Graph Mining on Streams

Andrew McGregor
Microsoft Research SVC

SYNONYMS

Graph Streams; Semi-Streaming Model

DEFINITION

Consider a data stream $A = \langle a_1, a_2, \ldots, a_m \rangle$ where each data item $a_k \in [n] \times [n]$. Such a stream naturally defines an undirected, unweighted graph $G = (V, E)$ where

$$V = \{v_1, \ldots, v_n\} \text{ and } E = \{(v_i, v_j) : a_k = (i, j) \text{ for some } k \in [m]\} .$$

Graph mining on streams is concerned with estimating properties of G, or finding patterns within G, given the usual constraints of the data-stream model, i.e., sequential access to A and limited memory. However, there are the following common variants.

Multi-Pass Models: It is common in graph mining to consider algorithms that may take more than one pass over the stream. There has also been work in the *W-Stream* model in which the algorithm is allowed to write to the stream during each pass [9]. These *annotations* can then be utilized by the algorithm during successive passes and it can be shown that this gives sufficient power to the model for PRAM algorithms to be simulated [8]. The *Stream-Sort* model goes one step further and allows *sorting passes* in which the data stream is sorted according to a key encoded by the annotations [1].

Weighted, Dynamic, or Directed Graphs: For many problems it is implicitly assumed that the elements a_k are distinct. When the data items are not distinct, the stream naturally defines a multi-graph, i.e., an edge (v_i, v_j) has multiplicity equal to $|\{k : a_k = (i, j)\}|$. In such a model it would be natural to consider the deletion of edges but this has not been explored thoroughly. Finally, the definition can be generalized to define weighted graphs where a third component of the data item, $a_k \in [n] \times [n] \times \mathbb{R}^+$, would indicate a weight $w_{(u,v)}$ associated with the edge (u, v), or directed graphs.

Adjacency and Incidence Orderings: In some applications it is reasonable to assume that all edges incident on the same node arrive consecutively. This assumption defines the *incidence model*. In the *adjacency model* no such assumption is made.

HISTORICAL BACKGROUND

Graph problems were considered by Henzinger et al. [15] in one of the earliest papers on the data-stream model. Their work considered a variety of problems including pointer jumping problems based on the degrees of various nodes in a directed layered graph. Unfortunately, many of the results showed that a large amount of space is required for these types of problems. Subsequent early work considered counting the number of triangles in a graph [2] and estimating common neighborhoods [3]. Again, a large component of these results were negative. It seemed that more "complicated" computation was not possible in this model using only a small amount of space.

It seems that most graph algorithms need to access the data in a very adaptive fashion. Since the entire graph can not be stored, emulating a traditional algorithm may necessitate an excessive number of passes over the data. This has motivated various specific stream models tailored to processing graphs, including the *Semi-*

Streaming, *W-Stream*, and *Sort-Stream* models. The semi-streaming model is characterized by an $O(n \operatorname{polylog} n)$ space restriction, i.e., space proportional to the number of nodes rather than the number of edges. For dense graphs this represents considerably less space than that required to store the entire graph. This restriction was identified as an apparent "sweet-spot" for graph streaming in a survey article by Muthukrishnan [17] and was first explored by Feigenbaum et al. [13]. The W-Stream and Stream-Sort models, described earlier, were introduced by Demetrescu et al. [9] and Aggarwal et al. [1] respectively.

SCIENTIFIC FUNDAMENTALS

This section discusses the main upper and lower bounds known for solving graph problems in the data-stream model. The focus is on results for the standard data-stream model (including the Semi-Streaming space restriction) but note that some of the below problems, including testing connectivity, computing shortest paths, and constructing minimum spanning trees, have been considered in the W-Stream and Stream-Sort models [9, 1].

Connectivity and Minimum Spanning Trees: Determining whether a graph is connected is a fundamental problem. It can be solved in the semi-streaming model and it can be shown that any one-pass algorithm requires $\Omega(n)$ space [15]. More generally, connectivity is a *balanced property* where a graph property $\mathcal{P}$ is *balanced* if there exists a constant $c > 0$ such that for all sufficiently large n, there exists a graph $G = (V, E)$ with $|V| = n$ and $u \in V$ such that:

$$\min\{|\{v : (V, E \cup \{(u, v)\}) \text{ has } \mathcal{P}\}|, |\{v : (V, E \cup \{(u, v)\}) \text{ has } \neg\mathcal{P}\}| \} \geq cn \ .$$

It was shown that all balanced properties require $\Omega(n)$ space [12]. This was part of the justification for the semi-streaming space restriction. However, many problems become feasible with $O(n \operatorname{polylog} n)$ space such as testing planarity and determining whether a graph is bipartite. Testing higher degrees of vertex-connectivity has also been considered in the semi-streaming model. The below table summarizes the start of the art results for determining if a graph is k-connected, i.e., whether k vertices need to be removed in order to disconnect the graph:

k	Passes	Time (per-edge)	Ref.
1,2,3	1	$O(\alpha(n))$	[12]
4	1	$O(\log n)$	[12]
k	1	$O(k^2 n)$	[18]
k	$k+1$	$O(k + \alpha(n))$	[18]

where $\alpha(n)$ is the inverse Ackerman function. Lower bounds have also been investigated for k-edge and k-vertex connectivity [15]. Finally, a related result include a semi-streaming, one-pass algorithm for constructing the minimum spanning tree with $O(\log n)$ processing-time per edge.

Distances and Spanners: An undirected graph $G = (V, E)$ naturally defines a distance function $d_G : V \times V \to \mathbb{R}$ where $d_G(u, v)$ is the length of the shortest path in G between u and v. The diameter of G is the length of the longest shortest path, i.e.,

$$\text{Diam}(G) = \max_{u,v \in V} d_G(u, v) \ ,$$

and the girth of G is the length of the shortest cycle in G, i.e.,

$$\text{Girth}(G) = \min_{(u,v) \in E} [w_{(u,v)} + d_{G \setminus (u,v)}(u, v)] \ .$$

To date most of the algorithms for approximating quantities related to distance are based on constructing sparse *spanners*, where a subgraph $H = (V, E')$ is an (α, β)-spanner of $G = (V, E)$ if, for any vertices $x, y \in V$,

$$d_G(x, y) \leq d_H(x, y) \leq \alpha \cdot d_G(x, y) + \beta \ .$$

Note that constructing an $(\alpha, 0)$-spanner H for an unweighted graph also gives an indication of the girth of the original graph. In particular, if $H \neq G$ then $\text{Girth}(G) \leq \alpha + 1$ because there exists $(u, v) \in E(G) \setminus E(H)$ and this must satisfy $d_{G \setminus (u,v)}(u, v) \leq \alpha$ if H is an $(\alpha, 0)$-spanner.

The first spanner constructions in the data-stream model were presented in [13, 12]. The state of the art construction is due to Elkin [10] who presented a randomized, single-pass algorithm that constructs a $(2t-1,0)$-spanner for an unweighted graph in $O((t\log n)^{1-1/t}n^{1+1/t})$ space (with probability $1-1/n^{\Omega(1)}$). The algorithm processes each edge (u,v) in $O(1)$ expected time and $O(\log d_{u,v}/\log\log d_{u,v})$ worst-case time where $d_{u,v}$ is the sum of the degrees of u and v. The algorithm can be generalized to weighted graphs by rounding edge weights to powers of $(1+\epsilon)$, constructing spanners for each edge set with the same weight, and taking the union of these spanners. This adds a factor of $O(\epsilon^{-1}\log\omega)$ (where ω is the ratio between the biggest and smallest weights) in the space and time complexity and adds a factor of $(1+\epsilon)$ in the approximation factor of a distance. Elkin and Zhang [11] present various algorithms for constructing (α,β)-spanners.

If only a specific distance, $d_G(u,v)$, needs to be approximated then it may appear that constructing a spanner to the entire graph is excessive. However, this is not the case. In particular, it can be shown that any single-pass algorithm that approximates the (weighted) graph distance between two given nodes up to a factor t with probability at least 3/4 requires $\Omega(n^{1+1/t})$ space [13]. Furthermore, this bound also applies even with the promise $d_G(u,v)=\mathrm{Diam}(G)$. Consequently approximation via spanners is at most a factor 2 from optimal.

Counting Triangles: As was noted earlier, approximating the number of triangles in an undirected graph was one of the earliest problems considered in the data-stream model [2]. The number of triangles is related to the clustering and transitivity coefficients of the graph. The state of the art [4] are one-pass, randomized algorithms that estimate the number of triangles up to a multiplicative factor or $(1+\epsilon)$ with probability at least $1-\delta$ using space:

Model	Space
Adjacency	$O(\epsilon^{-2}(1+T_1/T_3+T_2/T_3)\log\delta^{-1})$
Incidence	$O(\epsilon^{-2}(1+T_2/T_3)\log\delta^{-1}\log n)$

where T_i is the number of node triples upon which the induced sub-graph has exactly i edges. While both of these space bounds can be large (e.g., for dense graphs with few triangles) this compares favorably to the

$$O(\epsilon^{-2}(1+T_0/T_3+T_1/T_3+T_2/T_3)\log\delta^{-1})$$

space required by the naive algorithm that randomly samples node-triples and computes the fraction that are triangles. It can also be shown that any p-pass algorithm determining if the number of triangles is non-zero requires $\Omega(p^{-1}n^2)$ space [2]. There has also been work on estimating the count of larger cycles or cliques. However, it can be shown using results in extremal graph theory and a reduction from the set-disjointness communication problem that any p-pass algorithm that determines if $\mathrm{Girth}(G)\geq g$ requires $\Omega\left(p^{-1}(n/g)^{1+4/(3g-4)}\right)$ space [13].

A related problem is that of estimating *path aggregates*. Define P_k to be the number of pairs of vertices that are joined by a simple path of length k. For a graph with r connected components, it is possible to approximate P_2 in one pass and $O(\epsilon^{-2}m(m-r)^{-1/4}\log\delta^{-1})$ space even if edges may be deleted [14]. A space lower bound of $\Omega(\sqrt{m})$ is also known.

BFS trees: A common subroutine in many traditional graph algorithms is that of constructing a breadth-first-search (BFS) tree. Unfortunately it can be shown that computing the first l layers of a breadth-first-tree from a prescribed node requires either $\lceil(l-1)/2\rceil$ passes or $\Omega(n^{1+1/l})$ of space [13].

Matchings: Given a graph $G=(V,E)$, the *Maximum Cardinality Matching* (MCM) problem is to find the largest set of edges such that no two adjacent edges are selected. More generally, for an edge-weighted graph, the *Maximum Weighted Matching* (MWM) problem is to find the set of edges whose total weight is maximized subject to the condition that no two adjacent edges are selected. The following semi-streaming algorithms are known for these problems:

Weighted	Passes	Approximation Ratio	Ref.
No	1	0.5	[13]
No	$O_\epsilon(1)$	$1-\epsilon$	[16]
Yes	1	0.179	[19]
Yes	$O_\epsilon(1)$	$0.5-\epsilon$	[16]

The first of the above algorithms is based on a simple greedy approach, i.e., the algorithm always maintains a matching and adds the latest edge if it is not adjacent to a currently stored edge. The third and fourth algorithms are variants on this basic idea. The second algorithm is based on finding augmenting paths for a currently stored matching, i.e., odd length paths such every second edge is in the currently stored matching. For each augmenting path found it is possible to increase the size of the currently stored matching by one. It can be shown that if there are only a relatively small number of $O(\epsilon^{-1})$-length augmenting paths then the current matching is already a good approximation. Alternatively, if there are many short augmenting paths then it is possible to find a constant fraction using a randomized approach.

Degree Distributions and Random Walks: Given a stream of edges with possible duplicates, a natural question that arises is to estimate properties of the underlying graph. Work has been done on estimating the frequency moments, heavy hitter, and range sums of the degrees of this underlying graph [6]. For example, if d_v is the degree of v in the underlying graph then it is possible to approximate $M_2 := \sum_v d_v^2$ in one pass and $O(\epsilon^{-4}\sqrt{n}\log n)$ space. Note that many of these problem are solvable using standard techniques if there are no duplicates in the stream of edges. A related problem is to estimate the entropy of a random walk on an undirected, unweighted graph. Here the graph stream is an observation of a random walk whose states are nodes of the graph. There exists a single-pass $O(\epsilon^{-4}\log^2 n \log^2 \delta^{-1})$ space algorithm that estimates the entropy of the walk [5]. These algorithms use a combination of algorithms for counting distinct items and the AMS sampling technique. The problem of actually constructing random walks has also been considered [7]. This has applications to estimating the page-rank vector, mixing time and conductance of graphs.

KEY APPLICATIONS*

Massive graphs arise naturally in many real world scenarios. Two examples are the *call-graph* and the *web-graph*. In the call-graph, nodes represent telephone numbers and edges correspond to calls placed during some time interval. In the web-graph, nodes represent web pages, and the edges correspond to hyper-links between pages. When processing these graphs it is often appropriate to use the data-stream model. For example, the graph may be revealed by a web-crawler or the graph may be stored on external memory devices and being able to process the edges in an arbitrary order improves I/O efficiency. One of the major drawbacks of traditional graph algorithms, when applied to massive graphs such as the web, is their need to have random access to the edge set. Massive graphs also arise in structured data mining, where the relationships among the data items in the data set are represented as graphs, and social networks.

FUTURE DIRECTIONS

There are numerous specific open problems that arise from the existing work on graph streams. For example, one could attempt to improve the approximation factors of the known approximate matching algorithms or the space required to approximate M_2. Another idea is to explore distance approximation in multiple passes. While computing exact distance may require many passes this does not preclude the possibility of better approximation with a few additional passes.

Other more general ideas include investigating graph problems in the *probabilistic data-stream model* or the *random-order data-stream model*. In the probabilistic data-stream model, a probability p_e would be assigned to each edge e. The goal of an algorithm would be to estimate the probability that the random graph generated has a certain property given that each edge is present independently with probability p_e. In the random-order data-stream model the assumption, as the name suggests, is that the edges arrive in random order. The goal is to design algorithms that estimate some property with high probability where the probability is defined over both the coin flips of the algorithm and the ordering of the stream.

CROSS REFERENCE*

stream models, synopsis structure, one-pass algorithm

RECOMMENDED READING

[1] Gagan Aggarwal, Mayur Datar, Sridhar Rajagopalan, and Matthias Ruhl. On the streaming model augmented with a sorting primitive. *IEEE Symposium on Foundations of Computer Science*, pages 540–549, 2004.

[2] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 623–632, 2002.

[3] Adam L. Buchsbaum, Raffaele Giancarlo, and Jeffery Westbrook. On finding common neighborhoods in massive graphs. *Theor. Comput. Sci.*, 1-3(299):707–718, 2003.

[4] Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. Counting triangles in data streams. In *ACM Symposium on Principles of Database Systems*, pages 253–262, 2006.

[5] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. A near-optimal algorithm for computing the entropy of a stream. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 328–335, 2007.

[6] Graham Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *ACM Symposium on Principles of Database Systems*, pages 271–282, 2005.

[7] Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. Estimating PageRank on graph streams. In *ACM Symposium on Principles of Database Systems*, pages 69–78, 2008.

[8] Camil Demetrescu, Bruno Escoffier, Gabriel Moruz, and Andrea Ribichini. Adapting parallel algorithms to the w-stream model, with applications to graph problems. In *Mathematical Foundations of Computer Science*, pages 194–205, 2007.

[9] Camil Demetrescu, Irene Finocchi, and Andrea Ribichini. Trading off space for passes in graph streaming problems. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 714–723, 2006.

[10] Michael Elkin. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. In *International Colloquium on Automata, Languages and Programming*, pages 716–727, 2007.

[11] Michael Elkin and Jian Zhang. Efficient algorithms for constructing $(1+\epsilon, \beta)$-spanners in the distributed and streaming models. *Distributed Computing*, 18(5):375–385, 2006.

[12] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM Journal of Computing* (to appear).

[13] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.

[14] Sumit Ganguly and Barna Saha. On estimating path aggregates over streaming graphs. In *International Symposium on Algorithms and Computation*, pages 163–172, 2006.

[15] Monika R. Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. *External memory algorithms*, pages 107–118, 1999.

[16] Andrew McGregor. Finding graph matchings in data streams. In *APPROX-RANDOM*, pages 170–181, 2005.

[17] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Now Publishers, 2006.

[18] Mariano Zelke. k-connectivity in the semi-streaming model. *CoRR*, cs/0608066, 2006.

[19] Mariano Zelke. Weighted matching in the semi-streaming model. In *Symposium on Theoretical Aspects of Computer Science*, 2008.