Information Processing Letters 111 (2011) 145–150

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Intractability of min- and max-cut in streaming graphs

Mariano Zelke

Institut für Informatik, Goethe-Universität, Frankfurt am Main, Germany

ARTICLE INFO

Article history:
Received 31 March 2010
Received in revised form 27 September 2010
Accepted 23 October 2010
Available online 28 October 2010
Communicated by F.Y.L. Chin

Keywords:
Graph algorithms
Streaming algorithms
Intractability
Min-cut
Max-cut

ABSTRACT

We show that the exact computation of a minimum or a maximum cut of a given graph G is out of reach for any one-pass streaming algorithm, that is, for any algorithm that runs over the input stream of G's edges only once and has a working memory of $o(n^2)$ bits. This holds even if randomization is allowed.

1. Introduction

The key assumption of the traditional RAM-model, cf. [2], is the existence of a main memory containing the whole input data and allowing fast random access to this data. This assumption is unrealistic when facing massive input data that exceed the sizes of common main memories. To process such massive input data, the area of streaming algorithms was developed. A *streaming algorithm* [21] reads the input as a stream of data items in arbitrary order and is restricted to consume less memory than the size of the input. Because of these features, streaming algorithms are suitable when processing data created by sensor networks in an online fashion without the need to completely store the data. In addition, streaming algorithms are appropriate when tackling data that is stored on external memory devices. On these devices it is very time-consuming to fetch data items via random access, but if the data items are read in the order they are stored, i.e., as a stream, the access time decreases by magnitudes.

There is a rich literature about streams comprising of numerical values; for an overview we refer the reader to [3] and [27]. However, one of the earliest papers [21] in the area of streaming also considers algorithms for graph problems. Muthukrishnan [27] proposed the model of a *semi-streaming algorithm* to approach graph problems in the streaming context. The input for such an algorithm is a stream containing the edges of the input graph G in arbitrary order. The algorithm's working memory is restricted to $\mathcal{O}(n \cdot \operatorname{polylog} n)$ bits where n is the number of vertices in G. Hence, there is enough space to memorize a polylogarithmic number of edges on average for every vertex but graphs that are sufficiently dense cannot be stored completely by a semi-streaming algorithm.

There has been progress in developing streaming algorithms for graph problems. Feigenbaum et al. [13,14] consider semi-streaming algorithms for computing the connected components and a bipartition of a graph, to find a minimum spanning forest, and for testing k-vertex and k-edge connectivity. In [28] we point out how to reduce the running time of each of these semi-streaming algorithms to the corresponding running time of the fastest known RAM-model algorithm. For the problem of approximating the maximum weighted matching in a graph, a series of papers [13,26,29,12] reduced the best known approxima-

E-mail address: zelke@em.uni-frankfurt.de.

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2010.10.017

tion ratio for a semi-streaming algorithm to $4.91+\varepsilon$. There are randomized streaming algorithms that estimate the number of triangles in a graph [5,22,10] and that compute a graph spanner to approximate pairwise vertex distances [11,6].

On the other hand there are graph problems known that are intractable for any streaming computation, that is, for any algorithm that can only access the input as a stream in one or more passes and that is unable to memorize the whole input graph. Henzinger et al. [21] show the intractability of a constant factor approximation of the transitive closure's size of a directed graph for any one-pass streaming algorithm. It is known that there cannot be a one-pass streaming algorithm that identifies all vertex pairs with $1 < s < n-1$ common neighbors in directed graphs [9] or that finds a maximum matching in unweighted bipartite graphs [13]. Ganguly and Saha [17] show that to estimate the number of vertex pairs of distance k for $k \geqslant 3$ in a single pass $\Omega(n^2)$ bits are required which rules out any one-pass streaming algorithm. Feigenbaum et al. [14] establish a bound on multi-pass algorithms by stating that no semi-streaming algorithm can compute the first d levels of a breadth-first-search tree in less than d passes over the input graph for $d < \log n/(\log\log n)^2$.

The results of [14,28,30] about computing k-edge connectivity for $k = \mathcal{O}(\log n)$ in the semi-streaming model can be regarded as the first approach to graph cuts in the streaming context. Apart from this work on cuts of small values, there have been no considerations about cut problems in the domain of streaming algorithms until recently. A paper of Ahn and Guha [1] shows how to transfer ideas of [7] about computing a sparsification of a graph G to the semi-streaming model. Such a sparsification allows a randomized $(1 \pm \varepsilon)$-approximation of the value of every cut in G, particularly of a minimum and a maximum cut, by a one-pass semi-streaming algorithm. The authors of [1] also give lower bounds on the memory consumption of streaming algorithms computing a sparsification.

Independently of these results, the PhD thesis of the present paper's author [30] points out one-pass semi-streaming algorithms approximating minimum and maximum cuts by random sampling techniques. The intractability results presented here are part of the same PhD thesis.

In this paper we show that the exact computation of minimum and maximum cuts is out of reach for any one-pass streaming algorithm, i.e., for any algorithm that runs over the input stream only once and has a working memory of $o(n^2)$ bits, even if randomization is allowed.

In Section 2 we give the required definitions and a short account on the theory of communication complexity that we utilize. While the proofs of intractability can be found in Sections 3 and 4, we conclude in Section 5.

2. Preliminaries

Let $G = (V, E)$ be an undirected simple graph with unweighted edges. We denote the number of vertices by n, the number of edges by m. A *cut* (V_1, V_2) is a partition of the vertices V into two nonempty sets V_1 and V_2. An edge uv *crosses* the cut if one endvertex of uv is in V_1 while the other one is in V_2. We denote by $|(V_1, V_2)|$ the *value* of the cut (V_1, V_2) which is the total number of the edges crossing it.

The *minimum cut problem* is to find a minimum cut in G, that is, a cut of minimum value. We denote this value by c. Correspondingly, the *maximum cut problem* asks for a cut of maximum value; we name this value $\hat{c}$.

A sequence of the edges of G in arbitrary order is called a *graph stream*. A *streaming algorithm* is presented a graph stream of G as the input and uses a working memory restricted to $o(n^2)$ bits.

A streaming algorithm may access the graph stream for P *passes*. Each pass starts at the beginning of the stream and goes over it in the same sequential one-way order. In this paper we are only concerned with algorithms that are limited to a single pass over the input stream.

2.1. Communication complexity

For our intractability proofs we make use of the theory of *communication complexity*. As only a restricted setting of this theory is utilized, the reader is referred to Kushilevitz and Nisan [25] for a comprehensive overview. Let X, Y, and Z be finite sets and $f : X \times Y \to Z$ be a function. There are two players, Alice and Bob, such that Alice is given an $x \in X$ and Bob is given an $y \in Y$. They want to compute $f(x, y)$ but since Alice does not know y and Bob does not know x, they have to communicate, that is, to exchange bits according to some agreed-upon communication protocol depending on f. Such a protocol tells each of the players depending on the own input and the received communication so far what message to send next. The *cost of a protocol* is the number of bits that have to be exchanged to evaluate f in the worst case, i.e., that is maximized over all inputs (x, y).

We consider the problem of *bit vector probing* where Alice knows a bit vector x of length ℓ and Bob has an index $1 \leqslant i \leqslant \ell$. Bob wants to know x_i, that is, the ith bit of x but the communication is only allowed from Alice to Bob, not in the opposite direction. It is known [25] that every protocol that enables Bob to detect x_i is of cost ℓ, that is, requires the communication of the entire vector in the worst case.

The approach to exploit lower bounds of communication complexity to show lower bounds on the memory consumption for streaming computations on graphs has been used for example by Henzinger et al. [21], by Ganguly and Saha [17], and by Feigenbaum et al. [14]. The rough idea is to point out how a streaming algorithm using a small working memory could be used to create a protocol for a problem of communication complexity whose cost contradicts a known lower bound. We follow this idea to prove our theorems.

3. Minimum cut

The first traditional RAM-model algorithm that approaches the minimum cut problem is due to Ford and Fulkerson [15]. It uses the duality between a minimum cut separating a vertex s from another vertex t and a maximum flow from s to t. The minimum cut reflects the

connectivity structure of the graph and can be used to cluster the vertices. An example for the usage of minimum cuts is given by [8] where documents that are linked via a hypertext system are clustered into topically related groups.

The currently fastest algorithm to compute a minimum cut in the traditional RAM-model is due to Gabow [16]. It requires a running time of $\mathcal{O}(m + c^2 n \log(n/c))$, which depends on the minimum cut value c, and uses a space of $\mathcal{O}(m)$. Such an exact computation is unobtainable for any one-pass streaming algorithm as shown by the next theorem.

Theorem 1. *There is no one-pass streaming algorithm using $o(n^2)$ bits of working memory that is able to find a minimum cut in every graph.*

Proof. Let A be a one-pass streaming algorithm using $o(n^2)$ bits of working memory that computes a minimum cut in every graph. We will use A to construct a protocol of cost $o(n^2)$ that solves the bit vector probing problem of communication complexity.

Let Alice have a bit vector x of length $(n^2 - n)/2$. She interprets this vector as the upper half of the adjacency matrix of the graph $G = (V, E)$ on n vertices. After feeding the edges of G into A, she sends the memory configuration of A to Bob followed by a sequence containing the degree of every vertex in G. Since A's memory configuration comprises of $o(n^2)$ bits and $\mathcal{O}(n \log n)$ bits suffice to transmit the vertex degrees, a total of $o(n^2)$ bits are sent from Alice to Bob.

Bob regards his index i, $1 \leqslant i \leqslant (n^2 - n)/2$, as an edge ab whose existence in G he wants to probe. He continues the execution of A by feeding more edges into it, thereby extending G to $G^+ = (V^+, E^+)$ with $V \subsetneq V^+$ and $E \subsetneq E^+$. In particular, Bob adds two disjoint cliques S and T, each of size $3n$ into G where $(S \cup T) \cap V = \emptyset$. Additionally, Bob connects every vertex in $V \setminus \{a, b\}$ to every vertex in T and both a and b to every vertex in S. Define $L := S \cup \{a, b\}$ and $R := T \cup V \setminus \{a, b\}$. Finally, Bob joins a vertex c, $c \notin \{L \cup R\}$, to $d_G(a) + d_G(b) - 1$ vertices in R where $d_G(v)$ denotes the degree of the vertex v in G. Note that $d_G(a) + d_G(b) \geqslant 2$ since otherwise Bob knows immediately that ab cannot be present in G. Fig. 1 depicts the general layout of G^+.

Due to the high connectivity within each of L and R, every cut in G^+ that separates two vertices in L (R, respectively) from each other is of value at least $3n - 1$. There are only two cuts of smaller value in G^+, namely $C_1 := (L, R \cup \{c\})$ and $C_2 := (L \cup R, \{c\})$; both have a value that cannot exceed $2n$.

Now Bob can ask A for the minimum cut of G^+ to decide the existence of ab in G: If ab is in G, the minimum cut of G^+ is given by C_1 of value $d_G(a) + d_G(b) - 2$. Otherwise, $|C_1| = d_G(a) + d_G(b)$ and C_2 is the minimum cut in G^+.

At the end, a one-pass streaming algorithm using $o(n^2)$ bits of memory that computes a minimum cut of every graph could be used to design a communication protocol transmitting $o(n^2)$ bits that solves the bit vector probing problem on a vector of $\Theta(n^2)$ bits. That contradicts the lower bound of $\Omega(n^2)$ bits whose communication is required in the worst case [25].

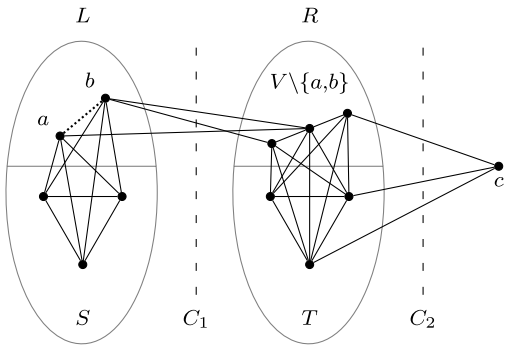

Fig. 1. Sketch of the graph G^+ used in the proof of Theorem 1. The minimum cut is given by either C_1 or C_2 depending on the existence of the edge ab.

Note that not only the structures of the two possible minimum cuts C_1 and C_2 differ but also do their values. A one-pass streaming algorithm computing only the value of the minimum cut without its partition suffices for Bob to detect ab in G. Thus, for such an algorithm the same lower bound holds. □

The area of communication complexity also considers randomized protocols. Such a protocol does not depend only on the players' input strings; in addition to that it also takes random variables $r_1, r_2, \ldots$ into account when specifying the messages exchanged by the players. Thus, the communication itself on a given input (x, y) is not fixed anymore but becomes a random variable instead. The cost of a randomized protocol is the worst case over all inputs and choices of the r_i. The knowledge about randomized protocols allows us to extend the result of Theorem 1 to randomized streaming algorithms.

Corollary 2. *There is no randomized one-pass streaming algorithm using $o(n^2)$ bits of working memory which succeeds to find a minimum cut in every graph with a probability of at least $2/3$.*

Proof. On the same lines as in the proof of Theorem 1, we can use a randomized one-pass streaming algorithm with $o(n^2)$ bits of working memory and success probability p to construct a randomized communication protocol of cost $o(n^2)$ with the same success probability. However, it is known [25] that every protocol solving the bit vector probing problem on a $\Theta(n^2)$ bit vector correctly with a probability of at least $2/3$ must be of cost $\Omega(n^2)$. □

4. Maximum cut

The problem of finding a maximum cut in a weighted graph is one of Karp's original $\mathcal{NP}$-complete problems [23]. On unweighted graphs the problem remains $\mathcal{NP}$-complete [18]; it is even known that the computation of a $(1.0625 - \varepsilon)$-approximative solution is $\mathcal{NP}$-complete for every $\varepsilon > 0$ [20]. The best known approximation ratio of 1.1383 is given by the semi-definite programming

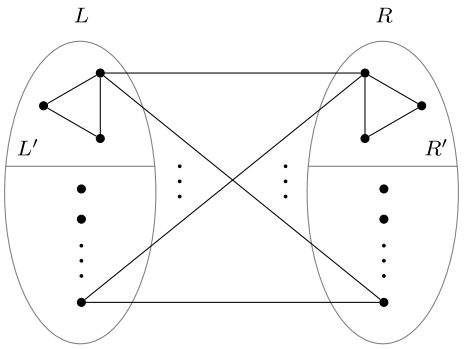

Fig. 2. General layout of the graph H of Lemma 3. Every possible edge between L and R is present, additionally each of L' and R' build a clique.

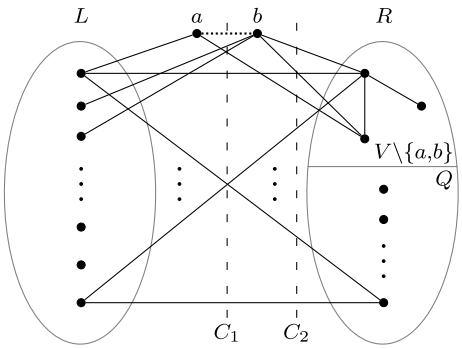

Fig. 3. Sketch of the graph G^+ utilized in the proof of Theorem 4. The existence of the edge ab determines which one of C_1 and C_2 is a maximum cut.

approach due to Goemans and Williamson [19]. Some evidence towards the optimality of this algorithm is presented by Khot et al. [24]. In addition to its theoretical importance, the maximum cut problem has applications in the design of circuit layouts and in statistical physics [4].

In order to prove the intractability of the maximum cut problem for a one-pass streaming algorithm, we need the following technical lemma. The addressed graph is sketched in Fig. 2.

Lemma 3. *Let H be a graph on the vertex classes L and R with $|L| = |R| = 5n - 1$ and let $L' \subsetneq L$ and $R' \subsetneq R$ be both of size n with $n > 0$. Let H contain every possible edge between L and R and let additionally H contain every possible edge inside L' and inside R'. Then any cut in H differing from (L, R) is at most of value $|L| \cdot |R| - 4n + 1$.*

Proof. Starting from the cut (L, R) of value $|L| \cdot |R|$, we can produce any other cut by exchanging vertices in $A \subseteq L$ with vertices in $B \subseteq R$. We can assume that $|A| + |B| \leqslant 5n - 1$ since otherwise $|L \setminus A| + |R \setminus B| < 5n - 1$ whose exchange produces the same cut. After moving A to R, we can estimate the value of the resulting cut $(L \setminus A, R \cup A)$ to be smaller than $|L| \cdot |R| - |A|(5n-1) + |A|n$. After moving B to $L \setminus A$, the resulting cut is of value smaller than

$$|L| \cdot |R| - |A|(5n-1) + |A|n - |B|\big(5n - 1 - |A|\big) + |B|n + |A| \cdot |B|$$

which equals $|L| \cdot |R| + |A|(|B| - 4n + 1) + |B|(|A| - 4n + 1)$. For $|A|, |B| \geqslant 0$ and $0 < |A| + |B| \leqslant 5n - 1$ this term takes its maximum with $|A| + |B| = 1$ at a value of $|L| \cdot |R| - 4n + 1$. □

Now we can show the very same intractability result for the maximum cut problem as we did for the minimization problem.

Theorem 4. *There is no one-pass streaming algorithm using $o(n^2)$ bits of working memory that is able to find a maximum cut in every graph.*

Proof. The proof is similar to the one of Theorem 1 showing the intractability of the minimum cut problem, so we borrow the framework of the proof from there. Remember that Bob receives the memory configuration of the streaming algorithm after reading $G = (V, E)$, which is only known to Alice, followed by the degree of every vertex. Again, Bob wants to probe the edge ab in G.

To this aim he again extends G to obtain G^+. In contrast to the proof of Theorem 1, Bob adds a complete bipartite graph on the color classes L and Q with $(L \cup Q) \cap V = \emptyset$ and $|L| = 5n - 1$, $|Q| = 4n + 1$. He connects every vertex in $V \setminus \{a, b\}$ to every vertex in L and joins vertex a to $d_G(a) - 1$ vertices in L and b to $d_G(b) - 1$ vertices in L. It must be $d_G(a) > 0$ and $d_G(b) > 0$ since otherwise Bob knows directly that ab is not in G.

We define $R := Q \cup V \setminus \{a, b\}$, thus $|L| = |R| = 5n - 1$. See Fig. 3 for the general shape of G^+.

Because each of a and b has at most $n - 1$ neighbors in G and gained at most $n - 2$ new neighbors in G^+, the sum of the degrees of a and b in G^+ is at most $4n - 6$. Let G' be the graph induced by $L \cup R$ and note that G' is a subgraph of H of Lemma 3. Hence, any cut of G' that does not separate L from R has size at most $|L| \cdot |R| - 4n + 1$. Since the sum of the degrees of a and b in G^+ is at most $4n - 6$, any cut in G^+ that does not separate L from R is of size at most $|L| \cdot |R| - 4n + 1 + (4n - 6) = |L| \cdot |R| - 5$, irrespective of where a and b are placed in the cut.

As a result, we get a maximum cut in G^+ when separating L from R and by adding a and b each to L or R. Where a and b are placed in a maximum cut depends on the existence of ab in G.

If ab is in G, a (b, respectively) has the same number of neighbors in L and R. Then every maximum cut separates a from b to make ab an edge crossing the cut. If ab is not in G, the number of a's (b's) neighbors in R exceeds the number of a's (b's) neighbors in L by one. Thus, the maximum cut is given by $(L \cup \{a, b\}, R)$.

Bob can probe the existence of ab in G by asking the one-pass streaming algorithm for the maximum cut of G^+. If the algorithm was content with $o(n^2)$ bits of working memory, that gives rise to a communication protocol solving the bit vector probing problem with cost $o(n^2)$. This contradicts the lower bound of $\Omega(n^2)$ for the cost of every protocol tackling this problem if Alice holds a vector of length $\Theta(n^2)$ [25].

As in the proof of Theorem 1, the value of the maximum cut in G^+ differs depending on the presence of ab in G. Hence, for any one-pass streaming algorithm computing this value without yielding a partition, the proved bound holds as well. □

We can deduce a lower bound for a randomized algorithm in the same way as in Corollary 2.

Corollary 5. *There is no randomized one-pass streaming algorithm using $o(n^2)$ bits of working memory succeeding to find a maximum cut in every graph with a probability of at least $2/3$.*

5. Conclusion

We showed that a precise solution to the minimum/maximum cut problem is unattainable for any one-pass streaming algorithm by utilizing the problem of bit vector probing. This problem, however, does not allow any implications about streaming algorithms using more than one pass over the input. Therefore, we leave open the natural question on the existence of a multi-pass streaming algorithm that exactly computes a minimum or maximum cut.

In the traditional RAM-model, the complexity of the minimum cut problem fundamentally falls below the one of the maximum cut problem, provided $\mathcal{P} \neq \mathcal{NP}$. The streaming model, however, does not emphasize the running time of an algorithm as the most important measure of complexity. Rather, the model focuses on the general question of to which extend a problem is solvable when forbidding random access and restricting working memory.

From this streaming viewpoint, the minimum cut problem does not seem to be easier than the maximum cut problem. For both problems we proved the intractability of finding an exact solution, even if randomization is allowed.

Moreover, a sparsification S of a graph G can be computed by a one-pass semi-streaming algorithm [1]. In a postprocessing step the semi-streaming algorithm can run traditional algorithms to figure out a minimum or a maximum cut of the memorized sparsification S. This way a $(1 \pm \varepsilon)$-approximation of a minimum as well as of a maximum cut in G can be obtained. Admittedly, while there are polynomial time algorithms to identify the minimum cut of S, the postprocessing time for the maximum cut will be exponential. However, this difference is inherited from the traditional RAM-model and does not derogate our view that the minimum and maximum cut problem are equally accessible for streaming computations.

References

[1] K.J. Ahn, S. Guha, Graph sparsification in the semi-streaming model, in: Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP'09), in: LNCS, vol. 5556, Springer, Berlin/Heidelberg, 2009, pp. 328–338.

[2] A. Aho, J. Hopcroft, J. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley Publishing Company, 1974.

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, Models and issues in data stream systems, in: Proceedings of the 21st ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems (PODS'02), 2002, pp. 1–16.

[4] F. Barahona, M. Grötschel, M. Jünger, G. Reinelt, An application of combinatorial optimization to statistical physics and circuit layout design, Oper. Res. 36 (1988) 493–513.

[5] Z. Bar-Yossef, R. Kumar, D. Sivakumar, Reductions in streaming algorithms, with an application to counting triangles in graphs, in: Proceedings of the 13th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'02), 2002, pp. 623–632.

[6] S. Baswana, Streaming algorithm for graph spanners – single pass and constant processing time per edge, Inform. Process. Lett. 106 (2008) 110–114.

[7] A.A. Benczúr, D.R. Karger, Approximating $s-t$ minimum cuts in $O(n^2)$ time, in: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing STOC '96, 1996, pp. 47–55.

[8] R. Botafogo, Cluster analysis for hypertext systems, in: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1993, pp. 116–125.

[9] A. Buchsbaum, R. Giancarlo, J. Westbrook, On finding common neighborhoods in massive graphs, Theoret. Comput. Sci. 299 (2003) 707–718.

[10] L. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, C. Sohler, Counting triangles in data streams, in: Proceedings of the 25th ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems (PODS'06), 2006, pp. 253–262.

[11] M. Elkin, Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners, in: Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP'07), in: LNCS, vol. 4596, Springer, Berlin/Heidelberg, 2007, pp. 716–727.

[12] L. Epstein, A. Levin, J. Mestre, D. Segev, Improved approximation guarantees for weighted matching in the semi-streaming model, in: 27th International Symposium on Theoretical Aspects of Computer Science STACS 2010, 2010, pp. 347–358.

[13] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, J. Zhang, On graph problems in a semi-streaming model, in: Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP'04), in: LNCS, vol. 3142, Springer, Berlin/Heidelberg, 2004, pp. 531–543.

[14] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, J. Zhang, Graph distances in the streaming model: the value of space, in: Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'05), 2005, pp. 745–754.

[15] L.R. Ford, D.R. Fulkerson, Maximal flow through a network, Canad. J. Math. 8 (1956) 399–404.

[16] H.N. Gabow, A matroid approach to finding edge connectivity and packing arborescences, J. Comput. System Sci. 50 (2) (1995) 259–273.

[17] S. Ganguly, B. Saha, On estimating path aggregates over streaming graphs, in: Proceedings of the 17th International Symposium on Algorithms and Computation (ISAAC'06), in: LNCS, vol. 4288, Springer, Berlin/Heidelberg, 2006, pp. 163–172.

[18] M.R. Garey, D.S. Johnson, Computers and Intractability – A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, 1979.

[19] M.X. Goemans, D.P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, J. ACM 42 (1995) 1115–1145.

[20] J. Håstad, Some optimal inapproximability results, J. ACM 48 (2001) 798–869.

[21] M.R. Henzinger, P. Raghavan, S. Rajagopalan, Computing on data streams, in: External Memory Algorithms, in: DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 50, AMS/DIMACS, 1999, pp. 107–118.

[22] H. Jowhari, M. Ghodsi, New streaming algorithms for counting triangles in graphs, in: Proceedings of the 11th Annual International Computing and Combinatorics Conference (COCOON '05), in: LNCS, vol. 3595, Springer, Berlin/Heidelberg, 2005, pp. 710–716.

[23] R.M. Karp, Reducibility among combinatorial problems, in: Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–103.

[24] S. Khot, G. Kindler, E. Mossel, R. O'Donnell, Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? in: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04), 2004, pp. 146–154.

[25] E. Kushilevitz, N. Nisan, Communication Complexity, Cambridge University Press, 1997.
[26] A. McGregor, Finding graph matchings in data streams, in: Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems and 9th International Workshop on Randomization and Computation (APPROX and RANDOM '05), in: LNCS, vol. 3624, Springer, Berlin/Heidelberg, 2005, pp. 170–181.
[27] S. Muthukrishnan, Data Streams: Algorithms and Applications, Foundations and Trends in Theoretical Computer Science, Now Publishers Inc., 2005, http://algo.research.googlepages.com/eight.ps.
[28] M. Zelke, Optimal per-edge processing times in the semi-streaming model, Inform. Process. Lett. 104 (3) (2007) 106–112.
[29] M. Zelke, Weighted matching in the semi-streaming model, in: Proceedings of the 25th Annual Symposium on the Theoretical Aspects of Computer Science (STACS'08), 2008, pp. 669–680.
[30] M. Zelke, Algorithms for streaming graphs, PhD thesis, Humboldt University, Berlin, 2009, http://nbn-resolving.de/urn:nbn:de:kobv:11-10097652.