - 28, 2006

**Overview:**

* present two space bounded random sampling algorithms that compute an approximation of the number of triangles in an undirected graph given as a stream of edges.

1. First algorithm: does not make any assumptions on the order of edges in the stream. It uses space that is inversely related to the ratio between the number of triangles and the number of triples with at least one edge in the induced subgraph, and constant expected update time per edge ------ adjacency stream(邻接流)
2. Second algorithm: is designed for incidence streams (all edges incident to the same vertex appear consecutively). It uses space that is inversely related to the ratio between the number of triangles and length 2 paths in the graph and expected update time O(log |V | · (1+s · |V |/|E|)), where s is the space requirement of the algorithm.------ incidence stream(发生流)

* implement both algorithms and evaluat their performance on networks from different application domains.

1. The sizes of the considered graphs varied from about 8, 000 nodes and 40, 000 edges to 135 million nodes and more than 1 billion edges.
2. For both algorithms we run experiments with parameter s = 1, 000, 10, 000, 100, 000, 1, 000, 000 to evaluate running time and approximation guarantee

**Content:**

1. **Introduction to graph computing , challenge and stream model**

Graph -> subgraph -> stream

Graphs are fundamental structures for modeling complex relationships between data in Web documents, chemical compounds, XML, social networks etc. A basic tool to uncover their structural design principles and to extract relevant information is to mine the most frequent interconnection patterns. The computation of network indices based on counting the number of certain small subgraphs is a basic tool in the analysis of the structure of large networks occurring in the graph.

Counting the number of certain subgraphs in a large graphis a challenging computational task, The current state of the art provides methods that are either computational infeasible on large data sets or do not provide any guarantee on the accuracy of the estimation. Eg. counting the number of triangles in a subgraph, reduces to matrix multiplication. This is not computational feasible even on graphs of medium size, because of time complexity and the space required to store the whole graph and the related data structures in main memory

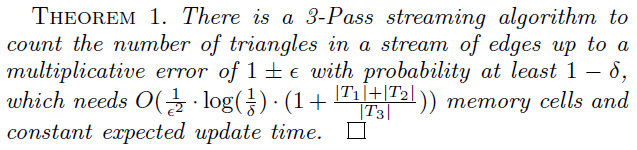
A natural way to address the problem of computing with massive data sets is to resort to the data stream model. In this model data arrives in a stream, one item at a time, and the algorithms are required to use very little space and per-item processing time. Secondary and slower memory storage devices naturally produce data streams for which multiple passes of computation are usually prohibitive due to the volumes of stored data. In several network contexts, the application receive data without pace from remote sources. Data stream computation allows also to compute on-line relevant quantities without incurring a large cost for organizing and storing data.

Bar- Yosseff, Kumar and Sivakumar [20] give a first solution for counting triangles in the data stream model. Consider both the “adjacency stream” model where the graph is presented as a sequence of edges in arbitrary order and there is no bound on the degree of a vertex, and the “incidence stream” model where they consider only bounded-degree graphs and all edges incident to a vertex are presented successively.

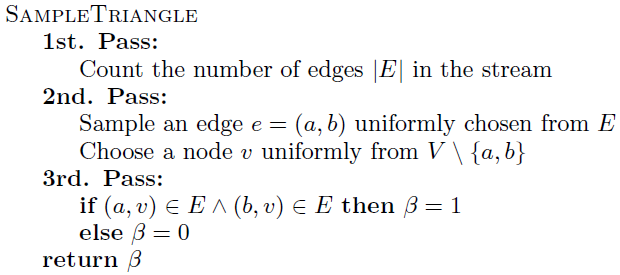
1. **Streams of edges in arbitrary order**

* undirected graphs without self-loops
* each edge is an unordered pair of nodes (v,w) such that (v,w) = (w, v).
* V = {1, . . . ,n} and n is known in advance
* have access to a stream consisting of all edges in the graph, and the edges appear in arbitrary order and no edge is repeated in the stream
* no bound on the degree of the nodes

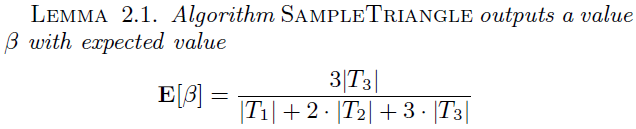
1. **3 Pass Algorithm**



**Algorithm:**

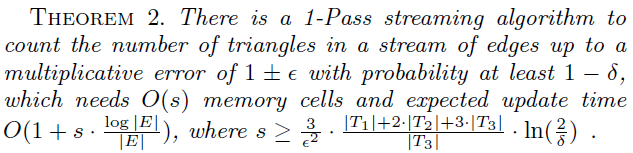


**Lemma**

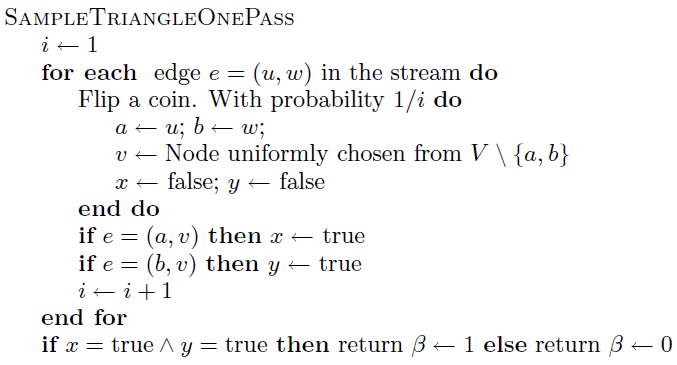




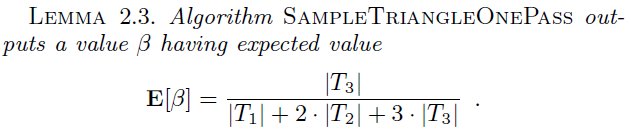
1. **1 Pass Algorithm**

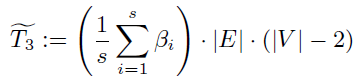


**Algorithm:**



**Lemma**

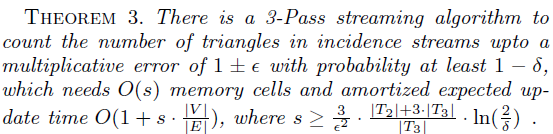




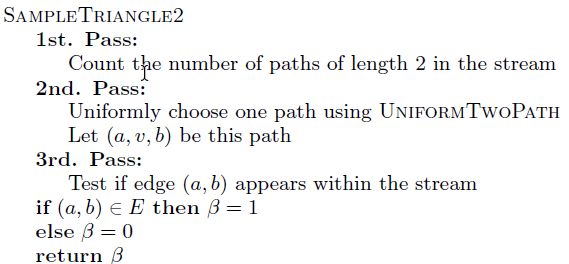
1. **Incidence streams**

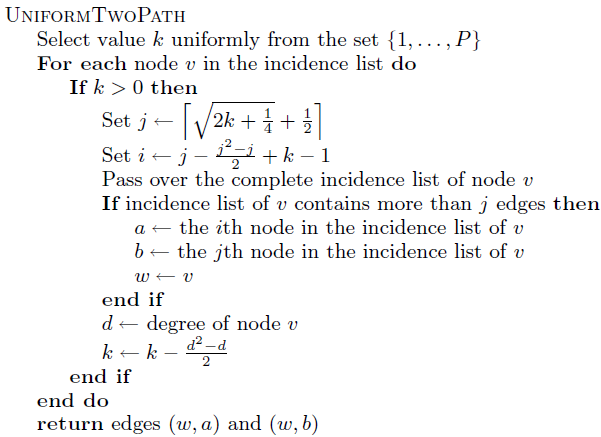
* all edges incident to the same vertexappear subsequently in the stream
* the ordering v1, . . . , vn of the vertices can be arbitrary
* undirected graphs and so each edge appears twice
* no bound on the degree of the nodes

1. **3 Pass Algorithm**

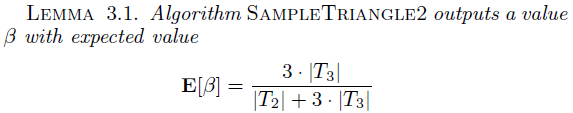


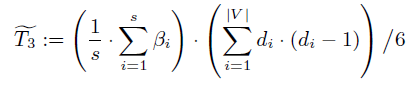
Algorithm:

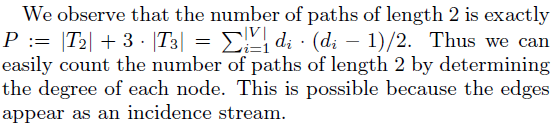




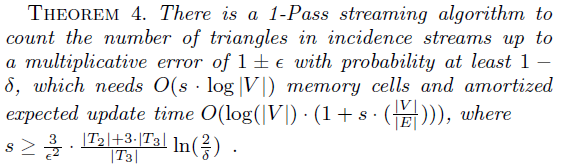
**Lemma**





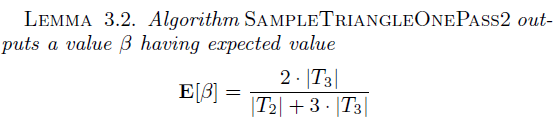


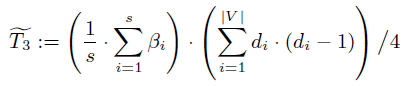
1. **1 Pass Algorithm**



**Algorithm**:

**Lemma**





1. **Experiments**

* One Pass Algorithm for Streams of Edges in Arbitrary Order
* One Pass Algorithm for Incidence Streams

Environment:

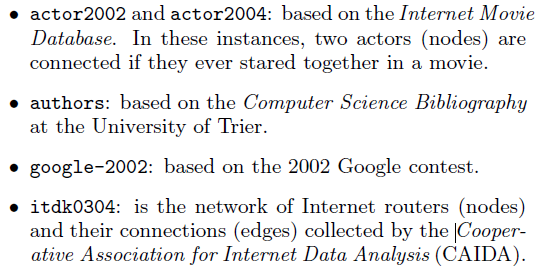
The codes were written in C/C++, and compiled with the gcc compiler version 3.2.2. The experiments were performed on a 2.4 GHz Intel Pentium IV computer with 512 MB of RAM, running Linux, and compiled with g++ version 3.3.2. Due to space requirements, the experiments for the webgraph were performed in a 2.8 GHz Intel Pentium IV computer with 1 GB of RAM, running Linux.

Dataset:

The datasets were divided in three subsets, all of them are comprised of real world instances.

① The first subset is composed of only one instance webgraph, 135 million nodes, 1 billion edges obtained from a graph extracted in 2001 by the WebBase project at Stanford [21] by removing the frontier nodes, i.e, the nodes that have indegree equal to one and outdegree equal to zero.

② The second set of instances is composed of instances used in the experiments reported in [15].



③ The third set of instances is originated from the link structure of WikipediaWikipedia is nowadays the largest online encyclopedia, available in more than 100 languages. In these graphs, each article is a node and each hyperlink between nodes identifies a directed arc. A graph is extracted from each language. The experiments were performed considering the graphs wikiEN, wikiDE, wikiFR, wikiES, wikiIT and wikiPT, extracted from the English, German, French, Spanish, Italian and Portuguese languages, respectively.

Result:

①One Pass Algorithm for Streams of Edges in Arbitrary Order

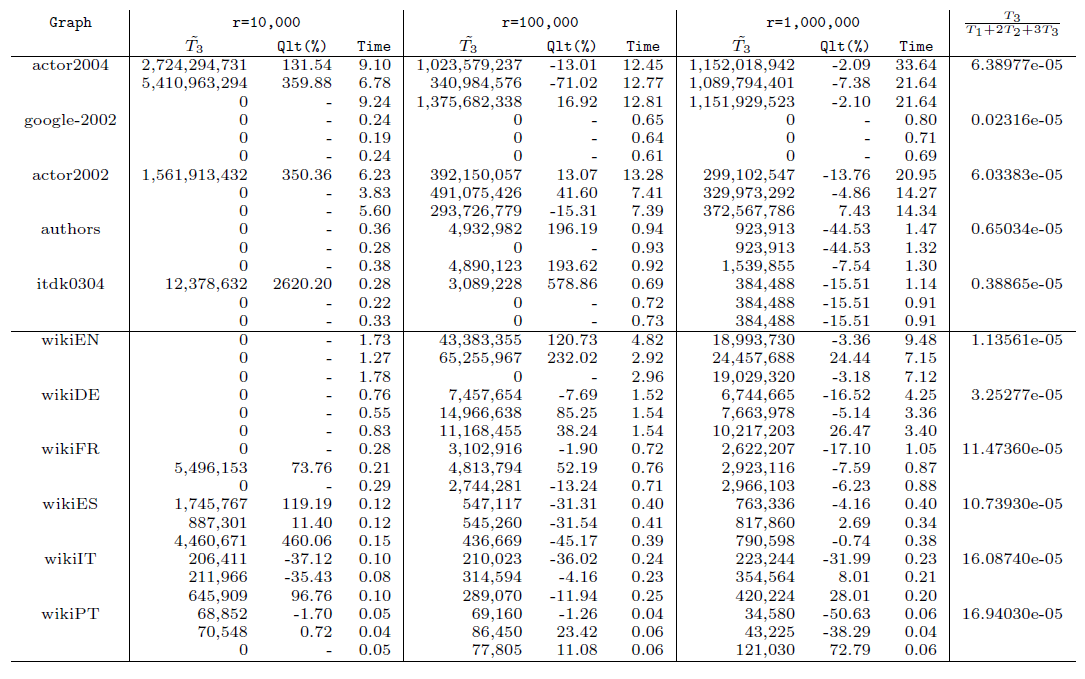
r : sample sizes

E(β) :T3/(T1+2T2+3T3)

(*T*˜3) : estimated number of triangles

Qlt(%) : the quality of the result

We used the algorithm from [15] for computing in main memory the exact number *T*3 of triangles



② One Pass Algorithm for Incidence Streams

The average percentage deviation is very good, even for sample size of 1,000 samples. Considering the absolute values, the average percentage deviation for all instances, but webgraph, are 17.72%, 5.10%, 2.17% and 0.85% for the sample sizes of 1,000, 10,000, 100,000 and 1,000,000, respectively.

We consider that an approximation of 5% is a very good estimative, and so, for this algorithm, a sample set of size 10,000 provides already good results.

