On the Locality of Nash-Williams Forest Decomposition and Star-Forest Decomposition

David G. Harris
University of Maryland
College Park, Maryland, USA
davidgharris29@gmail.com

Hsin-Hao Su
Boston College
Chestnut Hill, Massachusetts, USA
suhx@bc.edu

Hoa T. Vu
San Diego State University
San Diego, California, USA
hvu2@sdsu.edu

ABSTRACT

Given a graph $G = (V, E)$ with arboricity a, we study the problem of decomposing the edges of G into $(1+\varepsilon)a$ disjoint forests in the distributed LOCAL model. While there is a polynomial time centralized algorithm for a-forest decomposition (e.g. [Imai, J. Operation Research Soc. of Japan '83]), it remains an open question how close we can get to this exact decomposition in the LOCAL model.

Barenboim and Elkin [PODC '08] developed a LOCAL algorithm to compute a $(2+\varepsilon)a$-forest decomposition in $O(\frac{\log n}{\varepsilon})$ rounds. Ghaffari and Su [SODA '17] made further progress by computing a $(1+\varepsilon)a$-forest decomposition in $O(\frac{\log^3 n}{\varepsilon^4})$ rounds when $\varepsilon a = \Omega(\sqrt{a \log n})$, i.e., the limit of their algorithm is an $(a+\Omega(\sqrt{a \log n}))$-forest decomposition. This algorithm, based on a combinatorial construction of Alon, McDiarmid & Reed [Combinatorica '92], in fact provides a decomposition of the graph into *star-forests*, i.e., each forest is a collection of stars.

Our main goal is to reduce the threshold of εa in $(1+\varepsilon)a$-forest decomposition. We obtain the following main results:

- An $O(\frac{\Delta^\rho \log^4 n}{\varepsilon})$-round algorithm when $\varepsilon a = \Omega_\rho(1)$ in simple graphs and multigraphs, where $\rho > 0$ is any arbitrary constant.
- An $O(\frac{\log^4 n \log \Delta}{\varepsilon})$-round algorithm when $\varepsilon a = \Omega(\frac{\log \Delta}{\log \log \Delta})$ in simple graphs and multigraphs.
- An $O(\frac{\log^4 n}{\varepsilon})$-round algorithm when $\varepsilon a = \Omega(\log n)$ in simple graphs and multigraphs. This also covers an extension of the forest-decomposition problem to list-edge-coloring.
- An $O(\frac{\log^3 n}{\varepsilon})$-round algorithm for star-forest decomposition for $\varepsilon a = \Omega(\sqrt{\log \Delta} + \log a)$ in simple graphs. When $\varepsilon a \geq \Omega(\log \Delta)$, this also covers a list-edge-coloring variant.

Our techniques also give an algorithm for $(1+\varepsilon)a$-outdegree-orientation in $O(\log^3 n/\varepsilon)$ rounds, which is the first algorithm with *linear dependency* on ε^{-1}.

At a high level, the first three results come from a combination of network decomposition, load balancing, and a new structural result on local augmenting sequences. The fourth result uses a more careful probabilistic analysis for the construction of Alon, McDiarmid & Reed; these bounds on star-forest-decomposition were not previously known to be possible, even non-constructively.

PODC '21, July 26–30, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8548-0/21/07…$15.00
https://doi.org/10.1145/3465084.3467908

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**; **Graph algorithms analysis**; • **Mathematics of computing → Graph theory**.

KEYWORDS

forest decompositions, local algorithms, graph algorithms

ACM Reference Format:
David G. Harris, Hsin-Hao Su, and Hoa T. Vu. 2021. On the Locality of Nash-Williams Forest Decomposition and Star-Forest Decomposition. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC '21), July 26–30, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3465084.3467908

1 INTRODUCTION

Consider a loopless (multi-)graph $G = (V, E)$ with $n = |V|$ vertices, $m = |E|$ edges, and maximum degree Δ. A k-forest decomposition (abbreviated k-FD) is a partition of the edges into k forests. The *arboricity* of G, denoted $a(G)$, is a measure of sparsity defined as the minimum number k for which a k-forest decomposition of G exists. We also write $a(E)$ or just a when G is understood. An elegant result of Nash-Williams [45] shows that $a(G)$ is given by the formula:

$$a(G) = \max_{\substack{H \subseteq G \\ |V(H)| \geq 2}} \left\lceil \frac{|E(H)|}{|V(H)| - 1} \right\rceil.$$

Note that the RHS is clearly a lower bound on a since each forest can only consume at most $|V(H)| - 1$ edges in a subgraph H. In the centralized setting, a series of polynomial-time algorithms have been developed for a-forest decomposition [23, 24, 37, 49].

In this work, we study the problem of computing forest decompositions in the LOCAL model of distributed computing [42]. In this model, the vertices operate in synchronized rounds where each vertex sends and receives messages of arbitrary size to its neighbors, and performs arbitrary local computations. The complexity of an algorithm is defined to be the number of rounds used. An r-round LOCAL algorithm implies that each vertex only uses information in its r-hop neighborhood to compute the answer, and vice versa.

There has been growing interest in investigating the gap between what can be computed efficiently and the existential bounds of various combinatorial structures. For example, consider the problem of proper edge coloring. The classical result of Vizing [57] shows that a $(\Delta+1)$-edge-coloring exists in simple graphs. A long series

of works have developed efficient LOCAL algorithms using smaller number of colors [15, 18, 20, 29, 46, 56]. This culminated with [12], which matched the optimal existential bound with a poly$(\Delta, \log n)$ algorithm for $(\Delta+1)$-edge-coloring. Forest decomposition can be viewed as a variant of proper edge coloring: in the latter problem, the edges induced by each color should form a matching, while in the former they should form a forest. Similar to edge-coloring, distributed forest decomposition has applications to scheduling of radio networks and wireless networks [33, 48], where a smaller number of forests corresponds to a more compact schedule.

In contrast with edge-coloring, computing an a-FD in the LOCAL model requires $\Omega(n)$ rounds even in simple graphs of constant maximum degree (see Proposition 6.3). We thus develop algorithms using $(1+\varepsilon)a$ forests, i.e. εa excess forests beyond the a forests required existentially. Beside round complexity, a key objective is the number of excess forests used, i.e. what is the minimum range of ε in which the algorithms can operate.

The forest decomposition problem was first studied in the LOCAL model by Barenboim and Elkin [7], who developed the H-partition algorithm to compute a $(2+\varepsilon)a$-forest decomposition in $O(\log n/\varepsilon)$ rounds. This has been a building block in many other distributed and parallel algorithms [7, 8, 10, 41, 54]. Barenboim and Elkin [7] also showed that an $O(a)$-FD would require $\Omega(\frac{\log n}{\log a} - \log^* n)$ rounds. In [9, Open Problem 11.10], they posed the problem of using fewer than $2a$ forests. Some progress was made by Ghaffari and Su [31] with an algorithm for $(1+\varepsilon)a$-forest decomposition in $O(\log^3 n/\varepsilon^4)$ rounds when $\varepsilon = \Omega(\sqrt{\log n/a})$ for simple graphs, i.e., the minimum number of obtainable forests is $a + \Omega(\sqrt{a \log n})$. A natural question is how far this threshold can be pushed down.

We make further progress on this objective with an algorithm for $(a+3)$-FD in poly$(\Delta, \log n)$ rounds in multigraphs. The polynomial dependence on Δ can be removed when εa is larger; for example, we obtain a $(1+\varepsilon)a$-FD in $O(1/\varepsilon) \cdot \text{polylog}(n)$ rounds for $\varepsilon a = \Omega(\log\Delta/\log\log\Delta)$.

Furthermore, we consider two important extensions to this basic framework.

List Forest Decomposition: Similar to edge coloring, there is a list version of the forest decomposition problem. Each edge e has a palette of colors $Q(e)$ from a color-space C, and we should choose a color $\phi(e) \in Q(e)$ so that, for any color c, the subgraph induced by the c-colored edges forms a forest. If $|Q(e)| \geq k$ for e, we refer to this as k-list-forest decomposition (abbreviated k-LFD).

This generalizes k-forest-decomposition, which can be viewed as the case where $Q(e) = C = \{1, \ldots, k\}$ for all edges e. The total number of forests (one for each color) may be much larger than a; in this case, the excess is measured in terms of the number of extra colors in edges' palettes (in addition to the a colors that are required by the lower bound).

Based on general matroid arguments, Seymour [53] showed that an a-LFD exists for any choice of palettes. This also can be turned into a polynomial-time centralized algorithm with standard matroid routines. Thus, from the point of view of combinatorial constructions and centralized algorithms, forest-decomposition and list-forest-decomposition are nearly equivalent. In the LOCAL model, though, list-forest-decomposition appears to be much harder. Unlike forest decomposition, it is not sufficient to color most of the edges with a small left-over uncolored component.

We give poly$(\log n, 1/\varepsilon)$-round algorithms for $(1+\varepsilon)a$-LFD when $\varepsilon a = \Omega(\min\{\log n, \sqrt{a \log\Delta}\})$. A key open problem is to find an efficient algorithm for $\varepsilon a \geq \Omega(1)$.

Low-Diameter and Star-Forest Decompositions: The second extension is to ensure that the forests in the decomposition have low diameter. We say that the decomposition has *diameter* D if every tree in every c-colored forest has strong diameter at most D. Low-diameter forest decompositions are interesting from both practical and theoretical aspects. For example, a k-FD of diameter D can be turned into k rooted forests in $O(D)$ rounds.

We develop a few post-processing techniques to convert an arbitrary k-FD into a $(1+\varepsilon)k$-FD with diameter $O(\log n/\varepsilon)$. When εk is large enough, the diameter can be reduced further to $O(1/\varepsilon)$, which is optimal for multigraphs (see the full paper for details).

In the most extreme case, each forest should be a collection of stars, i.e., a *star-forest*. We refer to the problem of decomposing the graph into k star-forests as *k-star-forest decomposition* (abbreviated k-SFD), and we call the list-coloring variant of star-forest decomposition *k-list-star-forest-decomposition* (abbreviated k-LSFD). This problem has received some attention in combinatorics. In simple graphs, we give an $O(\frac{\log^3 n}{\varepsilon})$-round algorithm for $(1+\varepsilon)a$-SFD when $\varepsilon a = \Omega(\log a + \sqrt{\log\Delta})$ as well as an $O(\frac{\log^3 n}{\varepsilon})$-round algorithm for $(1+\varepsilon)a$-LSFD when $\varepsilon a = \Omega(\log\Delta)$.

1.1 Summary of Results

Our results for forest decomposition balance a number of measures: the minimum number of obtainable forests, the running time, the tree diameters, LFD versus FD, and multigraphs versus simple graphs. See Table 1 for a summary of a some possible parameters. Here, $\rho > 0$ represents any desired constant and we use Ω_ρ to represent a constant terms which may depend on ρ.

Excess colors	Lists?	Multi-graph?	Runtime	Diameter
3	No	Yes	$\tilde{O}(\Delta^2 a \log^4 n)$	$\leq n$
≥ 4	No	Yes	$\tilde{O}(\Delta^2 \log^4 n/\varepsilon)$	$O(\log n/\varepsilon)$
$\Omega_\rho(1)$	No	Yes	$O(\Delta^\rho \log^4 n/\varepsilon)$	$O(\log n/\varepsilon)$
$\geq 4 + \rho\log\Delta$	No	Yes	$O_\rho(\log^4 n/\varepsilon)$	$O(\log n/\varepsilon)$
$\Omega(\sqrt{a\log\Delta})$	No	Yes	$O(\log^4 n/\varepsilon)$	$O(1/\varepsilon)$
$\Omega(\log n)$	No	Yes	$O(\log^3 n/\varepsilon)$	$O(1/\varepsilon)$
$\Omega(\sqrt{a\log\Delta})$	Yes	Yes	$O(\log^4 n/\varepsilon^2)$	$O(\log n/\varepsilon^2)$
$\Omega(\log n)$	Yes	Yes	$O(\log^4 n/\varepsilon)$	$O(\log n/\varepsilon)$
$\Omega(\sqrt{\log\Delta} + \log a)$	No	No	$O(\log^3 n/\varepsilon)$	2 (star)
$\Omega(\log\Delta)$	Yes	No	$O(\log^3 n/\varepsilon)$	2 (star)

Table 1: Algorithms for $(1+\varepsilon)a$-FD and $(1+\varepsilon)a$-LFD of G

Thus, for instance, the final listed algorithm requires excess $\overline{K}\log\Delta$ and the third listed algorithm requires excess $\overline{K}_\rho$, where $\overline{K}$ and $\overline{K}_\rho$ are universal constants.

All these algorithms here are randomized and succeed *with high probability* (abbreviated w.h.p.), i.e. succeed with probability at least

$1 - 1/\text{poly}(n)$. Since all the failure modes can be locally checked during the run of the algorithms, they can be derandomized with an additional polylog(n) factor in the runtime using the recent breakthrough of [26, 50]; we do not discuss any further issues of determinization henceforth.

We also show that $\Omega(1/\varepsilon)$ rounds are needed for $(1+\varepsilon)a$-FD in multigraphs (See Theorem 6.2).

1.2 Summary: Distributed Augmentation

The results for forest-decomposition and list-forest-decomposition are based on a distributed implementation of *augmenting paths*, in which we color one *uncolored* edge and possibly change some of the colored edges while maintaining the solution feasibility. Augmentation approaches have been used for many combinatorial constructions, such as coloring and matching.

Gabow and Westermann [24] described an approach for a-forest-decomposition via augmenting paths. Roughly speaking, this works as follows: Given an uncolored edge e_1, suppose that we try to assign it color c_1. If it does not create a cycle, then we are done. Otherwise, if a cycle is created, we recolor some edge e_2 on the cycle with different color $c_2 \neq c_1$. Continuing this process leads to an *augmenting sequence* $e_1, c_1, e_2, c_2 \ldots, e_\ell, c_\ell$, such that recoloring e_ℓ in c_ℓ does not create a cycle. Furthermore, a BFS algorithm can be used to find such an augmenting sequence efficiently in the centralized setting.

This approach faces two main challenges in the LOCAL model. First, in order to get a distributed algorithm, we must color many edges in parallel. Second, it is unclear if the recoloring can be done in the vicinity of the initial uncolored edge. Note that the augmenting sequences produced by the Gabow and Westerman algorithm can be long, and even if a short augmenting sequence exists, the consecutive edges in the sequence (e.g. e_1 and e_2) can be arbitrarily far from each other.

Structural Results on Augmenting Sequences: We first show a structural result on forest decomposition: given a partial $(1+\varepsilon)a$-FD (or, more generally, a partial $(1+\varepsilon)a$-LFD) in a multigraph, there is an augmenting sequence of length $O(\log n/\varepsilon)$ where, moreover, every edge in the sequence lies in the $O(\log n/\varepsilon)$-neighborhood of the starting uncolored edge. We show this through a key modification to the BFS algorithm for finding an augmenting sequence. In [24], when assigning e_i to color c_i creates a cycle, then all edges on the cycle get enqueued for the next layer; by contrast, in our algorithm, we only enqueue the edges adjacent to e_i itself. We show even with this modification, an augmenting sequence appears after $O(\log n/\varepsilon)$ steps. This characterization may potentially lead to other algorithms for forest decompositions.

Network Decomposition and Load Balancing Cut Edges: To apply augmenting paths in parallel, we will use network decomposition methods similar to [28] to break the graph into smaller subgraphs. However, we encounter a major roadblock to doing this directly: *identifying* an augmenting sequence may still require checking edges distant from the uncolored edge. For example, it is not clear how to tell if edge e_1 belongs to a color-c_1 cycle in the LOCAL model as the cycle may extend outside the vicinity of e_1.

To handle this, we develop a procedure CUT which removes edges to break long paths, thereby allowing augmenting sequences to be locally checkable. To get a $(1+\varepsilon)a$-forest decomposition of the full graph, we must ensure that the collection of edges removed by CUT (the "left-over graph") has arboricity $O(\varepsilon a)$. It suffices to bound the number of incident edges removed per vertex; this reduces to an online load-balancing problem similar to one encountered in [56], where here the load of a vertex is the number of its directed neighbors which are removed. In [56], paths come in an online fashion and we should remove some of their edges so that every vertex has small load at the end. Here, rooted trees come in an online fashion, and we need to remove some edges to disconnect the root from all the leaves.

If edges are removed independently with some uniform probability, then the number of incident edges per vertex would be stuck at $\Omega(\log n)$ due to the concentration threshold. To break this barrier, we generalize the conditioned sampling approach of [56]. The rough idea is that we randomly remove edges which are incident to vertices with small loads. We will show that throughout the algorithm, the root-leaf paths of the trees always contain a large number of these vertices; thus, it is likely that none of the long paths survive in any iteration.

Palette Partitioning for List-Coloring: The final step is to recolor the left-over edges using an additional $O(\varepsilon a)$ colors. For ordinary forest decomposition, this is nearly automatic due to our bound on the arboricity of the left-over graph. For list coloring, we must reserve a small number of back-up colors for the left-over edges. We develop two different methods for this; the first uses the Lovász Local Lemma and the second uses randomized network decomposition.

There are some additional connections in our work to two closely related graph parameters, *pseudo-arboricity* and *star-arboricity*. Let us summarize these next.

1.3 Pseudo-Forest Decomposition and Low Outdegree Orientation

There is a closely related decomposition using *pseudo-forests*, which are graphs with at most one cycle in each connected component. The *pseudo-arboricity* a^* is the minimum number of pseudo-forests into which a graph can be decomposed. A k-pseudo-forest decomposition is equivalent to an edge-orientation where every vertex has outdegree at most k. This characterization, which we call an *k-orientation*, is completely local.

A result of Hakimi [35] shows that pseudo-arboricity is given by an analogous formula to Nash-Williams' formula for arboricity. Loopless multigraphs have $a^* \leq a \leq 2a^*$, and simple graphs also satisfy $a \leq a^* + 1$. In some sense, a^* is a more fundamental graph parameter than a, and the problems of pseudo-forest decomposition, low outdegree orientation, and maximum density subgraph are more well-understood than forest decomposition. For example, the maximum density subgraph problem has been studied in many computational models, e.g. [5, 6, 13, 16, 21, 25, 30, 32, 38, 44, 47, 51, 52]. The low outdegree orientation problem has been studied in the centralized context in [11, 14, 24, 34, 39, 40].

There has been a long line of work on LOCAL algorithms for $(1+\varepsilon)a^*$-orientation [22, 26, 31, 36, 55]. Most recently, [55] gave an algorithm running in $\tilde{O}(\log^2 n/\varepsilon^2)$ rounds for $\varepsilon a^* \geq 32$; this algorithm also applies to the CONGEST model, which is a special

case of the LOCAL model where messages are restricted to $O(\log n)$ bits per round. However, none of the algorithms can achieve better than $\Omega(1/\varepsilon^2)$ dependencies on ε. Notably, the $1/\varepsilon^2$ factor in [55] comes from the number of iterations needed to solve the LP.

Our general strategy of augmenting paths and network decompositions can also be used for low outdegree orientations. We will show the following result:

Theorem 1.1. *For a (multi)-graph G with pseudo-arboricity a^* and $\varepsilon \in (0,1)$, there is a* LOCAL *algorithm to obtain $\lceil a^*(1+\varepsilon)\rceil$-orientation in $O(\frac{\log^3 n}{\varepsilon})$ rounds w.h.p.*

Note in particular the linear dependency on $1/\varepsilon$. Theorem 1.1 provides a simple warm-up exercise for our more advanced forest-decomposition algorithms.

1.4 Star-Arboricity and (List)-Star-Forest Decomposition for Simple Graphs

The *star-arboricity* a_{star} is the minimum number of star-forests into which the edges of a graph can be partitioned. This has been studied in combinatorics [1–3], where the main focus is to (non-constructively) bound a_{star} in terms of other graph parameters. We analogously define the *list star-arboricity* of a graph, $a^{\text{list}}_{\text{star}}$; namely, the smallest value k such that there is a LSFD whenever each edge has a palette of size k.

For general loopless multigraphs, it can be shown that $a_{\text{star}} \le 2a^*$ and $a^{\text{list}}_{\text{star}} \le 4a^*$. In simple graphs, Alon, McDiarmid & Reed [2] showed that $a_{\text{star}} \le a + O(\log\Delta)$. Our results for star-forest-decomposition in simple graphs come from strengthening this [2] construction, as well as making it algorithmic.

To briefly summarize, the idea is to start with a k-orientation where $k = (1+O(\varepsilon))a$. A subset of the vertices get marked as c-centers for each color c, otherwise they are c-leaves. Each vertex v gets marked as a c-center independently with some probability p. Finding the star-forest decomposition then reduces to finding a perfect matching, for each vertex u, between the colors c for which u is a c-leaf, and the neighbors v of u which are c-centers.

In the general LSFD case, we show the existence of this perfect matching for each vertex u when $\varepsilon k = \Omega(\log\Delta)$. This is based on more advanced analysis of concentration bounds (beyond Chernoff bounds) for the number of c-leaf neighbors. In the ordinary SFD case, instead of a perfect matching, we obtain a near-perfect matching, leaving εk unmatched edges per vertex. These left-over edges can be later decomposed into $2\varepsilon k$ stars. This allows us to strengthen the bound to $\varepsilon k = \Omega(\sqrt{\log\Delta} + \log a)$.

In addition to being powerful algorithmic results, these also show two new combinatorial bounds:

Corollary 1.2. *A simple graph has $a_{star} \le a + O(\log a + \sqrt{\log\Delta})$ and $a^{list}_{star} \le a + O(\log\Delta)$.*

For lower bounds, [2] showed that there are simple graphs with $a_{\text{star}} = 2a$ and $\Delta = 2^{\Theta(a^2)}$, while [1] showed that there are simple graphs where every vertex has degree $d = 2a$ and where $a_{\text{star}} \ge a + \Omega(\log a)$. These two lower bounds show that the dependence of a_{star} on a and Δ are nearly optimal in Corollary 1.2. In particular, the term $\log a$ cannot be replaced by a function $o(\log a)$ and the term $\sqrt{\log\Delta}$ cannot be replaced by a function $o(\sqrt{\log\Delta})$.

1.5 Preliminaries

The values ε and a are global parameters, along with related parameters such as m, n, Δ, a^*. As is usual in distributed algorithms, we suppose throughout that we are given some globally-known upper bounds on these values; when we write a, n etc. we are technically referring to input values $\hat{a}, \hat{n}$ etc. which are upper bounds on them. Almost all of our results become vacuous if $\varepsilon < 1/n$ (since, in the LOCAL model, we can simply read in the entire graph in $O(n)$ rounds), so we assume throughout that $\varepsilon \in (1/n, 1/2)$.

We define the r-neighborhood of a vertex v, denoted $N^r(v)$, to be the set of vertices within distance r of v. We likewise write $N^r(e)$ for an edge e and $N^r(X)$ for a set X of vertices or edges. For any vertex set X, we define $E(X)$ to be the set of induced edges on X. We define the power-graph G^r to be the graph on vertex set V and with an edge uv if u, v have distance at most r in G. Note that, in the LOCAL model, G^r can be simulated in $O(r)$ rounds of G.

For any integer $t \ge 0$, we define $[t] = \{1, \dots, t\}$. We write $A = B \sqcup C$ for a disjoint union, i.e. $A = B \cup C$ and $B \cap C = \emptyset$.

Network Decomposition: A (D, χ)-network decomposition of G is a partition of vertices into χ classes such that every connected component in every class has strong diameter at most D. We refer to each connected component within each class a *cluster*. There are known randomized LOCAL algorithms [4, 19, 43] to compute an $(O(\log n), O(\log n))$-network decomposition in $O(\log^2 n)$ rounds.

1.6 Basic Forest Decomposition Algorithms

There are a few simpler forest decomposition algorithms that will be important building blocks for our more advanced algorithms. See the full paper for proofs.

Theorem 1.3. *Let $t = \lfloor(2+\varepsilon)a^*\rfloor$. There are deterministic $O(\frac{\log n}{\varepsilon})$-round algorithms for the following decompositions of G:*

- *A partition of the vertices of G into $k = O(\frac{\log n}{\varepsilon})$ classes $H_1, \dots, H_k$, such that each vertex $v \in H_i$ has at most t neighbors in $H_i \cup \dots \cup H_k$.*
- *An orientation of the edges of G such that the resulting directed graph is acyclic and each vertex has outdegree at most t. We refer to this as an* acyclic t-orientation.
- *A $3t$-star-forest-decomposition of G.*
- *A t-list-forest-decomposition of G.*

Theorem 1.4. *For a multigraph G, there is a randomized algorithm for a $\lfloor(4+\varepsilon)a^*\rfloor$-LSFD in $\min\left(O(\frac{\log^3 n}{\varepsilon}), \tilde{O}(\frac{\log n \log^* m}{\varepsilon})\right)$ rounds w.h.p.*

Finally, we consider how to reduce the diameter of a given forest decomposition. This is often used in our algorithms, where we first obtain some k-FD of G, with unbounded diameter and then relax it to a $k(1+\varepsilon)$-FD with small diameter.

Proposition 1.5. *Let G be a multigraph with a k-FD. For any $\varepsilon > 0$, there is an $O(\frac{\log n}{\varepsilon})$-round to compute a $(k + \lceil\varepsilon a\rceil)$-FD of G of diameter $D \le O(\frac{\log n}{\varepsilon})$ w.h.p. If $a \ge \Omega\big(\min\{\frac{\log n}{\varepsilon}, \frac{\log\Delta}{\varepsilon^2}\}\big)$, we can get $D \le O(\frac{1}{\varepsilon})$ w.h.p. with the same runtime.*

2 ALGORITHM FOR LOW OUTDEGREE ORIENTATION

We now discuss a LOCAL algorithm for $(1+\varepsilon)a^*$-orientation, based on augmenting sequences and network decomposition. In addition to being a notable result in its own right, it serves as a good warm-up exercise to illustrate some of the main ideas of our later forest-decomposition algorithms.

Consider a multigraph G of pseudo-arboricity a^*. **For the purposes of this section only, we allow G to contain loops.** Following [31], we can "augment" a given edge-orientation ψ by reversing the edges in some directed path. For a given parameter $\varepsilon > 0$, let us say that a vertex v is *overloaded* with respect to a given orientation ψ, if the outdegree of v is strictly larger than $\lceil a^*(1+\varepsilon)\rceil$; otherwise, if the outdegree is at most $\lceil a^*(1+\varepsilon)\rceil$, it is *underloaded*.

We begin with the following observation, which is essentially a restatement of [31, 35] with more careful counting of parameters.

Lemma 2.1. *Let $\varepsilon \in (0,1)$. For a given edge-orientation ψ and any vertex $v \in G$, there is a directed path of length $O(\frac{\log n}{\varepsilon})$ from v to a vertex with outdegree strictly less than $a^*(1+\varepsilon)$.*

For the purposes of our algorithm, the main significance of this result is that it allows us to locally fix a given edge-orientation. We remark that this type of "local patching" result has been critical for other LOCAL algorithms, such as the Δ-vertex-coloring algorithm of [27] or the $(\Delta+1)$-edge-coloring algorithm of [12]. We summarize this as follows:

Proposition 2.2. *Suppose multigraph G has an edge-orientation ψ, and let $L \subseteq V$ be an arbitrary vertex set. Then there is an edge-orientation ψ' with the following properties:*

- *ψ' agrees with ψ outside $N^r(L)$ where $r = O(\frac{\log n}{\varepsilon})$.*
- *All vertices of L are underloaded with respect to ψ'.*
- *All vertices which are underloaded with respect to ψ remain underloaded with respect to ψ'.*

Proof. Following [31], we consider the following process: while some vertex of L is overloaded, we choose any arbitrary such vertex $v \in L$. We then find some directed path $v, v_1, \ldots, v_r$ from v where $r \le O(\log n/\varepsilon)$ and where vertex v_r has outdegree strictly less than $a^*(1+\varepsilon)$. Next, reverse the orientation of all edges along this path. This does not change the outdegree of the vertices $v_1, \ldots, v_{r-1}$, while it decreases the outdegree of v by one and increases the outdegree of v_r by one. □

We next use network decomposition to extend this local patching into a global solution, via the following Algorithm 1. Here, K is a universal constant to be specified.

Algorithm 1 Low-degree_Orientation_Decomposition(G)

1: Initialize ψ to be some arbitrary orientation of G.
2: Compute an $(O(\log n), O(\log n))$-network decomposition in G^{2R} for $R = \lceil K \log n/\varepsilon \rceil$.
3: **for each** class z in the network decomposition **do**
4: **for** each component C in the class z **in parallel do**
5: Modify ψ so that vertices inside C become underloaded, vertices outside $N^R(C)$ are unchanged, and no new overloaded vertices are created.

We remark that Algorithm 1 can be viewed as part of a family of algorithms based on network decomposition described in [28]. (In the language of [28], the algorithm can be implemented in SLOCAL with radius $r = O(\frac{\log n}{\varepsilon})$.) However, we describe the algorithm explicitly to keep this paper self-contained, and because we later need a more general version of Algorithm 1.

Theorem 2.3. *Algorithm 1 can be implemented in $O(\log^3 n/\varepsilon)$ rounds. At the termination, ψ is an edge-orientation with maximum outdegree $\lceil a^*(1+\varepsilon)\rceil$ w.h.p.*

Proof. For the first step, we use the algorithm of [19] to obtain the network decomposition for G^{2R} in $O(R\log^2 n)$ rounds. Algorithm 1 processes each cluster C of a given class simultaneously and tries to orient the edges that are adjacent to or inside C. We also define $C' = N^R(C)$. From Proposition 2.2, we know that it is possible to modify ψ within C' for sufficiently large K, such that all vertices in C become underloaded, and no additional overloaded vertices are created.

For any putative orientation ψ' which does not modify edges outside C', we can check if ψ' satisfies these properties by looking locally within $N^R(C)$. The distance between two clusters in the same class is at least $2R+1$. Moreover, if u, v are adjacent in G^{2R}, their distance is at most $2R$. So each cluster C has weak diameter at most $O(R\log n)$, and also the balls C'_1 and C'_2 must be disjoint for any two clusters C_1 and C_2 of the same class. So each cluster can be processed independently without interfering with others. Therefore, each iteration can be simulated locally in $O(R\log n)$ rounds. Since there are $O(\log n)$ classes, the total running time is $O(R\log^2 n) = O(\log^3 n/\varepsilon)$. □

This shows Theorem 1.1. We will use the same overall strategy for forest decomposition, but we will encounter two main technical obstacles. First, we must define an appropriate notion of local patching and augmenting sequences; this will be far more complex than Proposition 2.2. Second, and more seriously, forest-decomposition, unlike low-degree orientation, cannot be locally checked as a given (partial) forest decomposition may have long cycles. To circumvent this, we must remove edges at each step to destroy these cycles. These left-over edges will need some post-processing steps to handle them at the end.

3 AUGMENTING SEQUENCES FOR LIST-COLORING

We now show our main structural result on the existence of augmenting sequences in a multigraph. Given a partial LFD ψ of G and an edge $e = uv$, we define $C(e,c)$ to be the unique u-v path in the c-colored forest; if u and v are disconnected in the color-c forest then we write $C(e,c) = \emptyset$. We write $\psi(e) = \emptyset$ if an edge e is uncolored.

We define an *augmenting sequence* w.r.t. ψ to be a sequence $P = (e_1, e_2, \ldots, e_\ell, c)$, for edges e_i and color c, satisfying the following five conditions:

(A1) $\psi(e_1) = \emptyset$.

(A2) $e_i \in C(e_{i-1}, \psi(e_i))$ for $2 \le i \le \ell$.

(A3) $e_i \notin C(e_j, \psi(e_i))$ for every i and j such that $j < i-1$.

(A4) $C(e_\ell, c) = \emptyset$.

(A5) $\psi(e_{i+1}) \in Q(e_i)$ for each $i = 1, \ldots, \ell - 1$ and $c \in Q(e_\ell)$.

Recall that $Q(e)$ denotes the list of available colors for edge e. We say that ℓ is the *length* of the sequence. We define the *augmentation* $\psi \oplus P$ to be a new (partial) coloring obtained by setting $\psi'(e_i) = \psi(e_{i+1})$ for $1 \le i \le \ell - 1$ and $\psi'(e_\ell) = c$, and $\psi'(e) = \psi(e)$ for all other edges $e \in E \setminus \{e_1, \ldots, e_\ell\}$. See Figure 1.

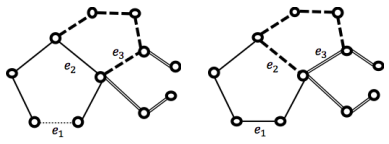


Figure 1: An illustration of an augmenting sequence before (left) and after (right) the augmentation process.

Lemma 3.1. *For an augmenting sequence P w.r.t ψ, the augmentation $\psi \oplus P$ remains a partial list-forest decomposition.*

With this definition, we will show the following main result:

Theorem 3.2. *Given a partial $(1+\varepsilon)a$-LFD of a multigraph G and an uncolored edge e, there is an augmenting sequence $P = (e, e_2, \ldots, e_\ell, c)$ from e where $e_2, \ldots, e_\ell \in N^r(e)$ for $r = O(\frac{\log n}{\varepsilon})$.*

The main significance of Theorem 3.2 is that it allows us to locally fix a partial LFD, in the same way Proposition 2.2 allows us to locally fix an edge-orientation. We summarize this as follows:

Corollary 3.3. *Suppose multigraph G has a partial $(1+\varepsilon)a$-LFD ψ, and let $L \subseteq E$ be an arbitrary edge set. Then there is a partial LFD ψ' with the following properties:*

- *ψ' agrees with ψ outside $N^r(L)$ where $r = O(\frac{\log n}{\varepsilon})$.*
- *ψ' is a full coloring of the edges L.*
- *All edges colored in ψ are also colored in ψ'.*

Proof. We go through each uncolored edge $e \in L$ in an arbitrary order, and obtain an augmenting sequence P from Theorem 3.2, and then modify ψ to $\psi \leftarrow \psi \oplus P$. □

To prove Theorem 3.2, we first find a weaker object called an *almost augmenting sequence*, which is a sequence satisfying properties (A1), (A2), (A4), (A5) but not necessarily (A3). See Algorithm 2.

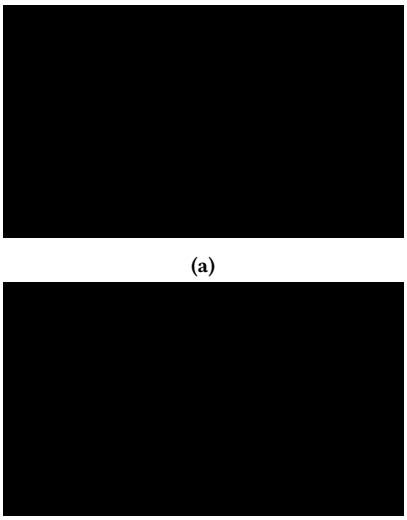
Algorithm 2 Find_Augmenting_sequence(e_{init})

```
E_1 = {e_init}
for i = 1 ... O(log n/ε) do
    E_{i+1} ← E_i.
    for each e ∈ E_i and each color c ∈ Q(e) do
        if C(e, c) ≠ ∅ then
            for each edge e' ∈ C(e, c) \ E_i adjacent to E_i do
                Set E_{i+1} ← E_{i+1} ∪ {e'} and π(e') ← e.
        else
            Return the sequence P = (e_init, ..., π(π(e)), π(e), e, c).
```

Lemma 3.4. *Algorithm 2 terminates with an almost augmenting sequence.*

Proof. In each iteration i, let V_i denote the endpoints of the edges in E_i, and let $E_{i,c}$ be the set of edges in E_i with color c under ψ. The sets $E_{i,c}$ partition E_i. An edge only gets added to E_{i+1} if it is adjacent to an edge in E_i. Thus, the graph spanned by E_i is connected and $E_i \subseteq N^{i-1}(e_{\text{init}})$.

Let us assume we are at some iteration i and $C(e, c) \neq \emptyset$ holds for all $e \in E_i$. For each color c, let $E_c^\star \subseteq E_i$ be the set of edges in E_i whose palette contains color c. Let n_c and $n_c^\star$ be the number of connected components in the subgraphs $G_c' = (V_i, E_{i,c})$ and $G_c^\star = (V_i, E_c^\star)$ respectively. Note that $E_{i,c} \subseteq E_c^\star$ and $n_c = |V_i| - |E_{i,c}|$ since G_c' is a forest. See Figure 2a.



(a)

(b)

Figure 2: In the first figure, the ovals represent components of G_c' while the larger rectangles represent components of $G_c^\star$. The second figure shows rooted forest H_c, where components of G_c' are contracted to nodes. Each contracted, non-root node has a parent edge (shown in bold) added to E_{i+1}.

For each color c, we can construct a rooted forest H_c from G_c' as follows (see Figure 2b): First, contract the edges of $E_{i,c}$; now H_c consists of n_c isolated vertices, which we call *contracted nodes*. Second, for every edge $e \in E_c^\star$, we add $C(e, c)$ to H_c (both vertices and edges). This path $C(e, c)$ connects the endpoints of e, and so the graph H_c now contains at most $n_c^\star$ components. Finally, choose an arbitrary rooting of the forest.

For every contracted, non-root node u', the parent edge of u' in H_c corresponds to some edge e of G. This edge is adjacent to exactly one vertex in V_i, and hence it will appear in $E_{i+1,c} \setminus E_{i,c}$. Thus there are at least $n_c - n_c^\star$ edges in $E_{i+1,c} \setminus E_{i,c}$. We can sum over colors c to count E_{i+1}:

$$|E_{i+1}| = \sum_{c \in C} \big(|E_{i,c}| + |E_{i+1,c} \setminus E_{i,c}|\big)$$
$$\ge \sum_{c \in C} \big((|V_i| - n_c) + (n_c - n_c^\star)\big) = \sum_{c \in C} (|V_i| - n_c^\star) .$$

To bound this sum, consider an arbitrary spanning tree T of G'. Since G' is connected we have $|T| = |V_i| - 1$, and also $|T \cap E_c^\star| \le$

$|V_i| - n_c^\star$ for each color c. We thus have:

$$|E_{i+1}| \geq \sum_{c \in C} |T \cap E_c^\star| = \sum_{e \in T} |Q(e)| \geq |T| \cdot (1+\varepsilon)a = (1+\varepsilon)a(|V_i|-1)\,.$$

Since $|V_1| = 2$, this implies that $|E_2| \geq (1+\varepsilon)a$. For iteration $i > 1$, note that by definition of arboricity, we have $|E_i|/(|V_i|-1) \leq a$, and so $|E_{i+1}| \geq (1+\varepsilon)a \cdot |E_i|/a = (1+\varepsilon)|E_i|$. Hence $|E_{\ell+1}| \geq (1+\varepsilon)^\ell a$ for each $\ell \geq 1$. The overall graph has $m \leq na$ edges, so the process must terminate by iteration $\ell = \lceil \log_{1+\varepsilon} n \rceil$. □

Having found the almost-augmenting sequence from a given starting edge e, we can short-circuit it into an augmenting sequence as shown in the following result:

Proposition 3.5. *If there exists an almost augmenting sequence P from e to e', then there exists an augmenting sequence from e to e' which is a subsequence of P.*

Proof. Let $P = (e_1, e_2, \ldots e_\ell, c)$ be an almost augmenting sequence with $e_1 = e$ and $e_\ell = e'$ of minimal length ℓ. If P satisfies (A3) we are done. If not, suppose that $e_i \in C(e_j, \psi(e_i))$ for $j < i-1$. Then, $P' = (e_1, \ldots, e_j, e_i, \ldots e_\ell, c)$ would also be an almost augmenting path of length $\ell' < \ell$ which is a subsequence of P. This contradicts minimality of ℓ. □

Theorem 3.2 now follows immediately from Lemma 3.4 and Proposition 3.5.

4 LOCAL FOREST DECOMPOSITIONS VIA AUGMENTATION

Algorithm 3 is a high-level description of our forest decomposition algorithm. It involves a parameter R and a subroutine $\texttt{CUT}(X, R)$; their precise specification will be given next, but, as a summary: given a partial forest decomposition, the procedure $\texttt{CUT}(X, R)$ should remove some edges in $E(N^R(X)) \setminus E(X)$ so there is no monochromatic path from X to $V \setminus N^R(X)$. If every execution of CUT disconnects X and $V \setminus N^R(X)$, we say that the execution of Algorithm 3 *is good.*

Algorithm 3 Forest_Decomposition(G)

1: Initialize $\psi \leftarrow \emptyset$.
2: Compute an $(O(\log n), O(\log n))$-network decomposition in $G^{2\cdot(R+R')}$ for $R' = \lceil K' \log n/\varepsilon \rceil$.
3: **for each** class z in the network decomposition **do**
4: **for** each component C in the class z **in parallel do**
5: Let $C' = N^{R'}(C)$ and $C'' = N^{R+R'}(C)$.
6: Execute $\texttt{CUT}(C', R)$.
7: Modify ψ so that edges inside C become colored and edges outside $N^{R'}(C)$ are unchanged.

Here K' is a universal constant to be specified later. We summarize the algorithm as follows:

Theorem 4.1. *Algorithm 3 can be implemented in $O(R\log^2 n + \log^3 n/\varepsilon)$ rounds. If the execution of the algorithm is good and every edge has a palette of size $\lceil (1+\varepsilon)a \rceil$, then at the termination, ψ is a list forest decomposition of all edges of G not removed by* CUT.

Proof. Let us define $D = R + R'$. At the first step, we use the algorithm of [19] to obtain an $(O(\log n), O(\log n))$-network decomposition in the power graph G^{2D} in $O(D\log^2 n)$ rounds.

Algorithm 3 processes each cluster C of a given class, and colors all edges that are adjacent to or inside C (Line 4 to Line 7). Thus, if an edge uv is not removed, it will become colored when we process the first class containing u or v. From Corollary 3.3, we know that it is possible to modify ψ within $N^{R'}(C)$ alone, such that all edges within C become colored. Furthermore, if the execution is good, then $\texttt{CUT}(C', R)$ disconnects C' from $V \setminus C''$ for every subgraph induced by each color class c. Thus, for any putative coloring ψ' which fully colors the edges inside C and does not modify edges outside C'', we can check if ψ' is indeed an LFD by looking locally only within C''. The reason for this is that there cannot be any new cycles in ψ' outside C'', since the edges are not modified there, and thus any cycles would have to be confined to C''.

The distance between clusters in the same class is at least $2D+1$. Moreover, if u, v are adjacent in G^{2D}, their distance is at most $2D$. So each cluster C has weak diameter at most $O(D\log n)$, and also the balls C_1'' and C_2'' must be disjoint for any two clusters C_1 and C_2 of the same class. So each cluster can be processed independently without interfering with others. Therefore, Line 4 to Line 7, including implementation of CUT, can be simulated locally in C'' in $O(D\log n)$ rounds. Since there are $O(\log n)$ classes, the total running time is $O(D\log^2 n)$. □

It now remains to specify $\texttt{CUT}(C', R)$. For each color c, define $H_c[C'']$ to be the collection of all c-colored edges in $C'' \setminus C'$. The goal of $\texttt{CUT}(C', R)$ is to break all paths in each $H_c[C'']$ from C' to vertices outside C''. We call the subgraph induced by the removed edges from all $\texttt{CUT}(C', R)$ instances the *leftover subgraph.* When we remove an edge, we can orient it toward either of its vertices. We want to bound the maximum number of removed out-neighbors of any vertex; this can be viewed as a load-balancing problem, where the load of a vertex is its outdegree in the leftover subgraph. We describe this in Section 4.1, with appropriate choices for parameter R. See Figure 3.

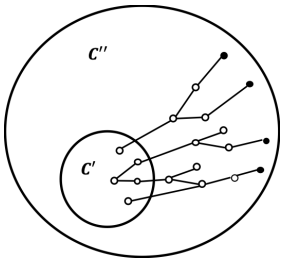


Figure 3: $H_c[C'']$ with $R = 3$. We want to disconnect C' from all nodes with distance R (black nodes) from C'.

At the end of this process, we combine the list-forest-decomposition of the main graph with a forest decomposition on the deleted edges. We summarize this in Section 4.2.

4.1 Implementing CUT

Let us define $T = O(\log n)$ to be the number of classes in the network decomposition. We now describe a few load balancing

strategies to implement CUT, with different parameter choices for the radius R. We summarize these rules as follows:

Theorem 4.2. *The procedure* CUT *can be implemented so that w.h.p. the leftover subgraph has pseudo-arboricity at most $\lceil \varepsilon a \rceil$ and the execution of Algorithm 3 is good, with the following values for parameter R:*

(1) $R = O(\frac{\log^2 n}{\varepsilon})$ *if* $\varepsilon a \geq \Omega(\log n)$.

(2) $R = O(\frac{\log n}{\varepsilon})$ *if* $\varepsilon a \geq \Omega(\log n)$ *for forest decomposition.*

(3) $R = O(\frac{\Delta^{2/\lceil a\varepsilon \rceil} \log \Delta \log^2 n}{a\varepsilon^2})$ *if* $\varepsilon a \leq \log \Delta$

(4) $R = O(\frac{\log^2 n}{\varepsilon})$ *if* $\varepsilon a \geq \log \Delta$

The first two results here follow from straightforward diameter-reduction algorithms.

Proof of Theorem 4.2(1). For the first algorithm for $\mathsf{CUT}(C', R)$, we apply Proposition 1.5 to the forests $H[C'']$, using parameter $\varepsilon' = \varepsilon/(2T)$ in place of ε. This reduces the diameter of each forest $H_c[C'']$ to $O(\log^2 n/\varepsilon)$, while deleting an edge-set of arboricity at most $\lceil \varepsilon a/(2T) \rceil$. In particular, when $R \geq \Omega(\log^2 n/\varepsilon)$, there cannot be any R-long path in $H_c[C'']$ for any color c. Over the T iterations of Algorithm 3, the leftover subgraph has arboricity at most $T \cdot \lceil \varepsilon a/(2T) \rceil$; since $T = O(\log n)$ and $\varepsilon a \geq \Omega(\log n)$, this is at most εa. □

Proof of Theorem 4.2(2). To implement $\mathsf{CUT}(C', R)$ for a color c, we choose an arbitrary root in C' for each tree of $H_c[C'']$. Next, we choose an integer J_c uniformly at random from $[N]$, where $N = \lfloor R/2 \rfloor$ and $R = \lceil 80T/\varepsilon \rceil = \Theta(\frac{\log n}{\varepsilon})$. We then delete all edges e in $H_c[C']$ whose depth d_e from the root satisfies $d_e \equiv J_c \mod N$. After this deletion step, each component of $H_c[C'']$ has depth at most N, and hence has maximum path length of R. So $V \setminus C''$ is disconnected from C' with probability one and Algorithm 3 is always good. By a union bound over vertices, w.h.p. every vertex has at most εa out-neighbors in the orientation. □

We now turn to the last two results of Theorem 4.2. We assume for this that $\varepsilon a = O(\log n)$, as otherwise we could apply Theorem 4.2(1). In particular, due to our assumption that $\varepsilon \geq 1/n$, we have $m \leq na \leq \text{poly}(n)$.

We use the following algorithm: first, before running Algorithm 3, use Theorem 1.3 to obtain a $3a$-orientation J of G. We maintain a counter $L(v)$ for each vertex v. To execute $\mathsf{CUT}(C', R)$, we go through each vertex v with $L(v) < \varepsilon a$; with probability p (to be specified), we delete one random out-neighbor of v with respect to J and increment $L(v)$ by one.

We say a vertex u is *overloaded* if $L(u) \geq \varepsilon a$, otherwise it is *underloaded*. For an edge e oriented from u to v in J, we say that e is *overloaded* or *underloaded* if u is. Given a path P, let $E_0(P)$ and $E_1(P)$ denote the set of underloaded and overloaded edges in P respectively. A length-R path in $H_c[C'']$ is called a *live branch*.

Proposition 4.3. *Suppose that $p \geq \frac{K'' a \log n}{\eta R}$, where $\eta \in [0, 1/2]$ and K'' is a sufficiently large constant. Then w.h.p., either the execution of Algorithm 3 is good, or some live branch P has $|E_0(P)| < \eta R$.*

Proof. Any path from C' to $V \setminus C''$ must have length at least R, hence will pass over some live branch. So it suffices to show that any live branch P in $H_c[C'']$ during an invocation of $\mathsf{CUT}(C', R)$ is cut. Each underloaded edge of P gets deleted with probability at least $p/(3a)$. Furthermore, such deletion events are negatively correlated, since at most outgoing edge per vertex can be deleted. Thus, assuming a live branch P has $|E_0(P)| \geq \eta R$, the probability that P remains is at most $(1 - p/(3a))^{\eta R} \leq e^{-pR\eta/(3a)}$. By our choice of p, this is at most $e^{-K'' \log n/3} \leq 1/\text{poly}(n)$.

Since each $H_c[C'']$ has at most n^2 live branches, by a union bound w.h.p. all live branches in $H_c[C'']$ are cut. Algorithm 3 invokes $\mathsf{CUT}(C', R)$ at most $O(n \log n)$ times, and the number of non-empty forests $H_c[C']$ is at most $m \leq \text{poly}(n)$. Hence, by a union bound over all the invocations, we conclude the algorithm is good or some live branch has $|E_0(P)| < \eta R$. □

Lemma 4.4. *Let $\eta \in [0, 1/2]$. If $R \geq \frac{K \Delta^{\frac{2+4\eta}{\lceil a\varepsilon \rceil}} \log^2 n}{\eta \varepsilon}$ for a sufficiently large constant K, then p can be chosen so that Algorithm 3 is good w.h.p.*

Proof. Let $t = \lceil \varepsilon a \rceil$, and let us set $p = \frac{K'' a \log n}{\eta R}$ according to Proposition 4.3. We first need to verify that $p \in [0, 1]$. For this, we can calculate:

$$p \leq \varepsilon a \cdot \frac{2K''}{K} \cdot \frac{\log n}{\log^2 n \Delta^{\frac{2+4\eta}{t}}} \leq \frac{\varepsilon a}{\Omega(K) \cdot \Delta^{\frac{2+4\eta}{t}} \log n}. \quad (1)$$

By our assumption that $\varepsilon a < \log n$, this is at most $\frac{1}{10e\Delta^{\frac{1+2\eta}{t}}} \leq 1$ for large enough K.

By Proposition 4.3, it suffices to show that $|E_1(P)| < (1-\eta)R$ for all live branches P during the algorithm execution. For this, it will suffice to bound the probability that all edges in S are overloaded where S is arbitrary subset of the edges in P. Since P is a path, edges at distance-2 are vertex-disjoint. Thus, S involves at least $|S|/2$ distinct vertices. For each such vertex u, the value $L(u)$ is a truncated Binomial random variable with mean at most Tp. Hence u is overloaded with probability at most $q = F_+(Tp, t)$, where $F_+(\mu, t)$ denotes the Chernoff upper-tail probability, i.e. the upper bound on the probability that a Binomial random variable with mean μ exceeds value t. Accordingly, the probability that all edges in S are overloaded is at most $q^{|S|/2}$.

Since $T \leq O(\log n)$, by Eq. (1) we have $pT \leq \frac{\varepsilon a}{10e\Delta^{\frac{2+4\eta}{t}}}$ for large enough K, and therefore

$$q = F_+(Tp, t) \leq \left(\frac{eTp}{t}\right)^t \leq \left(\frac{e \cdot \varepsilon a}{10e\Delta^{(2+4\eta)/t} \cdot \lceil \varepsilon a \rceil}\right)^t \leq \frac{1}{10\Delta^{2+4\eta}}.$$

Using this bound on q, and using the fact that Chernoff bounds apply to sums of variables which obey an upper negative-correlation property, we calculate:

$$\Pr(|E_1(P)| > (1-\eta)R) \leq F_+(R\sqrt{q}, (1-\eta)R) \leq \left(\frac{e\sqrt{q}}{1-\eta}\right)^{(1-\eta)R} \leq \left(\frac{e}{\sqrt{10}(1-\eta)\Delta^{1+2\eta}}\right)^{(1-\eta)R}.$$

Since $\eta \leq 1/2$ and $R \geq \omega(\log n)$, we get:

$$\Pr(|E_1(P)| > (1-\eta)R) \leq (e/\sqrt{10})^{R/2} \Delta^{-(1-\eta)(1+2\eta)R} \leq (0.93/\Delta)^R \leq 1/(\text{poly}(n)\Delta^R).$$

There are at most $n\Delta^{R-1}$ paths of length R. By a union bound, we conclude that $|E_0(P)| \geq \eta R$ holds w.h.p. for all such paths. □

We can now conclude our analysis by choosing appropriate values for parameters p, η, R:

Proof of Theorem 4.2(3),(4). In the algorithm for CUT, each vertex deletes at most εa of its outgoing neighbors under J. Hence, the leftover subgraph has pseudo-arboricity εa with probability one. Given η, we choose R, p according to Lemma 4.4 so that Algorithm 3 is good w.h.p. For the first result, we set $\eta = \frac{\lceil a\varepsilon \rceil}{2 \log \Delta}$, giving $R = (K\Delta^{\frac{2+4\eta}{\lceil a\varepsilon \rceil}} \log^2 n)/(\varepsilon\eta) \leq O(\frac{\Delta^{2/\lceil \varepsilon a \rceil} \log \Delta \log^2 n}{a\varepsilon^2})$. For the second result, we set $\eta = 1/2$; the bound on R is completely analogous. □

4.2 Putting Everything Together

The following result now summarizes the situation after applying Algorithm 3. The runtime bounds follow from Theorem 4.1 and the bounds on R given in Theorem 4.2.

Theorem 4.5. *If every edge has a palette of size $\lceil (1+\varepsilon)a \rceil$, then w.h.p. Algorithm 3 generates a partial list-forest decomposition, such that the uncolored edges have pseudo-arboricity at most $\lceil \varepsilon a \rceil$. It has the following complexity:*

- *With no restriction on εa, complexity is $O(\frac{\Delta^{2/\lceil a\varepsilon \rceil} \log \Delta \log^4 n}{a\varepsilon^2})$.*
- *If $\varepsilon a \geq \log \Delta$, complexity is $O(\frac{\log^4 n}{\varepsilon})$.*
- *If $\varepsilon a \geq \Omega(\log n)$, complexity is $O(\frac{\log^4 n}{\varepsilon})$.*
- *If $\varepsilon a \geq \Omega(\log n)$ for forest decomposition, complexity is $O(\frac{\log^3 n}{\varepsilon})$.*

We now need to combine the forest decomposition of the main graph with a forest decomposition on the leftover graph. For ordinary coloring, this is straightforward; we summarize it as follows:

Theorem 4.6. *We can obtain an $(1+\varepsilon)a$-FD of G of diameter D, under the following conditions:*

- *If $\varepsilon a \geq 3$, then $D \leq n$, and the complexity is $O(\frac{\Delta^2 \log \Delta \log^4 n}{\varepsilon})$.*
- *If $4 \leq \varepsilon a \leq \log \Delta$, then $D \leq O(\frac{\log n}{\varepsilon})$, and the complexity is $O(\frac{\Delta^{2/\lceil a\varepsilon \rceil} \log \Delta \log^4 n}{a\varepsilon^2})$.*
- *If $\varepsilon a \geq \log \Delta$, then $D \leq O(\frac{\log n}{\varepsilon})$, and the complexity is $O(\frac{\log^4 n}{\varepsilon})$.*
- *If $\varepsilon^2 a \geq \Omega(\log \Delta)$, then $D \leq O(\frac{1}{\varepsilon})$, and the complexity is $O(\frac{\log^4 n}{\varepsilon})$.*
- *If $\varepsilon a \geq \Omega(\log n)$, then $D \leq O(\frac{1}{\varepsilon})$, and the complexity is $O(\frac{\log^3 n}{\varepsilon})$.*

Proof. Applying Theorem 4.5 with $\varepsilon' = \varepsilon/10$ in place of ε gives a partial $\lceil a(1+\varepsilon/10) \rceil$-FD of G, where the uncolored edges E' have $a^*(E') \leq \lceil \varepsilon a/10 \rceil$. Theorem 1.3(3) then yields a $\lfloor 2.01 a^*(E') \rfloor$-FD of E'. Taken together, these give a k-FD of G for $k = a + \lfloor 2.01 \lceil \varepsilon a/10 \rceil \rfloor + \lceil \varepsilon a/10 \rceil$; since $\varepsilon a \geq 3$, this is at most $a(1+\varepsilon)$. For the next four results, we apply Proposition 1.5 to convert this into a $k + \lceil \varepsilon' a/10 \rceil$-FD of G, with the given bounds on the diameter. □

For list-coloring, we piece together the main graph and leftover graph by partitioning the color-space C for each vertex. See the full paper for details. We summarize the results here:

Theorem 4.7. *Suppose that G is a multigraph where each edge has a palette of size $(1+\varepsilon)a$. We can obtain a list-forest-decomposition of G of diameter D w.h.p., under the following conditions:*

- *If $\varepsilon a \geq \Omega(\log n)$, the complexity is $O(\frac{\log^4 n}{\varepsilon})$ and $D = O(\frac{\log n}{\varepsilon})$.*
- *If $\varepsilon^2 a \geq \Omega(\log \Delta)$, the complexity is $O(\frac{\log^4 n}{\varepsilon^2})$ and $D = O(\frac{\log n}{\varepsilon^2})$.*

5 STAR-FOREST DECOMPOSITION FOR SIMPLE GRAPHS

Consider a simple graph G of arboricity a which is equipped with a t-orientation for $t = \lceil (1+\varepsilon)a \rceil$. Let us consider the following process: each vertex v in the graph selects a color set $C_v \subseteq C$. For $c \in C_v$, we say that v is a c-leaf and for $c \notin C_v$ we say that v is a c-center. For each vertex v, we construct an associated bipartite graph $H_v(C)$, whose left-nodes correspond to C and whose right nodes correspond to the out-neighbors $A(v)$, and there is an edge from left-node i to right-node u iff $i \in C_v \setminus C_u$ and $i \in Q(uv)$.

Proposition 5.1. *If each graph $H_v(C)$ has a matching of size at least $t - \delta$, then in $O(1)$ rounds we can partition the edges as $E = E_0 \sqcup E_1$ and obtain a LSFD ϕ_0 of E_0, such that E_1 has pseudo-arboricity at most δ.*

Proof. For each edge (i, u) in the matching M_v of $H_v(C)$, we set $\phi(vu) = i$. Thus, all color-i edges have the form vu for $i \in C_v \setminus C_u$ and $(i, u) \in M_v$. Since M_v is a matching, the edges of each color i are a collection of stars on the i-centers (nodes u with $i \notin C_u$) and i-leaves (nodes u with $i \in C_u$). The residual uncolored graph has a δ-orientation, by orienting each vertex v to each unmatched vertex $u \in A(v)$, hence its pseudo-arboricity is at most δ. □

So we need to choose C so that every graph $H_v(C)$ has a large matching. The following two results show that an appropriately chosen random coloring has this property with good probability.

Lemma 5.2. *Suppose that $a\varepsilon \geq 100(\sqrt{\log \Delta} + \log a)$. If $C = [t]$ and each set C_u is chosen uniformly at random among a-element subsets of C, then for any vertex v there is a probability of at least $1 - 1/\Delta^{10}$ that $H_v(C)$ has a matching of size at least $a(1-\varepsilon)$.*

Lemma 5.3. *Suppose that $\varepsilon \leq 10^{-6}$ and $a\varepsilon \geq 10^6 \log \Delta$ and each edge has a palette of size $a(1+200\varepsilon)$. If we form each set C_u by selecting each color independently with probability $1-\varepsilon$, then for any vertex v there is a probability of at least $1 - 1/\Delta^{10}$ that $H_v(C)$ has a matching of size t.*

This leads to our main results for star-forest decomposition:

Theorem 5.4. *Let G be a simple graph with arboricity a. If $a\varepsilon \geq \Omega(\sqrt{\log \Delta} + \log a)$, we get an $a(1+\varepsilon)$-SFD in $O(\frac{\log^3 n}{\varepsilon})$ rounds w.h.p. If $a\varepsilon \geq \Omega(\log \Delta)$, we get an $a(1+\varepsilon)$-LSFD in $O(\frac{\log^3 n}{\varepsilon})$ rounds w.h.p.*

Proof. For the first result, we apply Theorem 1.1 to obtain the required t-orientation in $O(\log^3 n/\varepsilon)$ rounds. Next, use the Lovász Local Lemma algorithm of [17] to choose C such that every graph $H_v(C)$ has a matching of size at least $t - 2a\varepsilon$. Each vertex v corresponds to a bad-event that the matching size is too small. By Lemma 5.2, this event has probability at most $p = \Delta^{-10}$ and depends on $d = \Delta^2$ other such events (u and v can only affect each

other if they have distance at most 2). Thus, the criterion $pd^2 \ll 1$ is satisfied and the LLL algorithm runs in $O(\log n)$ rounds.

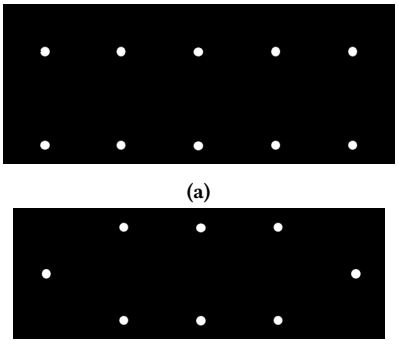
Given the choice of sets C_v for all v, we then apply Proposition 5.1 to get a $(1+\varepsilon)a$-SFD of G, plus a left-over graph of pseudo-arboricity at most $2a\varepsilon$. We finish by applying Theorem 1.3 to get a $6.01a\varepsilon$-SFD of the left-over graph. Overall, we get a $(1+7.01\varepsilon)a$-SFD; the result then holds by reparametrizing with $\varepsilon' = \varepsilon/100$ in place of ε.

The second result is completely analogous, except we use Lemma 5.3 to obtain the matchings of H_v. In this cases, there is no left-over graph to recolor. □

We remark that the main algorithmic bottleneck for Theorem 5.4 is obtaining the t-orientation. For example, we could alternatively use the algorithm of [55] to obtain the t-orientation, and hence obtain the $a(1+\varepsilon)$-LSFD, in $\tilde{O}(\log^2 n/\varepsilon^2)$ rounds.

6 LOWER BOUNDS ON ROUND COMPLEXITY

For parameters $a \geq 2$ and $t \geq 1$, we construct multigraph G as follows. We begin with four named vertices x_1, x_2, y_1, x_2. There are $\lfloor a/2 \rfloor$ parallel edges from x_1 to x_2 and $\lfloor a/2 \rfloor$ parallel edges from y_1 to y_2. We construct a path P_1 of t vertices arranged from x_1 to y_1, with k parallel edges between successive vertices on the path (including x_1 and y_1 themselves). We construct another path P_2 of t vertices arranged in a line from x_2 to y_2 with k parallel edges. From G, we can form a related graph G' by contracting x_1 to x_2 to a single vertex x, as well as y_1 to y_2 to a single vertex y. See Figure 4.



(a)

(b)

Figure 4: An illustration of G and G' when $k = 4$ and $t = 4$.

It can be seen that the graph G has arboricity a, has $n = 2t + 4$ vertices and has maximum degree $\Delta = O(a)$. We have the following crucial observation for these graphs:

Lemma 6.1. *Suppose that $r \leq t/2$ and $\ell \leq 2a$. Then for any r-round algorithm A for ℓ-forest-decomposition on G with success probability p, which does not use information about vertex ID's, there is a probability of $\frac{p^2}{128-p^2}$ that there are at least $p^2a/64$ colors c such that (x_1, x_2) and (y_1, y_2) both have a c-colored edge.*

Proof. For any color i, let X_i denote the event that (x_1, x_2) has an i-colored edge and Y_i denote the event that (y_1, y_2) has an i-colored edge, after we run algorithm A on the graph.

Since the edges (x_1, x_2) and (y_1, y_2) have distance t in either graph, the random variables X_i, Y_i are independent for each i. Furthermore, since the view from (x_1, x_2) is isomorphic to the view from (y_1, y_2), they follow the same distribution. Thus, if we let p_i denote the probability of X_i, then we have $\mathrm{E}[X_i Y_i] = \mathrm{E}[X_i]\,\mathrm{E}[Y_i] = p_i^2$. Hence, letting Z denote the number of colors appearing on both (x_1, y_1) and (x_2, y_2), we have $\mathrm{E}[Z] = \sum_{i=1}^{\ell} p_i^2$. On the other hand, whenever A returns a forest-decomposition, we have $\sum_{i=1}^{\ell} X_i = \sum_{i=1}^{\ell} Y_i = \lfloor a/2 \rfloor$. Taking expectations, it can then be shown that $\mathrm{E}[Z] = \sum_{i=1}^{\ell} p_i^2 \geq p^2 a/32$. Markov's inequality applied to $\ell - Z$ gives $\Pr(Z < p^2a/64)2 - \frac{1}{1-p^2/128}$. □

Putting these results together, we obtain the following:

Theorem 6.2. *Let $a, n \geq 2$ be an arbitrary integers and $\varepsilon \in (0, 1)$. Any randomized algorithm for $(1+\varepsilon)a$-forest-decomposition on n-node graphs of arboricity a with success probability at least 0.993 requires $\Omega(\min\{n, 1/\varepsilon\})$ rounds. This bound holds even on graphs of maximum degree $\Delta = O(a)$.*

Proof. It suffices to show this under the assumption that $1/\varepsilon \leq 0.0001n$. Also, we can assume without loss of generality that A does not depend upon any vertex ID's; if A does so, we simply choose new independent random vertex ID's at the beginning of the process. Now suppose for contradiction that A runs in fewer than $0.001/\varepsilon$ rounds and, with probability $p = 0.993$, produces a forest-decomposition ψ on G.

By Lemma 6.1, there is a probability of at least $\frac{p^2}{128-p^2} \geq 0.00776$ that there at least $p^2a/32 \geq 0.03a$ colors c such that (x_1, x_2) and (y_1, y_2) both have c-colored edges. By a counting argument, it can be shown that whenever ψ is a forest-decomposition, then there are at most $4t\varepsilon a \leq 0.004a$ colors such that that induced coloring on G' is acyclic. But note that if color c has edges on (x_1, x_2) and (y_1, y_2) and the induced coloring on G' is cyclic, then also color c on G is cyclic. Hence, with probability at least 0.00776, at least $0.03a - 0.004a > 0$ colors induce a cycle. In particular, ψ is a forest decomposition with probability at most $1 - 0.00776 < 0.993$, contradiction. □

Proposition 6.3. *In simple graphs with arboricity 2, computing a 2-forest-decomposition with success probability at least 0.5 requires $\Omega(n)$ rounds.*

Proof. Construct G with parameters $a = 2$ and $t = \lfloor (n-2)/8 \rfloor$, and replace every set of parallel edges by a copy of the complete graph K_4. The resulting simple graph H has $2(4t+1) \leq n$ nodes and it has arboricity 2. Along similar lines to Proposition 6.2, it can be shown that any algorithm for 2-forest-decomposition with more than 0.5 probability requires at least $n/100$ rounds. □

7 ACKNOWLEDGMENTS

Hsin-Hao Su is supported by NSF Grant No. CCF-2008422.

Thanks to Vladimir Kolmogorov, for suggesting how to set the parameters for Lemma 5.2. Thanks to Noga Alon, for explaining some lower bounds for star arboricity. Thanks to Louis Esperet for some suggestions on notations and terminology. Thanks to anonymous conference reviewers for helpful comments and suggestions.

REFERENCES

[1] Ilan Algor and Noga Alon. 1989. The star arboricity of graphs. *Discrete Mathematics* 43, 1–3 (1989), 11–22.

[2] Noga Alon, Colin McDiarmid, and Bruce Reed. 1992. Star arboricity. *Combinatorica* 12, 4 (1992), 375–380.

[3] Yasukazu Aoki. 1990. The star-arboricity of the complete regular multipartite graphs. *Discrete Mathematics* 81, 2 (1990), 115–122.

[4] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. 1996. Fast Distributed Network Decompositions and Covers. *J. Parallel and Distrib. Comput.* 39, 2 (1996), 105–114.

[5] Bahman Bahmani, Ashish Goel, and Kamesh Munagala. 2014. Efficient Primal-Dual Graph Algorithms for MapReduce. In *WAW (Lecture Notes in Computer Science 8882)*. 59–78.

[6] Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. 2012. Densest Subgraph in Streaming and MapReduce. *Proc. VLDB Endowment* 5, 5 (2012), 454–465.

[7] Leonid Barenboim and Michael Elkin. 2010. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. *Distributed Computing* 22, 5-6 (2010), 363–379.

[8] Leonid Barenboim and Michael Elkin. 2011. Deterministic Distributed Vertex Coloring in Polylogarithmic Time. *J. ACM* 58, 5 (2011), Article #23.

[9] Leonid Barenboim and Michael Elkin. 2013. Distributed Graph Coloring: fundamentals and Recent Developments. *Synthesis Lectures on Distributed Computing Theory* 4, 1 (2013), 1–171.

[10] Soheil Behnezhad, Sebastian Brandt, Mahsa Derakhshan, Manuela Fischer, MohammadTaghi Hajiaghayi, Richard M. Karp, and Jara Uitto. 2019. Massively Parallel Computation of Matching and MIS in Sparse Graphs. In *PODC*. 481–490.

[11] Edvin Berglin and Gerth Stølting Brodal. 2020. A Simple Greedy Algorithm for Dynamic Graph Orientation. *Algorithmica* 82, 2 (2020), 245–259.

[12] Anton Bernshteyn. 2020. A Fast Distributed Algorithm for $(\Delta+1)$-Edge-Coloring. *arXiv preprint arXiv:2006.15703* (2020).

[13] Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos E. Tsourakakis. 2015. Space- and Time-Efficient Algorithm for Maintaining Dense Subgraphs on One-Pass Dynamic Streams. In *STOC*. 173–182.

[14] Gerth Stølting Brodal and Rolf Fagerberg. 1999. Dynamic Representations of Sparse Graphs. In *WADS*. 342–351.

[15] Y.-J. Chang, Q. He, W. Li, S. Pettie, and J. Uitto. 2018. The complexity of distributed edge coloring with small palettes. In *SODA*. 2633–2652.

[16] Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *APPROX (Lecture Notes in Computer Science 1913)*. 84–95.

[17] Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. 2017. Distributed algorithms for the Lovász local lemma and graph coloring. *Dist. Comp.* 30, 4 (2017), 261–280.

[18] Devdatt Dubhashi, David A Grable, and Alessandro Panconesi. 1998. Near-optimal, distributed edge colouring via the nibble method. *Theoretical Computer Science* 203, 2 (1998), 225–251.

[19] Michael Elkin and Ofer Neiman. 2016. Distributed Strong Diameter Network Decomposition: Extended Abstract. In *PODC*. 211–216.

[20] Michael Elkin, Seth Pettie, and Hsin-Hao Su. 2015. $(2\Delta-1)$-edge-coloring is much easier than maximal matching in the distributed setting. In *SODA*. 355–370.

[21] Hossein Esfandiari, MohammadTaghi Hajiaghayi, and David P. Woodruff. 2016. Applications of Uniform Sampling: densest Subgraph and Beyond. In *SPAA*. 397–399.

[22] Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. 2017. Deterministic Distributed Edge-Coloring via Hypergraph Maximal Matching. In *FOCS*. 180–191.

[23] Harold N. Gabow and Matthias Stallmann. 1985. Efficient algorithms for graphic matroid intersection and parity. In *ICALP*. 210–220.

[24] Harold N. Gabow and Herbert H. Westermann. 1992. Forests, frames, and games: Algorithms for matroid sums and applications. *Algorithmica* 7, 1 (1992), Article #465.

[25] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. 1989. A Fast Parametric Maximum Flow Algorithm and Applications. *SIAM J. Comput.* 18, 1 (1989), 30–55.

[26] Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. 2018. On Derandomizing Local Distributed Algorithms. In *FOCS*. 662–673.

[27] Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, and Yannic Maus. 2018. Improved distributed Δ-coloring. In *PODC*. 427–436.

[28] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. 2017. On the complexity of local distributed graph problems. In *STOC*. 784–797.

[29] Mohsen Ghaffari, Fabian Kuhn, Yannic Maus, and Jara Uitto. 2018. Deterministic Distributed Edge-coloring with Fewer Colors. In *STOC*. 418–430.

[30] Mohsen Ghaffari, Silvio Lattanzi, and Slobodan Mitrović. 2019. Improved Parallel Algorithms for Density-Based Network Clustering. In *ICML*, Vol. 97. 2201–2210.

[31] Mohsen Ghaffari and Hsin-Hao Su. 2017. Distributed Degree Splitting, Edge Coloring, and Orientations. In *SODA*. 2505–2523.

[32] A. V. Goldberg. 1984. *Finding a Maximum Density Subgraph.* Technical Report UCB/CSD-84-171. EECS Department, University of California, Berkeley.

[33] A. D. Gore, A. Karandikar, and S. Jagabathula. 2007. On High Spatial Reuse Link Scheduling in STDMA Wireless Ad Hoc Networks. In *IEEE Global Telecommunications Conference (GLOBALCOM)*. 736–741.

[34] Anupam Gupta, Amit Kumar, and Cliff Stein. 2014. Maintaining Assignments Online: Matching, Scheduling, and Flows. In *SODA*. 468–479.

[35] S Louis Hakimi. 1965. On the degrees of the vertices of a directed graph. *Journal of the Franklin Institute* 279, 4 (1965), 290–308.

[36] David G. Harris. 2020. Distributed local approximation algorithms for maximum matching in graphs and hypergraphs. *SIAM J. Comput.* 49, 4 (2020), 711–746.

[37] Hiroshi Imai. 1983. Network-Flow Algorithms For Lower-Truncated Transversal Polymatroids. *Journal of the Operations Research Society of Japan* 26, 3 (1983), 186–211.

[38] Samir Khuller and Barna Saha. 2009. On Finding Dense Subgraphs. In *ICALP*. 597–608.

[39] Tsvi Kopelowitz, Robert Krauthgamer, Ely Porat, and Shay Solomon. 2014. Orienting Fully Dynamic Graphs with Worst-Case Time Bounds. In *ICALP*. 532–543.

[40] Łukasz Kowalik. 2006. Approximation Scheme for Lowest Outdegree Orientation and Graph Density Measures. In *ISAAC*. 557–566.

[41] Fabian Kuhn. 2020. Faster Deterministic Distributed Coloring Through Recursive List Coloring. In *SODA*. 1244–1259.

[42] Nathan Linial. 1992. Locality in Distributed Graph Algorithms. *SIAM J. Comput.* 21, 1 (1992), 193–201.

[43] Nathan Linial and Michael E. Saks. 1993. Low diameter graph decompositions. *Combinatorica* 13, 4 (1993), 441–454.

[44] Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T. Vu. 2015. Densest Subgraph in Dynamic Graph Streams. In *MFCS (2) (Lecture Notes in Computer Science, Vol. 9235)*. Springer, 472–482.

[45] C. St.J. A. Nash-Williams. 1964. Decomposition of Finite Graphs Into Forests. *Journal of the London Mathematical Society* s1-39, 1 (1964), 12–12.

[46] Alessandro Panconesi and Aravind Srinivasan. 1997. Randomized Distributed Edge Coloring via an Extension of the Chernoff–Hoeffding Bounds. *SIAM J. Comput.* 26, 2 (1997), 350–368. https://doi.org/10.1137/S0097539793250767 arXiv:https://doi.org/10.1137/S0097539793250767

[47] Jean-Claude Picard and Maurice Queyranne. 1982. A network flow solution to some nonlinear 0-1 programming problems, with applications to graph theory. *Networks* 12, 2 (1982), 141–159.

[48] S. Ramanathan and E. L. Lloyd. 1993. Scheduling algorithms for multihop radio networks. *IEEE/ACM Transactions on Networking* 1, 2 (1993), 166–177.

[49] James Roskind and Robert E. Tarjan. 1985. A Note on Finding Minimum-Cost Edge-Disjoint Spanning Trees. *Mathematics of Operations Research* 10, 4 (1985), 701–708.

[50] Václav Rozhoň and Mohsen Ghaffari. 2020. Polylogarithmic-Time Deterministic Network Decomposition and Distributed Derandomization. In *STOC*. 350–363.

[51] Atish Das Sarma, Ashwin Lall, Danupon Nanongkai, and Amitabh Trehan. 2012. Dense Subgraphs on Dynamic Networks. In *DISC (Lecture Notes in Computer Science, Vol. 7611)*. Springer, 151–165.

[52] Saurabh Sawlani and Junxing Wang. 2020. Near-Optimal Fully Dynamic Densest Subgraph. In *STOC*. 181–193.

[53] Paul D Seymour. 1998. A note on list arboricity. *Journal of Combinatorial Theory Series B* 72 (1998), 150–151.

[54] Jessica Shi, Laxman Dhulipala, and Julian Shun. 2020. Parallel Clique Counting and Peeling Algorithms. arXiv:2002.10047 [cs.DS] arXiv:2002.10047.

[55] Hsin-Hao Su and Hoa T. Vu. 2020. Distributed Dense Subgraph Detection and Low Outdegree Orientation. In *DISC*. 15:1–15:18.

[56] Hsin-Hao Su and Hoa T. Vu. 2019. Towards the Locality of Vizing's Theorem. In *STOC*. 355–364.

[57] Vadim G. Vizing. 1964. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz* 3, 7 (1964), 25–30.