Tight Approximations of Degeneracy in Large Graphs

Martín Farach-Colton and Meng-Tsung Tsai(✉)

Rutgers University, New Brunswick, NJ 08901, USA
{farach,mtsung.tsai}@cs.rutgers.edu

Abstract. Given an n-node m-edge graph G, the degeneracy of graph G and the associated node ordering can be computed in linear time in the RAM model by a greedy algorithm that iteratively removes the node of min-degree [28]. In the semi-streaming model for large graphs, where memory is limited to $\mathcal{O}(n \operatorname{polylog} n)$ and edges can only be accessed in sequential passes, the greedy algorithm requires too many passes, so another approach is needed.

In the semi-streaming model, there is a deterministic log-pass algorithm for generating an ordering whose degeneracy approximates the minimum possible to within a factor of $(2+\varepsilon)$ for any constant $\varepsilon > 0$ [12]. In this paper, we propose a randomized algorithm that improves the approximation factor to $(1+\varepsilon)$ with high probability and needs only a single pass. Our algorithm can be generalized to the model that allows edge deletions, but then it requires more computation and space usage.

The generated node ordering not only yields a $(1+\varepsilon)$-approximation for the degeneracy but gives constant-factor approximations for arboricity and thickness.

Keywords: Degeneracy · Arboricity · Thickness · Semi-streaming algorithm · Space lower bound

1 Introduction

Any ordering of the nodes of an n-node, m-edge simple undirected graph G defines an acyclic orientation of the edges in which each edge is oriented from the earlier node in the ordering to the latter. The ***degeneracy*** of an ordering is the maximum out-degree it induces. The ***degeneracy*** of G, denoted by $d(G)$, is the smallest degeneracy among all orderings[1], and an ordering whose degeneracy

Work supported by CNS-1408782 and IIS-1247750.

[1] The degeneracy of a graph was originally defined to be the maximum minimum degree among all subgraphs [2,5–7,14,28,34]. The definition here is a slight modification of the ***coloring number*** [5,6,14] of a graph, a dual definition of degeneracy. The coloring number of a graph was shown to be one larger than the degeneracy [5,6,14], and our definition yields the same value as the original definition of degeneracy.

© Springer-Verlag Berlin Heidelberg 2016
E. Kranakis et al. (Eds.): LATIN 2016, LNCS 9644, pp. 429–440, 2016.
DOI: 10.1007/978-3-662-49529-2_32

is $d(G)$ is called a ***degenerate ordering***. An ordering is d***-degenerate*** if it has degeneracy at most d.

Degenerate orderings have many uses. Given a degenerate ordering, one can: decompose a graph into at most twice the minimum number of disjoint forests [2,5]; decompose a graph into at most six times the minimum number of disjoint planar graphs [5,9]; speed up the counting of the number of short paths or cycles [2], for example, counting the exact number of 3-cycles in $Q(md(G))$ time; find a component of density at least half the maximum density of any subgraph, i.e. a 1/2-approximation [7]; identify a dominating set of cardinality at most $Q(d^2(G))$ times the cardinality of a minimum dominating set [26] as well as some variations of dominating sets [10], e.g. k-dominating sets; etc. Although most of these problems can be solved exactly in polynomial time [7,15,16,24], the approximation algorithms based on degenerate orderings are faster, use less space or yield better approximation factors for large graphs. For example, such orderings yield a better approximation algorithm for decomposing a graph into a minimum number of planar subgraphs than other algorithms using $O(n)$ space [22,27]. Although all of the results listed originally relied on (optimally) degenerate orderings, in [12], we show that orderings that are nearly degenerate orderings, that is, whose degeneracy approximates rather than matches the graph degeneracy also yield good approximation algorithms.

The degeneracy of graph G and the associated node ordering can be computed in linear time in the RAM model by a greedy algorithm that iteratively removes the node of min-degree, as shown by Matula and Beck [28]. However, iteratively removing a single min-degree node is inefficient when graphs are larger than memory. Thus, another approach is needed.

We consider algorithms in the semi-streaming model [30,32,33], in which we are allowed $\mathcal{O}(n \operatorname{polylog} n)$ working space, and edges can be accessed in sequential read-only passes through the graph. The goal is then to minimize the number of passes and the time complexity of the algorithm.

Some graph problems that have similar complexities in the RAM model can have quite different complexities in the semi-streaming model. Some graph problems, e.g. connectivity, minimum spanning tree, finding bridges and articulation points, can be solved optimally [11,13]. Other graph problems, e.g. counting the number of 3-cycles, maximum matching and graph degeneracy, can be approximated [1,3,12]. Some fundamental problems, such as breath-first search, depth-first search, topological sorting, and directed connectivity, are believed to be difficult to solve in a small number of passes [18,32,33].

In [12], we give a deterministic log-pass algorithm for generating a node ordering whose degeneracy approximates the minimum possible to within a factor of $(2+\varepsilon)$ for any constant $\varepsilon > 0$. In this paper, we propose a randomized algorithm that improves the approximation factor to $(1+\varepsilon)$ and reduces the number of pass to one but which has a small probability of failure. Theorem 1 is our main result.

Theorem 1. *In the semi-streaming model, there exists an $\mathcal{O}(m)$-time 1-pass randomized algorithm that outputs a node ordering whose degeneracy approximates the minimum possible to within a factor of $(1+\varepsilon)$ for any constant $\varepsilon > 0$ with probability $1 - 1/n^{\Omega(1)}$ using a space of $\mathcal{O}(\varepsilon^{-2} n \log^2 n)$ bits.*

Our algorithm can be generalized to the model that allows edge deletions, known as the dynamic stream [4,8,17,20,21,25,29] or the turnstile model [23,35], by appealing the Jowhari et al. [19] result on building L_0-samplers, which are data structures that can be updated in the streaming model and can generate a sampled edge once the stream has been processed, and the algorithm that deals with sets of L_0-samplers efficiently, due to McGregor et al. [29]. In our case, we need $\mathcal{O}(\varepsilon^{-2}n\log n)$ L_0-samplers to produce a sampled subgraph with $\mathcal{O}(\varepsilon^{-2}n\log n)$ edges. The generalization to the turnstile model increases the time- and space-complexity by $\operatorname{polylog} n$ factors. We summarize the result in Theorem 2.

Theorem 2. *In the turnstile model, there exists an $\mathcal{O}(m \operatorname{polylog} n)$-time 1-pass randomized algorithm that outputs a node ordering whose degeneracy approximates the minimum possible to within a factor of $(1+\varepsilon)$ for any constant $\varepsilon > 0$ with probability $1 - 1/n^{\Omega(1)}$ using a space of $\mathcal{O}(\varepsilon^{-2}n\log^3 n)$ bits.*

In addition to these upper bounds, we also show that computing the degeneracy in the semi-streaming model in one pass with constant success rate has a space lower bound of $\Omega(n\log n)$ bits. The space lower bound also holds in the turnstile model, because the turnstile model is a generalization of the semi-streaming model. We note that our algorithm in the semi-streaming model is optimal in both time- and pass-complexity and has a nearly-optimal space-complexity, which is no more $\log n$ times optimal.

To illustrate how to apply the low-degeneracy node ordering to other problems, we also show constant-factor approximations for arboricity and thickness.

Our Techniques. We sample a small random subgraph $H \subseteq G$ such that H fits in memory. Then, we show that the degenerate ordering of H is a low-degenerate ordering of G, as follows:

In [12], we show that iteratively removing a node of degree no more than $(1+\varepsilon)$ times the minimum degree generates a node ordering whose degeneracy is no more than $(1+\varepsilon)$ times the graph degeneracy. This fact leaves some flexibility in picking the next node to remove, rather than always having to pick the min-degree node, as required in the exact algorithm [28]. Since low degree nodes in H are likely to be low degree nodes in G, we are about to exploit this flexibility to minimize the probability of error in the final order.

Organization. We prove that the degenerate ordering of a random subgraph H is a low-degenerate ordering of graph G in Sect. 2. To obtain a random subgraph H, in Sect. 3 we devise algorithms to sample an H in the semi-streaming model and in the turnstile model. In Sect. 4, we show a lower bound on the space needed to compute a low-degenerate ordering in one pass. Lastly, in Sect. 5, we present some applications of low-degenerate orderings.

2 Degeneracy and Random Subgraphs

We revisit some properties of degeneracy and, based on those, we show that the degenerate ordering of $H = G(p)$ is also a low-degenerate ordering of G, where

$G(p)$ is a random subgraph of G such that every edge in G is included in $G(p)$ independently with probability p.

To begin, let v (resp. $\hat{v}$) be the min-degree node of G (resp. H). By $d_G(v)$ we denote the degree of v in G. Intuitively, since $H = G(p)$ is a sketch of G, the difference between $d_G(v)$ and $d_G(\hat{v})$ is likely to be small. We claim that if p is set to be $\Omega(\varepsilon^{-2} n \log n/m)$,

$$d_G(\hat{v}) \leq \max\left\{(1+\varepsilon)d_G(v), m/n\right\}$$

with probability $1 - 1/n^{\Omega(1)}$. We prove a stronger form of this claim in Lemma 3, in which G_U denotes the subgraph of G induced by node set U, and $\delta(G_U)$ denotes the minimum node degree in G_U.

Lemma 3. *Let $H = G(p)$ be a random subgraph of an n-node m-edge graph G. For any node set U, the node $\hat{v}$ that has minimum degree in H_U has degree in G_U bounded by*

$$\max\left\{(1+\varepsilon)\delta(G_U), m/n\right\} \text{ for any constant } \varepsilon > 0$$

with probability $1 - 1/n^{\Omega(1)}$ if $p = \Omega(\varepsilon^{-2} n \log n/m)$.

Proof. Let v be the min-degree node in G_U, and let Q be the set of bad candidates of $\hat{v}$; formally,

$$Q = \left\{x \in G_U : d_{G_U}(x) > \max\left\{(1+\varepsilon)d_{G_U}(v), m/n\right\}\right\}.$$

We show that the degree $d_{G_U}(\hat{v})$ is bounded as required w.h.p. by considering the two probabilities

$$\Pr\left[\hat{v} \in Q \mid C_1 : d_{G_U}(v) \geq m/n\right] \text{ and } \Pr\left[\hat{v} \in Q \mid C_2 : d_{G_U}(v) < m/n\right],$$

such that $d_{G_U}(\hat{v})$ is not bounded as required. Here we bound the first probability by the Chernoff and Union bounds as follows:

$$\begin{aligned}
\Pr\left[\hat{v} \in Q \mid C_1\right] &\leq \sum_{x \in Q} \Pr\left[d_{H_U}(x) \leq d_{H_U}(v)\right] \\
&\leq \sum_{x \in Q} \Pr\left[d_{H_U}(x) \leq (1-c)pd_{G_U}(x) \vee d_{H_U}(v) \geq (1+c)pd_{G_U}(v)\right] \\
&\leq \sum_{x \in Q} \exp\left(-\frac{c^2}{2}pd_{G_U}(x)\right) + \exp\left(-\frac{c^2}{2+c}pd_{G_U}(v)\right) \qquad (1) \\
&\leq n \exp\left(-\Omega(\log n)\right) \\
&= 1/n^{\Omega(1)},
\end{aligned}$$

where $c = \varepsilon/(2+\varepsilon)$, so that $(1-c)pd_{G_U}(x) \geq (1+c)pd_{G_U}(v)$ if $x \in Q$. The second probability has the same upper bound as well because $\Pr\left[\hat{v} \in Q \mid C_1\right] \geq \Pr\left[\hat{v} \in Q \mid C_2\right]$. $\square$

Lemma 3 implies that the degenerate ordering of $H = G(p)$ is a low-degeneracy ordering of G w.h.p. Before proceeding to the proof of our main claim, we observe the following facts about degeneracy:

1. $d(G) \geq m/n$: the degeneracy of an n-node m-edge graph is at least m/n, because the sum of the out-degrees is m for any node ordering, and therefore the maximum out-degree induced by any ordering is at least m/n;
2. $d(G) \geq d(H)$: the degeneracy of graph G is no less than the degeneracy of any subgraph $H \subseteq G$, because adding edges cannot decrease the degeneracy;
3. $d(G) \geq \delta(G)$: the minimum degree is no more than the degeneracy, because for any node ordering, the induced out-degree of the first node equals its degree and of course cannot be less than the minimum.

We are now in a position to show our main claim, as stated in Theorem 4. We prove this by construction. We obtain the degenerate ordering of $H = G(p)$ by the greedy algorithm [28] that iteratively removes the min-degree node from H until H becomes empty. Such a node removal gives a node ordering $\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_n$. Let the remainder of the graph after the node removal be $G_0 = G$, $G_k = G_{k-1} \setminus \{\hat{v}_k\}$ for each $k > 0$. Note that G_k is a subgraph of G induced by the node set $\{\hat{v}_i : i > k\}$. By Lemma 3, if $p = \Omega(\varepsilon^{-2} n \log n/m)$, we have that

$$\Pr\left[d_{G_{k-1}}(\hat{v}_k) > \max\left\{(1+\varepsilon)\delta(G_{k-1}), m/n\right\}\right] < 1/n^{\Omega(1)} \text{ for each } k,$$

and therefore we have, by the Union bound,

$$\Pr\left[\bigvee_{k\in[1,n]} \left(d_{G_{k-1}}(\hat{v}_k) > \max\left\{(1+\varepsilon)\delta(G_{k-1}), m/n\right\}\right)\right] < 1/n^{\Omega(1)}.$$

In other words, we can say that with probability $1 - 1/n^{\Omega(1)}$ every $\hat{v}_k$ has degree no more than the minimum degree in G_{k-1} or the quantity m/n. Hence, $\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_k$ is a low-degeneracy ordering of G whose degeneracy is bounded by

$$\max\left\{(1+\varepsilon)\delta(G_{k-1}), m/n, d(G_{k-1} \setminus \{\hat{v}_k\})\right\} \leq (1+\varepsilon)d(G)$$

with probability $1 - 1/n^{\Omega(1)}$, where the inequality immediately follows from the abovementioned three facts and by induction on k. As a result, we have:

Theorem 4. *The degenerate ordering of $H = G(p)$ is a low-degeneracy ordering of G whose degeneracy approximates the minimum possible to within a factor of $(1+\varepsilon)$ with probability $1 - 1/n^{\Omega(1)}$ if $p = \Omega(\varepsilon^{-2} n \log n/m)$.*

3 Algorithms

We now present how to compute a node ordering in the considered models such that the computation takes a single pass, and the degeneracy of the ordering is a $(1+\varepsilon)$-approximation of the graph degeneracy w.h.p.

To recap, we propose an algorithm that samples a random subgraph H from the entire graph G, and then outputs the degenerate ordering of the random subgraph. We have shown in Theorem 4 that the ordering is a low-degenerate ordering of the entire graph with a fairly good probability. In particular, we let $H = G(p)$ for $p = \Theta(\varepsilon^{-2} n \log n/m)$, and thus the output ordering has degeneracy that approximates the minimum possible to within a factor of $(1+\varepsilon)$ with probability $1 - 1/n^{\Omega(1)}$.

The sampling procedure is quite different in the semi-streaming and the turnstile models. We discuss them separately. In the semi-streaming model, each edge in the data stream is contained in the final edge set of G. Thus, obtaining a sampled subgraph H is straightforward. To make the algorithm optimal in runtime, we batch the edges, as described below.

On the other hand, an edge on the data stream might be subsequently deleted in the turnstile model. This uncertainty makes sampling hard. To handle the uncertainty we appeal to the L_0-sampler construction of Jowhari et al. [19] and the algorithm for efficiently maintaining sets of such L_0-samplers due to McGregor et al. [29]. In this approach, the sampled subgraph H is an approximate of $G(p)$ that has a small distortion from $G(p)$. We adapt Lemma 3 to account for this distortion.

3.1 In the Semi-streaming Model

In this model the edge set of G is presented in a read-only stream. We assume that n, the number of nodes, is known at the beginning of data stream, but m, the number of edges, is not known until the end of data stream. The desired size of H is

$$s = \Theta(\varepsilon^{-2} n \log n).$$

We note here that s is known at the beginning because it only depends on n and ε. Our goal is to pick a p so that $s = mp$, that is, so that $H = G(p)$ has size $\mathcal{O}(s)$, w.h.p. However, m is unknown, and therefore p is also unknown, when we start sampling. To sample each edge with an unknown probability p, we guess that $m = s$ and set $p = 1$. If there are more than s edges, we adjust the guess to be $m = 2s$, set $p = 1/2$, and then kick out some sampled edges to make sure that all edges were sampled with probability $1/2$. We keep adjusting the guess for m and p and resampling, until we run out of edges. In order to implement this intuition efficiently, we use the following algorithm.

We allocate a working space of size $2s$ to hold the sampled edges, and call this space the ***pool***. If we ever end up selecting more than $2s$ edges, our algorithm goes into a low-probability failure mode in which it outputs an arbitrary node ordering.

Now suppose, for ease of presentation, that m is a multiple of s. The pool is empty at the beginning. Our algorithm works in rounds: in the k-th round, the algorithm brings the k-th group of s edges into a buffer. Then, the algorithm kicks out some of the edges that are already in the pool, if any, each with probability $p_k = 1/k$, and migrates the edges that are in the buffer to the pool, each with probability $q_k = 1/k$, discarding those that fails to migrate. At the end of the (m/s)-th round, the pool contains a randomly sampled subgraph H in which each edge is sampled independently from G with probability $p = s/m$ even though m was unknown initially. See Fig. 1 for an illustration of the sampling procedure.

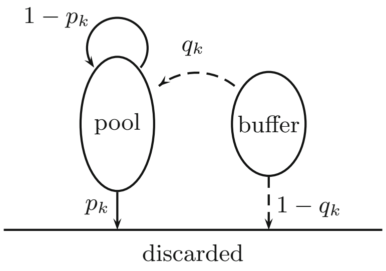

Fig. 1. Sampling procedure in the k-th round.

In the case that m is not a multiple of s, then there are αs edges for $\alpha \in (0, 1)$ in the last-round buffer. For this last round, it suffices to set $p_k = \alpha/(k - 1 + \alpha)$ and $q_k = 1/(k - 1 + \alpha)$.

To quantify how often the sampling procedure has a pool of size greater than 2s, we give a bound on the probability of such a pool overflows in a round. By the Chernoff bound, we have

$$\Pr\left[\sum_{i=[1,m]} X_i > 2mp\right] \le \exp\left(-\Omega(\varepsilon^{-2} n \log n)\right) \le 1/n^{\Omega(n)},$$

where X_i is an indicator variable denoting whether the i-th edge of graph G is included in the random subgraph H. Then, by the Union bound, the probability that the pool overflows at some point of the entire sampling procedure is $1/n^{\Omega(1)}$, which is dominated by the claimed failure rate.

Lastly, we analyze the time complexity of the proposed algorithm. The above sampling procedure has $\mathcal{O}(m/s)$ rounds, and in each round it deals with at most $3s$ edges. Thus, the sampling procedure takes $\mathcal{O}(m)$ time. After obtaining the random sampled subgraph H, the algorithm computes the degenerate ordering of H using the in-memory algorithm for computing degenerate ordering introduced by Matula and Beck [28], which takes $\mathcal{O}(s) = \mathcal{O}(mp)$ time. Hence, the total runtime is bounded by $\mathcal{O}(m)$.

This completes the proof of Theorem 1.

3.2 In the Turnstile Model

In this model a sequence of edge insertions and deletions is presented in a read-only stream. The procedure described in the Sect. 3.1 does not work in this model because the pool in Sect. 3.1 might end up choosing edges that are about to be deleted, making $H = \emptyset$ rather than $H = G(p)$.

Since that the working space cannot keep the entire edge set E ($|E| = m$), using an L_0-sampler [19] one can sample an edge e from E with success rate at least $1 - \delta$ using $\mathcal{O}(\log^2 n \log(1/\delta))$ bits, such that the probability that an edge $e \in E$ is picked in the sample is

$$1/m + 1/n^{\Omega(1)},$$

which is roughly the desired $1/m$ but has a small distortion $1/n^{\Omega(1)}$.

To sample a random subgraph $H = G(p)$, one can use $2s$ L_0-samplers to sample $2s = 2mp/(1-\delta) = \Theta(\varepsilon^{-2} n \log n)$ for constant $\delta < 1$ independent edges from the final edge set E of G. These might be repeated, so pick the first t distinct edges from the $2s$ samples, where t is a random variate sampled from $Bin(m, p)$, the binomial distribution of m trials and success rate p. Note that an L_0-sampler fails to return a sample with probability $1-\delta$, and so we set $2s = 2mp/(1-\delta)$ to balance the failure rate. One can assert that there exist t distinct edges among the $2s$ samples with probability $1 - 1/n^{\Omega(1)}$ by the Chernoff bound. To handle the failure case, again, the algorithm outputs an arbitrary node ordering.

It follows from the above sampling procedure, for each edge $e \in G$, that the probability that H contains e is

$$\Pr\left[e \in H\right] = \sum_{k=[0,m]} \Pr\left[t \sim Bin(m,p) = k\right] \left(k/m + k/n^{\Omega(1)}\right) = p\left(1 + 1/n^{\Omega(1)}\right).$$

Hence, $H \approx G(p)$.

Since $H \neq G(p)$, to prove the correctness of Theorem 2, we adapt Lemma 3 to accommodate the small distortion. The distortion changes the expected values of $d_{H_U}(x)$ and $d_{H_U}(\hat{v})$. However the change is so small that the bounds on the tail probabilities in Eq. 1 still hold. Therefore, Lemma 3 works as well for $H \approx G(p)$.

A naïve implementation of the above requires $\mathcal{O}(\varepsilon^{-2} nm \,\mathrm{polylog}\, n)$ time. We appeal to the alternative sampling procedure devised by McGregor et al. [29] to obtain the abovementioned $2s$ samples in $\mathcal{O}(m \,\mathrm{polylog}\, n)$ time.

This establishes Theorem 2.

4 Space Lower Bounds

In this section, we show that any randomized algorithm in the semi-streaming model that can approximate degeneracy, arboricity or thickness in one sequential pass with constant success rate requires a working space of $\Omega(n \log n)$ bits. Our proofs rely on the space lower bounds of cycle-freeness [35, Theorem 7] and planarity testing [35, Corollary 12] shown recently by Sun and Woodruff.

We observe that, combining the space lower bound of cycle-freeness testing [35, Theorem 7] with Lemma 5, the space lower bound for approximating degeneracy to within a factor of $(2-\varepsilon)$ is immediate, summarized in Theorem 6.

Lemma 5. *Graph G has degeneracy $d(G) = 1$ if and only if G is cycle-free.*

Proof. If G is cycle-free, then G is a forest and one can make every tree in G rooted. In this way, let the orientation of the edges in G from the descendant to the ancestor. Since every node except the roots in such rooted cycle-free graph has a single parent node. The out-degree of every node is either 0 or 1. In other words, the degeneracy $d(G) = 1$.

Otherwise, G contains a cycle C. Since the degeneracy $d(C) \geq \delta(C)$ (Fact 3 in Sect. 2) and $\delta(C) = 2$, then $d(C) \geq 2$. Combining that $d(G) \geq d(C)$ (Fact 2 in Sect. 2), $d(G) \geq 2$. □

Theorem 6. *In the semi-streaming model, any randomized algorithm that can approximate the degeneracy to within a factor of $(2 - \varepsilon)$ for any constant $\varepsilon > 0$ with constant success rate has a space lower bound of $\Omega(n \log n)$ bits.*

In addition, we show similar space lower bounds for computing arboricity and thickness. The proof directly follows from the definition of arboricity and thickness. Recall that the arboricity (resp. thickness) of graph G is the minimum number of forests (resp. planar subgraphs) whose union forms G. Therefore, computing the arboricity (resp. thickness) is no easier than cycle-freeness (resp. planarity) testing. Combining the space lower bounds of cycle-freeness and planarity testing shown in [35], we have Theorem 7.

Theorem 7. *In the semi-streaming model, any randomized algorithm that can approximate arboricity or thickness to within a factor of $(2-\varepsilon)$ for any constant $\varepsilon > 0$ with constant success rate has a space lower bound of $\Omega(n \log n)$ bits.*

We note that the above space lower bounds also hold in the turnstile model, because the turnstile model is a generalization of the semi-streaming model.

5 Applications

A low-degeneracy node ordering has many applications. Here we present how to use the ordering to partition a graph into edge-disjoint forests such that the number of forests approximates the minimum possible, i.e. the arboricity.

The Nash-William theorem [31] states that if m_S (resp. n_S) denotes the number of edges (resp. nodes) in the subgraph S, the arboricity can be stated as

$$\alpha(G) = \max_{S \subseteq G} \lceil m_S/(n_S - 1) \rceil,$$

which has a form similar to that of the density of the densest subgraph. The algorithm in [29] that approximates the density of the densest subgraph can be adapted to approximate arboricity to within a factor of $(1 + \varepsilon)$.

It is unclear how to exploit the actual value of the arboricity to actually partition the input graph into a small number of forests, however a d-degenerate node ordering has a direct application to such a partition. Considering that in the acyclic orientation induced by the ordering, the out-degree of each node is no more than d, and thus one can partition the edge set into d subgraphs, each of which contains the i-th out-going edge of each node, if any. Note that the subgraphs are forests because all of them have degeneracy 1 or, equivalently, are cycle-free due to Lemma 5.

The above procedure has a simple two-pass implementation in the semi-streaming model. We use the d-degenerate ordering obtained in the first pass to assign the orientation of the incoming edges in the second pass. Then, maintaining the out-degree of each node suffices to partition the edges as described.

Since the computed d-degenerate ordering has $d \leq (1+\varepsilon)d(G)$, and each of the degeneracy, arboricity and thickness approximates each other, we have the result in Theorem 8. Our result improves the approximation ratio by a factor of 2, compared to that in [12].

Theorem 8. *In the semi-streaming model, there exists an $\mathcal{O}(m)$-time 2-pass randomized algorithm that partition the edges into forests (resp. planar graphs) such that the number of forests (resp. planar graphs) approximates the minimum possible to within a factor of $(2+\varepsilon)$ (resp. $(6+\varepsilon)$) with probability $1 - 1/n^{\Omega(1)}$ using a space of $\mathcal{O}(\varepsilon^{-2} n \log^2 n)$ bits.*

References

1. Ahn, K.J., Guha, S.: Linear programming in the semi-streaming model with application to the maximum matching problem. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part II. LNCS, vol. 6756, pp. 526–538. Springer, Heidelberg (2011)
2. Alon, N., Yuster, R., Zwick, U.: Finding and counting given length cycles. Algorithmica **17**(3), 209–223 (1997)
3. Bar-Yossef, Z., Kumar, R., Sivakumar, D.: Reductions in streaming algorithms, with an application to counting triangles in graphs. In: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms (SODA), pp. 623–632. SIAM (2002)
4. Bhattacharya, S., Henzinger, M., Nanongkai, D., Tsourakakis, C.E.: Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In: Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC), pp. 173–182 (2015)
5. Bollobás, B.: Extremal Graph Theory. Academic Press, London (1978)
6. Bollobás, B.: The evolution of sparse graphs. In: Graph Theory and Combinatorics, Proceedings of the Cambridge Combinatorial Conference in honor of Paul Erdős, pp. 35–57. Academic Press (1984)
7. Charikar, M.: Greedy approximation algorithms for finding dense components in a graph. In: Jansen, K., Khuller, S. (eds.) APPROX 2000. LNCS, vol. 1913, pp. 84–95. Springer, Heidelberg (2000)
8. Chitnis, R.H., Cormode, G., Esfandiari, H., Hajiaghayi, M., McGregor, A., Monemizadeh, M., Vorotnikova, S.: Kernelization via sampling with applications to dynamic graph streams, CoRR abs/1505.01731 (2015)
9. Dean, A.M., Hutchinson, J.P., Scheinerman, E.R.: On the thickness and arboricity of a graph. J. Comb. Theor. Series B **52**(1), 147–151 (1991)

10. Dvořák, Z.: Constant-factor approximation of the domination number in sparse graphs. Eur. J. Comb. **34**(5), 833–840 (2013)
11. Farach-Colton, M., Hsu, T., Li, M., Tsai, M.-T.: Finding articulation points of large graphs in linear time. In: Dehne, F., Sack, J.-R., Stege, U. (eds.) WADS 2015. LNCS, vol. 9214, pp. 363–372. Springer, Heidelberg (2015)
12. Farach-Colton, M., Tsai, M.-T.: Computing the degeneracy of large graphs. In: Pardo, A., Viola, A. (eds.) LATIN 2014. LNCS, vol. 8392, pp. 250–260. Springer, Heidelberg (2014)
13. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. Theor. Comput. Sci. **348**(2), 207–216 (2005)
14. Frank, A., Gyarfas, A.: How to orient the edges of a graph. In: Proceedings of the Fifth Hungarian Colloquium on Combinatorics. vol. I, Combinatorics, pp. 353–364 (1976)
15. Gabow, H., Westermann, H.: Forests, frames, and games: algorithms for matroid sums and applications. In: Proceedings of the twentieth annual ACM Symposium on Theory of Computing (STOC), pp. 407–421. ACM (1988)
16. Goldberg, A.V.: Finding a maximum density subgraph. Technical report (1984)
17. Guha, S., McGregor, A., Tench, D.: Vertex and hyperedge connectivity in dynamic graph streams. In: Proceedings of the 34th ACM Symposium on Principles of Database Systems (PODS), pp. 241–247 (2015)
18. Guruswami, V., Onak, K.: Superlinear lower bounds for multipass graph processing. In: 28th Conference on Computational Complexity (CCC), pp. 287–298. IEEE (2013)
19. Jowhari, H., Sağlam, M., Tardos, G.: Tight bounds for L_p samplers, finding duplicates in streams, and related problems. In: Proceedings of the 30th ACM Symposium on Principles of Database Systems (PODS), pp. 49–58. ACM (2011)
20. Kapralov, M., Lee, Y.T., Musco, C., Musco, C., Sidford, A.: Single pass spectral sparsification in dynamic streams. In: 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS), pp. 561–570 (2014)
21. Kapralov, M., Woodruff, D.P.: Spanners and sparsifiers in dynamic streams. In: ACM Symposium on Principles of Distributed Computing (PODC), pp. 272–281 (2014)
22. Kawano, S., Yamazaki, K.: Worst case analysis of a greedy algorithm for graph thickness. Inf. Process. Lett. **85**(6), 333–337 (2003)
23. Konrad, C.: Maximum matching in turnstile streams. In: Bansal, N., Finocchi, I. (eds.) ESA 2015. LNCS, vol. 9294, pp. 840–852. Springer, Verlag (2015)
24. Kowalik, Ł.: Approximation scheme for lowest outdegree orientation and graph density measures. In: Asano, T. (ed.) ISAAC 2006. LNCS, vol. 4288, pp. 557–566. Springer, Heidelberg (2006)
25. Kutzkov, K., Pagh, R.: Triangle counting in dynamic graph streams. In: Ravi, R., Gørtz, I.L. (eds.) SWAT 2014. LNCS, vol. 8503, pp. 306–318. Springer, Heidelberg (2014)
26. Lenzen, C., Wattenhofer, R.: Minimum dominating set approximation in graphs of bounded arboricity. In: Lynch, N.A., Shvartsman, A.A. (eds.) DISC 2010. LNCS, vol. 6343, pp. 510–524. Springer, Heidelberg (2010)
27. Mansfield, A.: Determining the thickness of graphs is NP-hard. Math. Proc. Cambridge Philos. Soc. **93**, 9–23 (1983)
28. Matula, D.W., Beck, L.L.: Smallest-last ordering and clustering and graph coloring algorithms. J. ACM **30**(3), 417–427 (1983)

29. McGregor, A., Tench, D., Vorotnikova, S., Vu, H.T.: Densest subgraph in dynamic graph streams. In: Italiano, G.F., Pighizzini, G., Sannella, D.T. (eds.) MFCS 2015. LNCS, vol. 9235, pp. 472–482. Springer, Heidelberg (2015)
30. Muthukrishnan, S.: Data streams: algorithms and applications. Technical report (2003)
31. Nash-Williams, C.S.A.: Edge-disjoint spanning trees of finite graphs. J. Lond. Math. Soc. **s1–36**(1), 445–450 (1961)
32. O'Connell, T.C.: A survey of graph algorithms under extended streaming models of computation. In: Ravi, S.S., Shukla, S.K. (eds.) Fundamental Problems in Computing, pp. 455–476. Springer, The Netherlands (2009)
33. Ruhl, J.M.: Efficient algorithms for new computational models. Ph.D. thesis, Massachusetts Institute of Technology, September 2003
34. Schank, T., Wagner, D.: Finding, counting and listing all triangles in large graphs, an experimental study. In: Nikoletseas, S.E. (ed.) WEA 2005. LNCS, vol. 3503, pp. 606–609. Springer, Heidelberg (2005)
35. Sun, X., Woodruff, D.P.: Tight bounds for graph problems in insertion streams. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM, pp. 435–448 (2015)