S}I}A}M} J}. D}I}S}C}R}E}T}E} M}A}T}H}.
V}o}l}. 37, N}o}. 2, p}p}. 800--830

ON THE LOCALITY OF NASH-WILLIAMS FOREST DECOMPOSITION AND STAR-FOREST DECOMPOSITION*

DAVID G. HARRIS[r], HSIN-HAO SU[r], AND HOA T. VU[S]

Abstract. Given a graph $G = (V, E)$ with arboricity a, we study the problem of decomposing the edges of G into $(1+\eta)a$ disjoint forests in the distributed LOCAL model. Here G may be a simple graph or multigraph. While there is a polynomial time centralized algorithm for a-forest decomposition (e.g., [H. Imai, J. Oper. Res. Soc. Japan, 26 (1983), pp. 186--211]), it remains an open question how close we can get to this exact decomposition in the LOCAL model. Barenboim and Elkin [L. Barenboim and M. Elkin, Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition, Distrib. Comput., 22 (2010), pp. 363--379] developed a LOCAL algorithm to compute a $(2+\eta)a$-forest decomposition in $O(\frac{\log n}{\eta})$ rounds. Ghaffari and Su [Proc. 28th ACM-SIAM Symposium on Discrete Algorithms, 2017, pp. 2505--2523] made further progress by computing a $(1+\eta)a$-forest decomposition in $O(\frac{\log^3 n}{\eta^4})$ rounds when $\eta a = \Omega(\sqrt{a \log n})$; i.e., the limit of their algorithm is an $(a + \Theta(\sqrt{a \log n}))$-forest decomposition. This algorithm, based on a combinatorial construction of Alon, McDiarmid, and Reed [Combinatorica, 12 (1992), pp. 375--380], 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 ηa in $(1+\eta)a$-forest decomposition. We obtain a number of results with different parameters; some notable examples are the following: (1) An $O(\frac{a^{\circ} \log^4 n}{\eta})$-round algorithm when $\eta a = a^{\circ}(1)$ in multigraphs, where $\circ > 0$ is any arbitrary constant; (2) an $O(\frac{\log^4 n \log a}{\eta^2})$-round algorithm when $\eta a = a(\frac{\log a}{\log\log a})$ in multigraphs; (3) an $O(\frac{\log^4 n}{\eta})$-round algorithm when $\eta a = \Omega(\log n)$ in multigraphs (this also covers an extension of the forest-decomposition problem to list-edge-coloring); (4) an $O(\frac{\log^3 n}{\eta})$-round algorithm for star-forest decomposition for $\eta a = \Omega(\sqrt{\log a} + \log a)$ in simple graphs (when $\eta a \geq a(\log a)$, this also covers a list-coloring variant). Our techniques also give an algorithm for $(1+\eta)a$-outdegree-orientation in $O(\frac{\log^3 n}{\eta})$ rounds, which is the first algorithm with linear dependency on η^{-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, and Reed; the bounds on star-forest decomposition were not previously known even non constructively.

Key words. arboricity, LOCAL algorithm, star-arboricity, forest decomposition

MSC codes. 05C05, 68R10, 68W15, 05385

DOI. 10.1137/21M1434441

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 [47] shows that $a(G)$ is given by the formula

* Received by the editors July 18, 2021; accepted for publication (in revised form) October 23, 2022; published electronically June 7, 2023. This is an extended version of a paper appearing in the proceedings of the ACM Symposium on Principles of Distributed Computing (PODC) 2021.
https://doi.org/10.1137/21M1434441
Funding: The second author was supported by NSF grant CCF-2008422.

[r] Department of Computer Science, University of Maryland, Silver Spring, MD 20901, USA (davidgharris29@gmail.com).

[r] Department of Computer Science, Boston College, Chestnut Hill, MA 02467, USA (suhx@bc.edu).

[S] Department of Computer Science, San Diego State University, San Diego, CA 92182, USA (hvu2@sdsu.edu).

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

Note that the right-hand side is clearly a lower bound on a since each forest can consume at most $|V(H)| - 1$ edges in a subgraph H.

Forest decomposition can be viewed as a variant of proper edge coloring: in the latter problem, the edges should be partitioned into matchings, while in the former, they should be partitioned into forests. Like edge coloring, forest decomposition has applications to the scheduling of radio or wireless networks [34, 50]. In the centralized setting, a series of polynomial-time algorithms have been developed to compute a-forest decompositions [24, 25, 38, 51].

In this work, we study the problem of computing forest decompositions in the LOCAL model of distributed computing [43]. 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. Each vertex also has a unique ID which is a binary string of length $O(\log n)$. 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 eficient computation in the LOCAL model and the existential bounds of various combinatorial structures. For example, consider proper edge coloring. Vizing's classical result [59] shows that there exists a $(a + 1)$-edge-coloring in simple graphs. A long series of works have developed LOCAL algorithms using smaller number of colors [15, 19, 21, 30, 48, 58]. This culminated with a poly$(a, \log n)$-round algorithm in [12] for $(a + 1)$-edge-coloring, matching the existential bound.

Computing an a-forest decomposition in the LOCAL model requires $\Omega(n)$ rounds even in simple graphs with constant a (see Proposition 6.5). Accordingly, we aim for $(1 + \eta)a$-forests, i.e., ηa excess forests beyond the a-forests required existentially. Besides round complexity, a key objective is to minimize the value ηa.

The first results in the LOCAL model were due to Barenboim and Elkin [7], who developed an $O(\frac{\log n}{\eta})$-round algorithm for $(2 + \eta)a$-FD along with a lower bound of $\Omega(\frac{\log n}{\log a} - \log^* n)$ rounds for $O(a)$-FD. These have been building blocks in many distributed and parallel algorithms [7, 8, 10, 42, 56]. Open Problem 11.10 of [9] raised the question of whether it is possible to use fewer than $2a$ forests. Ghaffari and Su [32] made some progress with a randomized algorithm for $(1+\eta)a$-FD in $O(\log^3 n/\eta^4)$ rounds in simple graphs when $\eta = \Omega(\sqrt{\log n/a})$; i.e., the minimum number of obtainable forests is $a + \Omega(\sqrt{a \log n})$.

We make further progress with a randomized algorithm for $(a + 3)$-FD in poly$(a, \log n)$ rounds in multigraphs. The polynomial dependence on a can be removed when ηa is larger; for example, we obtain a $(1+\eta)a$-FD in $O(1/\eta)^t$ polylog(n) rounds for $\eta a = \Omega(\log a / \log\log a)$.

List forest decomposition. Similar to edge coloring, there is a list version of the forest decomposition problem: each edge e has a color palette $Q(e)$ and should choose a color $i(e) \in Q(e)$ so that, for any color c, the subgraph induced by the c-colored edges forms a forest. We refer to this as list forest decomposition (abbreviated LFD). We denote by $C = \bigcup_e Q(e)$ the set of all possible colors; this generalizes k-forest decomposition, which can be viewed as the case where $C = \{1, \dots, k\}$.

Based on general matroid arguments, Seymour [55] showed that an LFD exists whenever all of the palettes have size at least a. The total number of forests (one per color) may then be much larger than a; in this case, the excess is measured in

terms of the number of extra colors in the edges' palettes (in addition to the a colors required by the lower bound).

Seymour's construction can be turned into a polynomial-time centralized algorithm with standard matroid techniques. However, these do not extend to the LOCAL model. As a proof of concept, we give poly$(\log n, 1/\varepsilon)$-round algorithms when palettes have size $(1+\varepsilon)\alpha$ for $\alpha = \alpha(\min\{\log n, \alpha\sqrt{\log \alpha}\})$. A key open problem is to find an efficient algorithm for $\alpha \le \alpha(1)$.

Low-diameter and star-forest decompositions. We say the decomposition has diameter D if every tree in every c-colored forest has strong diameter at most D. Minimizing D is interesting from both practical and theoretical aspects. For example, given a k-FD of diameter D, we can find an orientation of the edges to make it into k rooted forests in $O(D)$ rounds of the LOCAL model.

In the extreme case $D = 2$, each forest is a collection of stars, i.e., a star-forest. This has received some attention in combinatorics. We refer to this as k-star-forest decomposition (abbreviated k-SFD); we give an $O(\frac{\log^3 n}{\varepsilon})$-round algorithm for $(1+\varepsilon)\alpha$-SFD when $\alpha = \alpha(\log \alpha + \alpha\sqrt{\log \alpha})$ in simple graphs. The algorithm also solves the list-coloring variant, which we call list-star-forest decomposition (abbreviated LSFD), when $\alpha = \alpha(\log \alpha)$.

For larger diameters, we show how to convert an arbitrary k-FD into a $(1+\varepsilon)k$-FD with diameter $D = O(\log n/\varepsilon)$; when εk is large enough, the diameter can be reduced further to $D = O(1/\varepsilon)$, which is optimal (see Proposition E.1).

1.1. Summary of results. Our results for forest decomposition balance a number of measures: the number of excess colors required, the running time, the tree diameters, LFD versus FD, and multigraphs versus simple graphs. Table 1 below summarizes a number of parameter combinations.

Here, $\rho > 0$ represents any desired constant and we use a_ρ to represent a constant term which may depend on ρ. Thus, for instance, the final listed algorithm requires excess $\overline{K}\log \alpha$ and the third listed algorithm requires excess $\overline{K_\rho}$, where $\overline{K}$ and $\overline{K_\rho}$ are universal constants.

We also show that $\alpha(1/\varepsilon)$ rounds are needed for $(1+\varepsilon)\alpha$-FD in multigraphs (see Theorem 6.4).

Note on deterministic algorithms. All of the algorithms we consider (unless specifically stated otherwise) are randomized algorithms which succeed with high

Table 1
Possible algorithms for forest decompositions of G.

Excess colors	Lists?	Multigraph?	Runtime	Forest Diameter
3	No	Yes	$O(\Delta^2 \alpha \log^4 n \log \Delta)$	$\le n$
≥ 4	No	Yes	$O(\Delta^2 \log^4 n \log \Delta/\varepsilon)$	$O(\log n/\varepsilon)$
$\Omega_\rho(1)$	No	Yes	$O(\Delta^\rho \log^4 n/\varepsilon)$	$O(\log n/\varepsilon)$
$\Omega_\rho(\frac{\log \Delta}{\log\log \Delta})$	No	Yes	$O_\rho(\log^4 n \log^\rho \Delta/\varepsilon)$	$O(\log n/\varepsilon)$
$\ge 4+\rho\log \Delta$	No	Yes	$O_\rho(\log^4 n/\varepsilon)$	$O(\log n/\varepsilon)$
$\Omega(\sqrt{\alpha \log \Delta})$	No	Yes	$O(\log^4 n/\varepsilon)$	$O(1/\varepsilon)$
$\Omega(\log n)$	No	Yes	$O(\log^3 n/\varepsilon)$	$O(1/\varepsilon)$
$\Omega(\sqrt{\alpha \log \Delta})$	Yes	Yes	$O(\log^4 n/\varepsilon^2)$	$O(\log n/\varepsilon^2)$
$\Omega(\log n)$	Yes	Yes	$O(\log^4 n/\varepsilon)$	$O(\log n/\varepsilon)$
$\Omega(\sqrt{\log \Delta} + \log \alpha)$	No	No	$O(\log^3 n/\varepsilon)$	2 (star)
$\Omega(\log \Delta)$	Yes	No	$O(\log^3 n/\varepsilon)$	2 (star)

probability (abbreviated w.h.p.), i.e., with probability at least $1 - 1/\text{poly}(n)$. It will turn out that the algorithms we develop have the property that if the algorithm fails (i.e., the output does not satisfy desired properties), then this can be detected by a node checking its local neighborhood during the algorithm run. Such randomized algorithms are referred to as Las Vegas algorithms in [27]. Using a recent breakthrough of [27, 52], such Las Vegas algorithms can be automatically derandomized with an additional polylog(n) factor in the runtime. For brevity, we will not explicitly show that the algorithms are Las Vegas and do not discuss any further issues of determinization henceforth.

1.2. Technical summary: Distributed augmentation. The results for forest decomposition in multigraphs are based on augmenting paths, where we color one uncolored edge and possibly change some of the colored edges while maintaining solution feasibility. Augmentation approaches have been used for many combinatorial constructions, such as coloring and matching. The forest-decomposition algorithm of Gabow and Westermann [25] also follows this approach. Roughly speaking, it works as follows: given an uncolored edge e_1, we try to assign it color c_1. If no cycle is created, we are done. Otherwise, if it creates a cycle C_1, we recolor some edge e_2 on C_1 with a different color $c_2 = c_1$. Continuing in this way gives an augmenting sequence $e_1, c_1, e_2, c_2, \ldots, e_l, c_l$, such that recoloring e_l in c_l does not create a cycle. This can be found using a Breadth-first Search algorithm in the centralized setting.

There are two main challenges for the LOCAL model. First, to get a distributed algorithm, we must color edges in parallel. Second, to get a local algorithm, we must restrict the recoloring to edges which are near the initial uncolored edge. Note that the augmenting sequences produced by the Gabow-Westermann algorithm can be long, and 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 $(1+n)a$-FD (or, more generally, an LFD) in a multigraph, there is an augmenting sequence of length $O(\log n/n)$ where, moreover, every edge in the sequence lies in the $O(\log n/n)$-neighborhood of the starting uncolored edge e_{init}. This characterization may potentially lead to other algorithms for forest decompositions. We show this through a key modification to the BFS algorithm for finding an augmenting sequence. In [25], if 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, only the edges within distance i of e_{init} get enqueued.

Network decomposition and removing edges. We will parallelize the algorithm by breaking the graph into low-diameter subgraphs similarly to [29]. However, there is a major roadblock we need to address: identifying an augmenting sequence may require checking edges distant from the uncolored edge. For example, edge e_1 may belong to a color-c_1 cycle which extends far beyond the vicinity of e_1.[1]

To sidestep this issue, we develop a procedure CUT to remove edges, thereby breaking long paths and allowing augmenting sequences to be locally checkable. At the same time, we must ensure that the collection of edges removed by CUT (the

[1] A closely related computational model called SLOCAL was developed in [29], where each vertex sequentially (in some order) reads its r-hop neighborhood for some radius r and then produces its answer. If a problem has an SLOCAL algorithm with radius r, then it can be solved in $O(r \log^2 n)$-rounds in the LOCAL model. Again, augmenting sequences need not lead to SLOCAL algorithms because of the need to check faraway edges.

``leftover graph'') has arboricity $O(n a)$. This can be viewed as an online load-balancing problem, where the load of a vertex is the number of directed neighbors which get removed. It is similar to the load-balancing problem encountered in [58], where paths come in an online fashion and internal edges need to be removed. Here, we encounter rooted trees instead of paths, and we need to remove edges to disconnect the root from all the leaves.

If edges were removed independently, then the load of a vertex would be stuck at $a(\log n)$ due to the concentration threshold. To break this barrier, as in [58] we randomly remove edges incident to vertices with small load. We show that throughout the algorithm, the root-leaf paths of the trees always contain many such vertices; thus, long paths are always killed with high probability.

Palette partitioning for list coloring. The final step is to recolor the leftover edges using an additional $O(n a)$ colors. For ordinary forest decomposition, this is nearly automatic due to our bound on the arboricity of the leftover graph. For list coloring, we must reserve a small number of backup colors for the leftover 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 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^t is the minimum number of pseudo-forests into which a graph can be decomposed. A result of Hakimi [36] shows that pseudo-arboricity is given by an analogous formula to the Nash-Williams formula for arboricity, namely,

$$a^t(G) = \max_{H \subseteq G,\ |V(H)| \geq 1} \left\lceil \frac{|E(H)|}{|V(H)|} \right\rceil .$$

In particular, as noted in [49], loopless multigraphs have $a^t \leq a \leq 2a^t$, and simple graphs have $a \leq a^t + 1$.

There is an equivalent, completely local, characterization of pseudo-arboricity: a k-orientation of a graph is an orientation of the edges where every vertex has outdegree at most k. It turns out that $k = a^t$ is the minimum value for which such a k-orientation exists. In a sense, a^t is a more fundamental graph parameter than a, and the problems of pseudo-forest decomposition, low outdegree orientation, and maximum density subgraph are better understood than forest decomposition. For example, maximum density subgraph has been studied in many computational models, e.g., [5, 6, 13, 16, 22, 26, 31, 33, 39, 45, 49, 53, 54]. Low outdegree orientation has been studied in the centralized context in [11, 14, 25, 35, 40, 41].

There has been a long line of work on LOCAL algorithms for $(1+\epsilon)a^t$-orientation [23, 27, 32, 37, 57].[2] Most recently, [57] gave an algorithm in $O(\log^2 n/\epsilon^2)$ rounds for $\epsilon a^t \geq 32$; this algorithm also works in the CONGEST model, which is a special case of the LOCAL model where messages are restricted to $O(\log n)$ bits per round.

[2] For many of these works, the graph was implicitly assumed to be simple, and the algorithm provides a $(1+\epsilon)a$-orientation; since simple graphs have $a^t \leq a \leq a^t + 1$, this is a minor adjustment of the parameters. Also note that [37] claims a $(1+\epsilon)a$-orientation in multigraphs, but the algorithm actually provides a $(1+\epsilon)a^t$-orientation.

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^t and $\varepsilon \in (0, 1)$, there is a LOCAL algorithm for obtaining $\lfloor a^t(1+\varepsilon)\rfloor$-orientation in $O(\frac{\log^3 n}{\varepsilon})$ rounds w.h.p.

Note in particular the linear dependency on $1/\varepsilon$. For example, if $a^t = \Omega(\sqrt{n})$, we can get an $(a^t + 1)$-orientation in $O(\sqrt{n})$ rounds, while previous results require $\Omega(n)$ rounds. Notably, the $1/\varepsilon^2$ factor in [57] comes from the number of iterations needed to solve the LP. In addition to being a notable result on its own, Theorem 1.1 provides a simple warmup exercise for our more advanced forest-decomposition algorithms.

1.4. Star-arboricity and list-star-arboricity 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, 2, 3]. We analogously define a^{list}_{star} as the smallest value k such that an LSFD exists whenever each edge has a palette of size k (this has not been studied before to our knowledge). For general loopless multigraphs, it can be shown that $a_{star} \le 2a^t$ (see Proposition E.2) and $a^{list}_{star} \le 4a^t$ (see Theorem E.3). In simple graphs, Alon, McDiarmid, and Reed [2] showed that $a_{star} \le a + O(\log a)$.

Our algorithms for star-forest decomposition in simple graphs come from a strengthened version of the construction of [2]. To briefly summarize, consider some fixed k-orientation of the graph. For each color c, mark each vertex as a c-center independently with some probability p. Then finding a star-forest decomposition reduces to finding a perfect matching, for each vertex u, between the colors c for which u is a not a c-center, and the out-neighbors of u which are c-centers.

In the general LSFD case, we show that these perfect matchings exist, and can be found eficiently, when $nk = a(\log a)$. This is based on more advanced analysis of concentration bounds for the number of c-leaf neighbors. In the ordinary SFD case, instead of perfect matchings, we obtain near-perfect matchings, leaving εnk unmatched edges per vertex. These leftover edges can later be decomposed into $2\varepsilon nk$ stars. This gives a bound $nk = a(\sqrt{\log a} + \log a)$.

In addition to being powerful algorithmic results, these also give two new combinatorial bounds.

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

For lower bounds, [2] showed that there are simple graphs with $a_{star} = 2a$ and $a = 2^{a(a^2)}$, while [1] showed that there are simple graphs where every vertex has degree exactly $d = 2a$ and where $a_{star} \ge a + a(\log a)$. These two lower bounds show that the dependence of a_{star} on a and a 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 $\sqrt{\log a}$ cannot be replaced by a function $o(\sqrt{\log a})$.

1.5. Preliminaries. Our algorithms will frequently use global parameters such as n, a, a^t, m, n, a. As usual in distributed algorithms, we always suppose that we are given some globally known upper bounds on such values as part of the input; 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 $n < 1/n$ (since, in the LOCAL model, we can simply read in the entire graph in $O(n)$ rounds), so we assume throughout that $n \ge (1/n, 1/2)$.

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

Concentration bounds. At several places, we refer to Chernoff bounds on sums of random variables. To simplify formulas, we define $F_+(u, t) = \left(\frac{e^a}{(1+a)^{1+a}}\right)^u$ where $a = t/u - 1$, i.e., the upper bound on the probability that a Binomial random variable with mean u takes a value as large as t. Some well-known bounds are $F_+(u, t) \leq (eu/t)^t$ for any value $t \geq 0$, and $F_+(u, u(1+a)) \leq e^{-ua^2/3}$ for $a \in [0, 1]$. Chernoff bounds also apply to certain types of negatively correlated random variables; for instance, we have the following standard result (see, e.g., [48]).

Lemma 1.3. Suppose that $X_1, \dots, X_k$ are Bernoulli random variables and for every $S \subseteq [k]$ it holds that $\Pr(\bigwedge_{i \in S} X_i = 1) \leq q^{|S|}$ for some parameter $q \in [0, 1]$. Then, for any $t \geq 0$, we have $\Pr(\sum_{i=1}^k X_i \geq t) \leq F_+(kq, t)$.

Lovász local lemma (LLL). The LLL is a general principle in probability theory which states that for a collection of "bad" events $B = \{B_1, \dots, B_t\}$ in a probability space, where each event has low probability and is independent of most of the other events, there is a positive probability that none of the events B_i occurs. It often appears in the context of graph theory and distributed algorithms, wherein each bad event is some locally checkable property on the vertices.

We will use a randomized LOCAL algorithm of [17] to determine values for the variables to avoid the bad events. This algorithm runs in $O(\log n)$ rounds under the criterion $epd^2 \leq 1 - a$ (1), where p is the maximum probability of any bad event, and d is the maximum number of bad events B_j dependent on any given B_i. Note that this is stronger than the general symmetric LLL criterion, which requires merely $epd \leq 1$.

Network decomposition. A (D, i)-network decomposition is a partition of the vertices into i classes such that every connected component in every class has strong diameter at most D. Each connected component within each class is called a cluster. An $(O(\log n), O(\log n))$-network decomposition can be obtained in $O(\log^2 n)$ rounds by randomized LOCAL algorithms [4, 20, 44].

We also consider a related notion of (D, a)-stochastic network decomposition, which is a randomized procedure for selecting an edge-set $L \subseteq E$ such that (i) the connected components of the graph $G^e = (V, L)$ have strong diameter at most D, and (ii) any given edge goes into L with probability at least $1 - a$. There is an algorithm in [46] for producing an $(O(\frac{\log n}{a}), a)$-stochastic network decomposition in $O(\frac{\log n}{a})$ rounds of the LOCAL model.

1.6. Basic forest decomposition algorithms. We list here some simpler algorithms for forest decomposition. These will be important building blocks later and may also be of independent combinatorial and algorithmic interest.

Theorem 1.4. Let $t = r(2+\eta)a^t r$ for $\eta \in (0, 1)$. There are deterministic $O(\frac{\log n}{r})$-round algorithms for obtaining the following decompositions of G:

- A partition of the vertices into $k = O(\frac{\log n}{r})$ 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 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.
- A list-forest decomposition when every edge has a palette of size t.

Theorem 1.5. If every edge has a palette of size $r(4+\eta)a^t$ for $\eta \in (0,1)$, then an LSFD can be computed in $\min\{O(\frac{\log^3 n}{\eta}), O(\frac{\log n \log^t m}{r})\}$ rounds w.h.p.

Proposition 1.6. Suppose we are given some k-FD of the multigraph G of arbitrary diameter. For any $\eta > 0$, there is an $O(\frac{\log n}{r})$-round algorithm for computing a $(k + \text{final})$-FD of diameter $D \le O(\frac{\log n}{\eta})$ w.h.p. If $a \ge a \min\{\frac{\log n}{\eta}, \frac{\log a}{\eta^2}\}$, we can get $D \le O(\frac{1}{\eta})$ w.h.p. with the same runtime.

The first two results of Theorem 1.4 were shown (with slightly different terminology) in the H-partition algorithm of [7]; we also provide full proofs in Appendix A. The proof of Theorem 1.5 appears in Appendix B. The proof of Proposition 1.6 appears in Appendix C, along with a more general result we will need later for reducing the diameter of list forest decompositions.

2. Algorithm for low outdegree orientation.

We now discuss a LOCAL algorithm for $(1+\eta)a^t$-orientation, based on augmenting sequences and network decomposition. Consider a multigraph G of pseudo-arboricity a^t. For the purposes of this section only, we allow G to contain loops. Following [32], we can "augment" a given edge-orientation σ by reversing the edges in some directed path. We begin with the following observation, which is essentially a reformulation of results of [32, 36] with a more careful counting of parameters.

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

Proof. For $i \ge 0$, let V_i denote the vertices at distance at most i from v. If all vertices in V_i have outdegree at least $a^t(1+\eta)$, then for each $j < i$ we can count the edge-set $E(V_{j+1})$ in two ways. First, each vertex in V_j has outdegree at least $a^t(1+\eta)$, and these edges have both endpoints in V_{j+1}, so $|E(V_{j+1})| \ge |V_j| a^t(1+\eta)$. On the other hand, by definition of pseudo-arboricity, we have $|E(V_{j+1})| \le |V_{j+1}| a^t$. Putting these two observations together, we see that

$$|V_{j+1}| \ge |V_j| t(1+\eta) \qquad \text{for } j = 0, \dots, i-1.$$

Since $V_0 = \{v\}$, by telescoping products this implies that $|V_i| \ge (1+\eta)^i$. Since clearly $|V_i| \le n$, we must have $i \le \log_{1+\eta} n \le O(\frac{\log n}{\eta})$. ∎

For the purposes of our algorithm, the main significance of this result is that we can locally fix a given edge-orientation. For a given parameter $\eta > 0$, let us say that a vertex v is overloaded with respect to a given orientation σ if the outdegree of v is strictly larger than $\lfloor a^t(1+\eta) \rfloor$; otherwise, if the outdegree is at most $\lfloor a^t(1+\eta) \rfloor$, it is underloaded. We summarize this as follows.

Proposition 2.2. Suppose multigraph G has an edge-orientation σ, and let $U \subseteq V$ be an arbitrary vertex set. Then there is an edge-orientation σ^e with the following properties:

- σ^e agrees with σ outside $N^r(U)$ where $r = O(\frac{\log n}{\eta})$.
- All vertices of U are underloaded with respect to σ^e.

t All vertices which are underloaded with respect to i remain underloaded with respect to i^e.

Proof. Following [32], consider the following process: while some vertex of U is overloaded, we choose any arbitrary such vertex $v \in U$. We then find a shortest directed path $v, v_1, \ldots, v_r$ from v where vertex v_r has outdegree strictly less than $a^t(1+\eta)$. Next, we 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.

This process never creates a new overloaded vertex while decreasing the outdegree of v at each step. Thus, after a finite number of steps, it terminates, and all the vertices in U are underloaded. Also, each step of this process only modifies edges within distance $r \le O(\frac{\log n}{\eta})$ of some vertex of U. Hence, i^e agrees with i for all other edges. ∎

We remark that this type of "local patching" has also been critical for other LOCAL algorithms, such as the α-vertex-coloring algorithm of [28] or the $(\alpha+1)$-edge-coloring algorithm of [12]. We next use network decomposition to extend this local patching into a global solution via the following Algorithm 2.1. Here, K is a universal constant to be specified.

Algorithm 2.1 Low-degree_Orientation_Decomposition(G).

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

Theorem 2.3. Algorithm 2.1 runs in $O(\log^3 n/\eta)$ rounds. At the termination, the edge-orientation i has maximum outdegree $\lfloor a^t(1+\eta) \rfloor$ w.h.p.

Proof. For line 2, we use the algorithm of [20] to obtain the network decomposition for G^{2R} in $O(R \log^2 n)$ rounds. Algorithm 2.1 processes each cluster U of a given class simultaneously, and we also define $U^e = N^R(U)$. From Proposition 2.2, it is possible to modify i within U^e for suficiently large K such that all vertices in U become underloaded, and no additional overloaded vertices are created. This can be done by having some "leader" vertex in each cluster U read in the neighborhood $N^R(U)$ and choose a modified i^e and broadcast it to the other nodes in the cluster.

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 in G is at most $2R$. So each cluster U has weak diameter at most $O(R \log n)$, and also the balls U_1^e and U_2^e must be disjoint for any two clusters U_1 and U_2 of the same class. 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 $O(R \log^2 n) = O(\log^3 n/\eta)$. ∎

Algorithm 2.1 can be viewed as part of a family of algorithms based on network decomposition described in [29]. (In the language of [29], the algorithm can be implemented in SLOCAL with radius $r = O(\frac{\log n}{\eta})$.) However, we describe the algorithm explicitly to keep this paper self-contained, and because we later need a more general version of Algorithm 2.1.

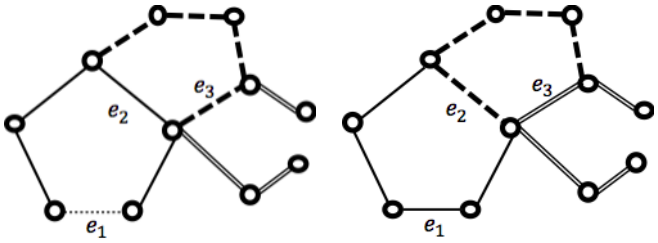

Fig. 1. An illustration of an augmenting sequence before and after the augmentation process.

We will use the same overall strategy for forest decomposition, but we will encounter two 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 due to the possibility of long cycles. To circumvent this, we must remove edges at each step to destroy these cycles. These leftover edges will need some postprocessing steps at the end.

3. Augmenting sequences for list-coloring. We now show our main structural result on augmenting sequences. Given a partial LFD φ of multigraph 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 $C(e, c) = \mathrm{t}$.

We define an augmenting sequence with respect to φ to be a sequence $P = (e_1, e_2, \dots, e_\ell, c)$, for edges e_i and color c, satisfying the following four conditions:

(A1) $e_i \in C(e_{i-1}, \varphi(e_i))$ for $2 \le i \le \ell$.
(A2) $e_i \notin C(e_j, \varphi(e_i))$ for every i and j such that $j < i - 1$.
(A3) $C(e_\ell, c) = \mathrm{t}$.
(A4) $\varphi(e_{i+1}) \in Q(e_i)$ for each $i = 1, \dots, \ell - 1$ 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 φ^e to be a new (partial) coloring obtained by setting $\varphi^e(e_i) = \varphi(e_{i+1})$ for $1 \le i \le \ell - 1$, and $\varphi^e(e_\ell) = c$, and $\varphi^e(e) = \varphi(e)$ for all other edges $e \in E \setminus \{e_1, \dots, e_\ell\}$. See Figure 1.

Lemma 3.1. For an augmenting sequence P with respect to φ, the augmentation φ^e remains a partial list-forest decomposition.

Proof. In this proof, the notation $C(e, b)$ always refer to the cycles with respect to the original coloring φ.

We first claim that $\varphi^e(e_i) = \varphi^e(e_{i+1})$ for $i = 1, \dots, \ell - 1$. For $i = \ell - 1$, this follows from (A3) since $C(e_\ell, \varphi^e(e_\ell)) = C(e_\ell, c) = \mathrm{t}$ while $C(e_\ell, \varphi^e(e_{\ell-1})) = C(e_\ell, \varphi(c_\ell)) = \{e_\ell\}$. For $i < \ell - 1$, suppose for contradiction that $\varphi^e(e_i) = \varphi^e(e_{i+1})$, i.e., $\varphi(e_{i+1}) = \varphi(e_{i+2})$. By (A1) applied at $i+2$, we have $e_{i+2} \in C(e_{i+1}, \varphi(e_{i+2})) = C(e_{i+1}, \varphi(e_{i+1})) = \{e_{i+1}\}$. So $e_{i+2} = e_{i+1}$. But then (A1) applied at $i + 1$ would give $e_{i+2} = e_{i+1} \in C(e_i, \varphi(e_{i+1}))$, which contradicts (A2).

Now for each $i = 1, \dots, \ell + 1$, define φ_i to be the coloring obtained by setting $\varphi_i(e_j) = \varphi^e(e_j)$ for all $j = i, \dots, \ell$, and setting $\varphi_i(e) = \varphi(e)$ for all other edges e. Thus, $\varphi = \varphi_{\ell+1}$ and $\varphi^e = \varphi_1$.

By (A3), φ_ℓ does not have a cycle. So if φ^e is not a partial LFD, let $i \in \{1, \dots, \ell - 1\}$ be maximal such that φ_i has a cycle. Since φ_i and φ_{i+1} differ only at edge e_i, it must be that φ_{i+1} has a path p_1 on color $\varphi^e(e_i)$ from u to v. Since $\varphi_{i+1}(e_{i+1}) = \varphi^e(e_{i+1}) =$

Algorithm 3.1 Find_Almost_Augmenting_sequence(e_{init}).

```
E_1 = {e_init}
for i = 1...n do
    E_{i+1} ← E_i.
    for each e ∈ E_i and each color c ∈ Q(e) do
        if C(e, c) = t then
            for each edge e^e ∈ C(e, c) ∖ E_i which is adjacent to an edge of E_i do
                Set E_{i+1} ← E_{i+1} ∪ {e^e} and i(e^e) ← e.
        else
            Return the almost augmenting sequence P = (e_init, ..., i(i(e)), i(e), e, c).
```

$i^e(e_i)$, path p_1 does not contain edge e_{i+1}. Since i_{i+2} and i_{i+1} only differ at edge e_{i+1}, where $e_{i+1} \notin p_1$, path p_1 is also present in i_{i+2}. On the other hand, by (A1) we have $e_{i+1} \in p_2$ for path $p_2 = C(e_i, i(e_{i+1}))$. By (A2), none of the edges $e_{i+2}, \dots, e_l$ were on p_2, and hence p_2 remains in i_{i+2}.

We must have $p_1 = p_2$ since $e_{i+1} \notin p_1$ but $e_{i+1} \in p_2$. Thus i_{i+2} contains two distinct paths of the same color from u to v. This contradicts the maximality of i. ∎

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

Theorem 3.2. Given a partial LFD of a multigraph G where every edge has palette size $(1+\eta)a$ and an uncolored edge e, there is an augmenting sequence $P = (e, e_2, \dots, e_l, c)$ from e where $e_2, \dots, e_l \in N^r(e)$ for $r = O(\frac{\log n}{\eta})$.

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 LFD i, and every edge has palette size $(1+\eta)a$, and let $L \subseteq E$ be an arbitrary edge-set. Then there is a partial LFD i^e with the following properties:

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

Proof (assuming Theorem 3.2). We can go through each uncolored edge $e \in L$ in an arbitrary order, obtain an augmenting sequence P from Theorem 3.2, and then replace i with its augmentation with respect to P. This ensures that e is colored and does not de-color any edges. Furthermore, since P lies inside $N^r(e)$, it does not modify any edges outside $N^r(L)$. At the end of this process, all edges in L have become colored, and none of the edges outside $N^r(L)$ have been modified. ∎

To prove Theorem 3.2, we first construct a weaker object called an almost augmenting sequence, which is a sequence satisfying properties (A1), (A3), and (A4) but not necessarily (A2). Algorithm 3.1 as follows finds an almost augmenting sequence starting from a given edge e_{init}.

Lemma 3.4. Algorithm 3.1 terminates within $O(\frac{\log n}{\eta})$ iterations.

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 whose palette contains color c. An edge only gets added to E_{i+1} if it is adjacent to an edge in E_i. Thus, the graph spanned by (V_i, E_i) is connected, and $E_i \subseteq N^{i-1}(e_{init})$.

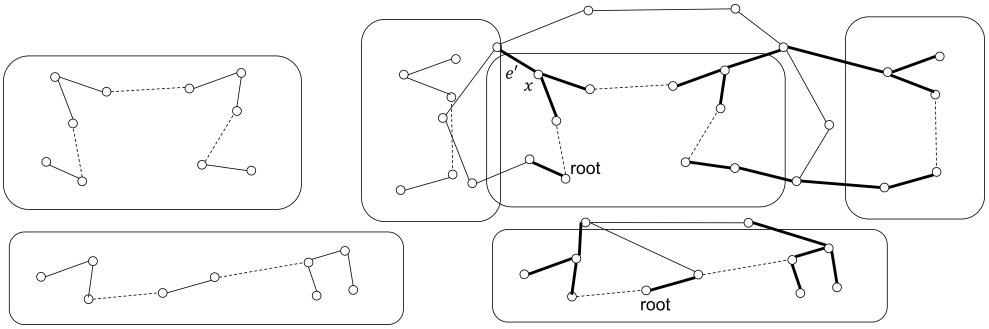

Fig. 2. In the leftmost subfigures, the ovals represent the connected component of G_c. The rightmost subfigures shows the graph H_c; here, the bold edges are guaranteed to be added to E_{i+1}.

Let us assume we are at some iteration i and the algorithm has not terminated, i.e., $C(e,c) = t$ for all $e \in E_{i,c}$. For each color c, let r_c be the number of connected components in the subgraph $G_c = (V_i, E_{i,c})$. Consider forming a graph H_c on vertex set V with edge-set given by $E(H_c) = \bigcup_{e' \in E_{i,c}} C(e,c)$. This is a forest consisting of c-colored edges. By our assumption that $C(e',c) = t$ for all $e \in E_{i,c}$, any vertices in V_i which are connected in G_c are also connected in H_c. Thus, H_c has at most r_c components, and if we choose some arbitrary rooting of forest H_c, then at most r_c vertices in V_i can be root nodes.

Now, consider any such nonroot node $x \in V_i$, with parent edge $e^* \in H_c$. We have $e^* \in C(e,c)$ for some edge $e \in E_{i,c}$. Since x is an endpoint of e^* and is also an endpoint of an edge in E_i, the new edge e^* gets added to E_{i+1} at iteration i, unless it was already part of E_i. This holds for every nonroot node in H_c, so E_{i+1} contains at least $|V_i| - r_c$ edges from H_c, which are c-colored. (See Figure 2.) We sum over colors c to get

$$|E_{i+1}| \geq \sum_{c \in C} (|V_i| - r_c).$$

To bound this sum, consider an arbitrary spanning tree T of the connected graph (V_i, E_i), where $|T| = |V_i| - 1$. Since T is a tree, we have $|T \cap E_{i,c}| \leq |V_i| - r_c$ for each color c, and so,

$$|E_{i+1}| \geq \sum_{c \in C} (|V_i| - r_c) \geq \sum_{c \in C} |T \cap E_{i,c}| = \sum_{e \in T} |Q(e)| \geq |T| \cdot (1+\eta)\alpha = (1+\eta)\alpha(|V_i| - 1).$$

Since $|V_1| = 2$, this implies $|E_2| \geq (1+\eta)\alpha$. For iteration $i > 1$, note that by definition of arboricity, we have $|E_i|/(|V_i| - 1) \leq \alpha$, and so

$$|E_{i+1}| \geq (1+\eta)\alpha \cdot |E_i|/\alpha = (1+\eta)|E_i|.$$

Hence $|E_{\ell+1}| \geq (1+\eta)^\ell \alpha$ for each $\ell \geq 1$. The overall graph has $m \geq n\alpha$ edges, so the process must terminate by iteration $\ell = \lfloor \log_{1+\eta} n \rfloor$. ∎

Note that if Algorithm 3.1 terminates by iteration $\ell = O(\frac{\log n}{\eta})$, then the sequence has length ℓ, and all edges are within distance ℓ of the starting edge e_{init}. We can then 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^e, then there exists an augmenting sequence from e to e^e which is a subsequence of P.

Proof. We show this by induction on l. If $l = 0$, it holds vacuously. Otherwise, consider an almost augmenting sequence $P = (e_1, e_2, \ldots, e_l, c)$ with $e_1 = e$ and $e_l = e^e$. If P satisfies (A2), we are done. If not, suppose that $e_i \in C(e_j, i(e_i))$ for $j < i - 1$. Then, $P^e = (e_1, \ldots, e_j, e_i, \ldots, e_l, c)$ is also an almost augmenting sequence of length $l^e < l$ which is a subsequence of P. By an induction hypothesis, it has a subsequence P^{ee} which is an augmenting sequence, which in turn is a subsequence of P. ∎

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

4. Local forest decompositions via augmentation.

Algorithm 4.1 is a high-level description of our forest decomposition algorithm, in terms of a parameter R, a constant K^e, and a subroutine CUT (all to be specified).

The subroutine CUT(U, R) removes edges from the graph so as to break all long monochromatic paths in the vicinity of U. We call the set of removed edges from all CUT(U, R) instances the leftover edges, denoted by E^{leftover}; the graph induced on them is the leftover graph, denoted G^{leftover}. We also define the main edges by $E^{\text{main}} = E \setminus E^{\text{leftover}}$ (i.e., the edges never removed by CUT) and the induced graph on them, the main graph, by G^{main}.

Formally, for each class U, define $U^e = N^{R^e}(U)$ and $U^{ee} = N^{R+R^e}(U)$, and define $H(U)$ to be the graph on the edges in $U^{ee} \setminus U^e$. Furthermore, for each color $c \in C$, define $H_c(U)$ to be the c-colored edges in $H(U)$; note that $H_c(U)$ is a forest. We say that the execution of Algorithm 4.1 is good if, after every application of CUT(U, R), the vertex sets U^e and $V \setminus U^{ee}$ are disconnected in $H_c(U)$ for every color c. (See Figure 3.)

We will show the following main result for Algorithm 4.1.

Algorithm 4.1 Forest_Decomposition(G).

1: Initialize i with t.
2: Compute an $(O(\log n), O(\log n))$-network decomposition in $G^{2t(R+R^e)}$ for $R^e = \lceil K^e \log n / n \rceil$.
3: for each class z in the network decomposition do
4: for each component U in the class z in parallel do
5: Execute CUT(U, R).
6: Modify i within $N^{R^e}(U)$ so that edges inside $N(U)$ become colored.

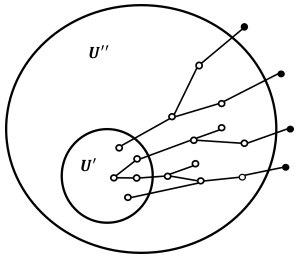

Fig. 3. $H_c(U)$ with $R = 3$. We want to disconnect U^e from all nodes with distance R from U^e (illustrated as black nodes).

Theorem 4.1. If every edge has a palette of size $(1+\eta)a$, then w.h.p. Algorithm 4.1 generates a list forest decomposition of G^{main} such that $a^t(G^{leftover}) \le \eta a$. It has the following complexity:

- If $\eta a \ge a(\log n)$, the complexity is $O(\frac{\log^4 n}{r})$ rounds.
- If $\eta a \ge a(\log n)$ and $C = \{1, \dots, a(1+\eta)\}$, the complexity is $O(\frac{\log^3 n}{r})$ rounds.
- If $\eta a \ge \log a$, the complexity is $O(\frac{a^{2/(\eta a)} \log a \log^4 n}{\eta a^2})$ rounds.
- If $\eta a \ge \log a$, the complexity is $O(\frac{\log^4 n}{r})$ rounds.

The key to the algorithm is to ensure that the CUT subroutine load-balances the number of removed out-neighbors of any vertex. We describe the implementation of CUT to achieve this, along with choice of parameter R, in section 4.1. At the end of this process, we finish with a decomposition of the leftover graph; this is summarized next in section 4.2. Putting aside the implementation of CUT for the moment, we summarize the algorithm as follows.

Theorem 4.2. Algorithm 4.1 runs in $O(R \log^2 n + \log^3 n/\eta)$ rounds. If the execution of the algorithm is good and every edge has a palette of size $(1+\eta)a$, then at the termination, ϕ is a list forest decomposition of G^{main}.

Proof. Let $R^e = R + R^e$. For line 2, we use the algorithm of [20] to get an $(O(\log n), O(\log n))$-network decomposition in the power graph G^{2R^e} in $O(R^e \log^2 n)$ rounds. Then Algorithm 4.1 colors all edges that are adjacent to or inside a cluster U of a class z (lines 4--6). Thus, if an edge uv is not removed, it will become colored when we process the first class containing u or v.

Consider some cluster U, and suppose the execution is good. The modified coloring ϕ^e can be found by some "leader" vertex in U, which reads in the neighborhood $U^{ee} = N^{R^e}(U)$. By Corollary 3.3, it is possible to modify edge colors within U^e so that all edges in U become colored for large enough constant K^e. Note that, since there are no paths in $H_c(U)$ from U to outside U^{ee}, we can check whether the coloring is acyclic by looking within U^{ee} alone.

The distance between clusters in the same class is at least $2R^e + 1$. Moreover, if u, v are adjacent in G^{2R^e}, their distance in G is at most $2R^e$. So each cluster U has weak diameter at most $O(R^e \log n)$, and also the balls $N^{R^e}(U_1)$ and $N^{R^e}(U_2)$ must be disjoint for any two clusters U_1 and U_2 of the same class. We can process each cluster independently without interfering with others. Therefore, lines 4--6, including implementation of CUT, can be simulated locally in $O(R^e \log n)$ rounds. Since there are $O(\log n)$ classes, the total running time is $O(R^e \log^2 n)$. ∎

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

Theorem 4.3. The procedure CUT can be implemented so that w.h.p. the leftover subgraph has pseudo-arboricity at most ηa, and the execution of Algorithm 4.1 is good, with the following values for parameter R:

1. $R = O(\frac{\log^2 n}{r})$ if $\eta a \ge a(\log n)$.
2. $R = O(\frac{\log n}{r})$ if $\eta a \ge a(\log n)$ and $C = \{1, \dots, a(1+\eta)\}$ (i.e. forest decomposition).
3. $R = O(\frac{a^{2/(\eta a)} \log a \log^2 n}{\eta a^2})$ if $\eta a \ge \log a$
4. $R = O(\frac{\log^2 n}{r})$ if $\eta a \ge \log a$

Theorem 4.1 will follow directly from Theorems 4.2 and 4.3. We show Theorem 4.3 here; the first two results follow from straightforward diameter-reduction algorithms.

Proof of Theorem 4.3(1). We apply Proposition 1.6 to $H(U)$ with parameter $n^e = n/(2T)$ in place of n. This reduces the diameter of each forest $H_c(U)$ to $D = O(\log^2 n/n)$ and removes an edge-set of arboricity at most $\lfloor \alpha n/(2T) \rfloor$. In particular, when $R \geq D$, the execution of Algorithm 4.1 is good. Over the T iterations of Algorithm 4.1, the arboricity of G^{leftover} is at most $T\lfloor \alpha n/(2T) \rfloor$; since $T = O(\log n)$ and $\alpha n \geq \alpha(\log n)$, this is at most αn. ∎

Proof of Theorem 4.3(2). For each color c, we choose an arbitrary root for each tree of $H_c(U)$. Next, we choose an integer J_c uniformly at random from $[N]$, where $N = \lfloor 4T/n \rfloor$, and set $R = 2N + 1 = \alpha(\frac{\log n}{r})$. Then CUT$(U, R)$ removes all edges e in each $H_c(U)$ whose tree-depth d_e satisfies $d_e \equiv J_c \mod N$. After this deletion step, each $H_c(U)$ has path length at most $2N < R$. So $V \setminus U^{ee}$ is disconnected from U^e with probability one, and Algorithm 4.1 is always good.

When the algorithm removes any edge $e = uv$, where u is the parent of v in the rooted tree of $H_c(U)$, we can orient edge e away from v in G^{leftover}. The outdegree of v in G^{leftover} is then $\sum_{i=1}^{T} \sum_{c \in C} Y_{i,c}$, where $Y_{i,c}$ is the indicator function that v has its c-colored parent edge removed when processing class i. For a subset S of $[T] \times C$, we have $\Pr(\bigwedge_{(i,c) \in S} Y_{i,c} = 1) \leq q^{|S|}$ where $q = 1/N$. Note that $|C|T \leq q \alpha n/2$, so by Lemma 1.3, the probability that the outdegree of v exceeds αn is at most $F_+(\alpha n/2, \alpha n) \leq e^{-\alpha n/6}$. When $\alpha n \geq \alpha(\log n)$, then w.h.p. every vertex has at most αn out-neighbors in the orientation. ∎

We now turn to the last two results of Theorem 4.3. We assume that $\alpha n = O(\log n)$, as otherwise we could apply Theorem 4.3(1). In particular, from our assumption that $\alpha > 1/n$, we have $\alpha \leq O(n \log n)$. The algorithm for CUT(U, R) here has two stages: an initialization procedure, which is called at the beginning of Algorithm 4.1, and an on-line procedure for a given cluster U^e.

We say a vertex u is overloaded if $L(u) \geq \alpha n$; otherwise, it is underloaded; thus, Algorithm 4.2 only modifies underloaded vertices. 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, we let $E_0(P)$ and $E_1(P)$ denote the set of underloaded and overloaded edges in P, respectively. A length-R path in $H_c(U)$ is called a live branch.

Proposition 4.4. Let $\alpha \in (0, 1/2]$. If $p \geq \frac{30\alpha \log n}{\alpha R}$, then w.h.p., either the execution of Algorithm 4.1 is good, or some live branch P has $|E_0(P)| < \alpha R$.

Algorithm 4.2 CUT(U, R).

```
1: function Initialization
2:     Using Theorem 1.4, obtain a 3α-orientation J of G
3:     For each vertex v ∈ V, set L(v) ← 0
4: function CUT(U, R) execution
5:     for each vertex v ∈ U^ee with L(v) < αn do
6:       Draw independent Bernoulli-p random variable X_v
7:       if X_v = 1 then
8:         Select an edge e uniformly at random from the outgoing edges of v under J
9:         Remove edge e from G
10:        L(v) ← L(v) + 1
```

Proof. Any path from U^e to $V \setminus U^{ee}$ has length at least R and hence will pass over some live branch. So it sufices to show that any live branch P in $H_c(U)$ during an invocation of CUT(U, R) is cut. Each underloaded edge of P gets removed with probability at least $p/(3a)$, and such removal events are negatively correlated. Thus, for $|E_0(P)| \geq aR$, the probability that P remains is at most $(1 - p/(3a))^{aR} \leq e^{-pRa/(3a)}$. By our choice of p, this is at most $e^{-30/3 \log n} \leq n^{-10}$.

Each forest $H_c(U)$ has at most n^2 live branches, and Algorithm 4.1 invokes CUT(U, R) at most $O(n \log n)$ times, and the number of colors c is at most $ma(1+n) \leq 2na^2 \leq O(n^3 \log^2 n)$. Hence, by a union bound, we conclude that either the algorithm is good or some live branch has $|E_0(P)| < aR$. ∎

Lemma 4.5. If $R \geq a\left(\frac{a^{\frac{2+4a}{na}} \log^2 n}{ar}\right)$ for some $a \in (0, 1/2]$, then p can be chosen so that Algorithm 4.1 is good w.h.p.

Proof. Let $t = na$ and $p = \frac{30a \log n}{aR}$. We set $R = \frac{Ka^{\frac{2+4a}{t}} \log^2 n}{an}$ for some constant K, and we can calculate

$$p = \frac{30a \log n}{aR} \leq nat \frac{30}{K} t \frac{1}{\log n \cdot t \cdot a^{\frac{2+4a}{t}}}. \tag{4.1}$$

Since we are assuming $na \leq O(\log n)$, we have $p \in [0, 1]$ for large enough K. By Proposition 4.4, it now sufices to show that w.h.p. $|E_1(P)| < (1 - a)R$ for all live branches P.

Consider the probability that all edges in S are overloaded where S is an arbitrary subset of the edges in a given live branch P. Since P is a path, 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 $r = F_+(Tp, t)$. Accordingly, the probability that all edges in S are overloaded is at most $r^{|S|/2}$.

Since $T \leq O(\log n)$, (4.1) implies that $pT \leq \frac{na}{10ea^{\frac{2+4a}{t}}}$ for large enough K, and therefore,

$$F_+(Tp, t) \leq \left(\frac{eTp}{t}\right)^t \leq \left(\frac{etna}{10ea^{(2+4a)/t} tna}\right)^t \leq \frac{1}{10a^{2+4a}}.$$

So we apply Lemma 1.3 with parameter $q = \frac{d}{r}$ for the random variable $|E_1(P)|$ to get

$$\Pr(|E_1(P)| > (1 - a)R) \leq F_+(R, (1 - a)R) \leq \left(\frac{er}{d(1 - a)}\right)^{(1-a)R}$$
$$\leq \left(\frac{d}{10(1 - a)a^{1+2a}}\right)^{(1-a)R} \leq (e/10)^{R/2} a^{-(1-a)(1+2a)R} \leq (0.93/a)^R.$$

Since $a \geq 1/2$ and $R \geq a(\log n)$, this is at most $\text{poly}(1/n) \cdot a^{-R}$. There are at most na^{R-1} paths of length R. By a union bound, we conclude that w.h.p. $|E_0(P)| \geq aR$ for all such paths. ∎

Proof of Theorem 4.3(3),(4). In the algorithm for CUT, each vertex removes at most na outgoing neighbors under J. Hence, $a^t(G^{\text{leftover}}) \leq na$. Given a, we choose R, p according to Lemma 4.5 so that Algorithm 4.1 is good w.h.p. For result (3), we set $a = \frac{\log a}{2 \log a}$, giving $R \leq O(a^{2/(na)} \frac{\log a \log^2 n}{an^2})$. For result (4), we set $a = 1/2$, giving $R \leq O(\frac{\log^2 n}{r})$. ∎

We remark that, by orienting edges in terms of the forest $H_c(U)$ instead of the fixed orientation J, the bound on R can be reduced to $\frac{a^{1/(n a)} \log a \log^2 n}{an^2}$; this leads to an $O(\frac{a \log a \log^4 n}{r})$-round algorithm for ordinary forest decomposition when $na \geq 3$. We omit this analysis here.

4.2. Putting everything together. 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 a $(1+\eta)a$-FD of G of diameter D, under the following conditions:

- If $na \geq 3$, then $D \geq n$, and the complexity is $O(\frac{a^2 \log a \log^4 n}{r})$.
- If $4 \geq na \geq \log a$, then $D \geq O(\frac{\log n}{\eta})$, and the complexity is $\frac{a^{O(1/n a)} \log a \log^4 n}{an^2}$.
- If $na \geq \log a$, then $D \geq O(\frac{\log n}{\eta})$, and the complexity is $O(\frac{\log^4 n}{\eta})$.
- If $n^2 a \geq a(\log a)$, then $D \geq O(\frac{1}{\eta})$, and the complexity is $O(\frac{\log^4 n}{\eta})$.
- If $na \geq a(\log n)$, then $D \geq O(\frac{1}{\eta})$, and the complexity is $O(\frac{\log^3 n}{r})$.

Proof. The first step for all these results is to apply Theorem 4.1, where each edge is given the palette $\{1, \dots, a + \lfloor na/10 \rfloor\}$. Then G^{leftover} has pseudo-arboricity at most $k^e = \lfloor na/10 \rfloor$, and Theorem 1.4(3) yields a $\lceil 2.01 k^e \rceil$-FD of these leftover edges. Taken together, these give a k-FD of G for $k = a + \lceil 2.01 \lfloor na/10 \rfloor \rceil + \lfloor na/10 \rfloor$.

The runtime bounds follow immediately from the four different cases of Theorem 4.1.

This immediately gives us the first result in our list. For the next four results, we apply Proposition 1.6 to convert this into a $k + \lfloor na/10 \rfloor$-FD of G, with the given bounds on the diameter. ∎

For list-coloring, we will combine the main graph and leftover graph by partitioning the color-space C. Specifically, each vertex v will choose a color set $C_v \subseteq C$; we also write $\mathcal{C}$ for the ensemble of values $C_u : u \in V$. Given $\mathcal{C}$, we define a new color palette for each edge $e = uv$ by

$$Q^{\text{main}}(uv) = Q(uv) \cap C_u \cap C_v,$$
$$Q^{\text{leftover}}(uv) = Q(uv) \setminus (C_u \cap C_v).$$

We can now describe two main algorithms for this type of color partition.

Theorem 4.7. Suppose that each edge has a palette of size $(1+\eta)a$. Then, w.h.p., we can choose $\mathcal{C}$ in $O(\frac{\log n}{r})$ rounds with the following palette sizes:

- If $na \geq a(\log n)$, every edge e has $|Q^{\text{main}}(e)| \geq (1+\eta/2)a$ and $|Q^{\text{leftover}}(e)| \geq \eta a/20$.
- If $n^2 a \geq a(\log a)$, every edge e has $|Q^{\text{main}}(e)| \geq (1+\eta/2)a$ and $|Q^{\text{leftover}}(e)| \geq \eta^2 a/200$.

Proof. For the first result, we independently draw a $(O(\frac{\log n}{\eta}), \frac{\eta}{10})$-stochastic network decomposition L_c for each color c, and for each connected component U in the graph (V, L_c) we draw a Bernoulli-$(\eta/10)$ random variable $X_{c,U}$. Then each vertex $v \in U$ has $c \in C_v$ if and only if $X_{c,U} = 0$.

Now consider some edge $e = uv$ and color $c \in Q(e)$. If $e \in L_c$, then vertices u and v are in the same component U of (V, L_c), so c is in $Q^{\text{main}}(e)$ or $Q^{\text{leftover}}(e)$ depending on the value $X_{c,U}$. Thus, c goes into $Q^{\text{main}}(e), Q^{\text{leftover}}(e)$ with probability at least $(1 - \eta/10)(1 - \eta/10) \geq 1 - \eta/5$ and $(1 - \eta/10)\eta/10 \geq \eta/11$, respectively.

Since each color operates independently, Chernoff bounds imply that $|Q^{main}(e)|$ and $|Q^{leftover}(e)|$ have respective sizes at least $a(1+n/2)$ and $na/20$ with probability at least $1-e^{-a(na)}$. There are $m \geq na$ edges, so the desired bounds hold with probability at least $1-nae^{-a(na)}$; since $n > 1/n$, this is $1-1/\mathrm{poly}(n)$ for $na \geq a(\log n)$.

For the second result, we draw an independent Bernoulli-$(n/10)$ random variable $X_{c,v}$ for each color c and vertex v, and we place c in C_v if and only if $X_{c,v} = 0$. For any edge e, the expected size of $Q^{main}(uv)$ is $a(1+n)(1-n/10)(1-n/10) \geq a(1+n/5)$, and the expected size of $Q^{leftover}(e)$ is $a(1+n)(n/10)(n/10) \geq n^2a/100$. We can use the LLL algorithm of [17], where each edge e has a bad-event B_e that $|Q^{main}(e)| < (1+n/2)a$ or $|Q^{leftover}(e)| < n^2a/200$. When $n^2a \geq a(\log a)$, a straightforward Chernoff bound shows $p \geq a^{-11}$. Also, B_e affects at most $d = 2a$ other bad-events (corresponding to its neighboring edges). So the criterion $pd^2 < 1$ holds, and the LLL algorithm runs in $O(\log n)$ rounds. ∎

These give the following final results for LFD.

Theorem 4.8. Suppose that G is a multigraph where each edge has a palette of size $(1+n)a$. We can obtain an LFD of G of diameter D w.h.p. under the following conditions:

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

Proof. For the first result, let $n^e = n/1000$. We begin by applying Theorem 4.7, obtaining palettes $Q^{main}, Q^{leftover}$. We then apply Theorem 4.1 with respect to palettes Q^{main}; given our bound $|Q^{main}(e)| \geq (1+n/2)a \geq (1+n^e)a$ for all e and $n^ea \geq a(\log n)$, this produces an LFD i of G^{main}, along with a leftover graph with pseudo-arboricity at most n^ea.

Next, we apply Proposition C.1 to i with parameter n^e to obtain an edge-set $E^{ee} \subseteq E^{main}$ such that $a(E^{ee}) \leq \lceil n^ea \rceil$ and i has diameter $O(\frac{\log n}{r})$ on $E^{main} \setminus E^{ee}$. Finally, we apply Theorem 1.5 to edge-set $E^{ee} = E^{leftover} \cup E^{ee}$ to get an LSFD i^{ee} of E^{ee} with palettes $Q^{leftover}$; note that $a^t(E^{ee}) \geq a^t(E^{leftover}) + a^t(E^{ee}) \geq 2\lceil n^ea \rceil$ and $|Q^{leftover}(e)| \geq na/20 \geq 5a^t(E^{ee})$ for all $e \in E^{ee}$.

Now consider the coloring $i^{\vee}$ defined by setting

$$i^{\vee}(e) = \begin{cases} i^{ee}(e) & \text{if } e \in E^{ee}, \\ i(e) & \text{if } e \in E \setminus E^{ee}. \end{cases}$$

We claim that any component of any color c can only contain edges from E^{ee} or $E \setminus E^{ee}$, but not both. For suppose some vertex v has c-colored edges vu, vu^{ee} in $E \setminus E^{ee}, E^{ee}$, respectively. Since $i(vu) = i^{\vee}(vu)$, we have $c \in Q^{main}(vu) \subseteq C_v$ but since $i(vu^{ee}) = i^{ee}(vu^{ee})$, we have $c \in Q^{leftover}(vu^{ee}) \subseteq C \setminus C_v$. This is a contradiction.

In particular, $i^{\vee}$ is an LFD of the full graph G, and the diameter of $i^{\vee}$ is the maximum of the diameters of i on $E \setminus E^{ee}$ and of i^{ee} on E^{ee}.

The second result is completely analogous, except we set $n^e = n^2/1000$. ∎

5. Star-forest decomposition for simple graphs. Let G be a simple graph of arboricity a. By using Theorem 1.1, we may assume that we have obtained some t-orientation J in $O(\log^3 n/n)$ rounds, where $t = \lceil (1+n)a \rceil$. We write $J(v)$ for the set of out-neighbors of each vertex v; by adding dummy directed edges as necessary, we may assume that $|J(v)| = t$ exactly.

To obtain a star-forest decomposition of the graph, consider the following process: each vertex v in the graph selects a color set $C_v \subseteq C$; again, we write $\mathbf{C}$ for the ensemble

of values $C_u : u \in V$. For each vertex v, we construct a corresponding bipartite graph $W_v(\mathcal{C})$, whose left-nodes correspond to C and whose right-nodes correspond to $J(v)$, with an edge from left-node c to right-node u if and only if $c \in Q(uv) \setminus (C_v \setminus C_u)$. We make the following key observation.

Proposition 5.1. Let $a \in \mathbb{Z}_{\geq 0}$ be a globally known parameter. If each $W_v(\mathcal{C})$ has a matching of size at least $t - a$, then in $O(1)$ rounds we can generate a partition of the edges $E = E_0 \cup E_1$, along with an LSFD i_0 of E_0, such that $a^t(E_1) \leq a$. (In particular, if $a = 0$, then i_0 is an LSFD of G.)

Proof. For each edge (c, u) in the matching M_v of $W_v(\mathcal{C})$, we set $i(vu) = c$. Thus, all color-c edges have the form vu where $c \in C_v \setminus C_u$ and $(c, u) \in M_v$. 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. Since M_v is a matching, the edges of each color c are a collection of stars on the c-centers and c-leaves. The original t-orientation J of G in turn yields a a-orientation for the residual uncolored edges. ∎

So, we need to select $\mathcal{C}$ so that every graph $W_v(\mathcal{C})$ has a large matching. The following two results show how to achieve this by random sampling; the precise details are different for list- and ordinary colorings. Given a fixed choice for C, we write W_v as shorthand for $W_v(C)$.

Lemma 5.2. Suppose that $na \geq 100(d \log a + \log a)$. 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 $1 - 1/a^{10}$ that $W_v(\mathcal{C})$ has a matching of size at least $a(1 - \eta)$.

Proof. Let us suppose that we have fixed C_v to some arbitrary value. By a slight extension of Hall's theorem, it sufices to show that any set $X \subseteq J(v)$ has at least $|X| - 2\eta a$ neighbors in W_v; equivalently, there is no pair of sets $X \subseteq J(v), Y \subseteq C_v$ with $|X| \geq 2\eta a$ and $|Y| = a(1 + 2\eta) - |X|$ such that W_v contains no edges between X and Y. We say that such a pair X, Y is bad. For any fixed X, Y with $x = |X|, y = |Y| = a(1 + 2\eta) - x$, the probability that X, Y is bad is given by

$$\binom{t-y}{a-y}/\binom{t}{a} = \left(\binom{t-y}{na}/\binom{t}{na}\right)^x \leq (1 - y/t)^{xna} \leq e^{-xyna/t} \leq e^{-xyn/2}.$$

Summing over the $\binom{a(1+\eta)}{x}$ choices for X and $\binom{a}{a(1+2\eta)-x}$ choices for Y of a given cardinality, the total probability of any bad pair X, Y is at most

$$\sum_{x=2\eta a}^{a(1+\eta)} \binom{a(1+\eta)}{x} \binom{a}{a(1+2\eta)-x} e^{-x(a(1+2\eta)-x)\eta/2}. \tag{5.1}$$

When $2\eta a \leq x \leq a/2$, the summand in (5.1) is at most

$$\binom{a(1+\eta)}{a(1+\eta)-x}\binom{a}{x-\eta a} e^{-x(a(1+2\eta)-a/2)\eta/2} \leq \binom{a(1+\eta)}{x}\binom{a}{\eta a} e^{-\eta a x/4} \leq (a(1+\eta))^x a^x e^{-\eta a x/4}$$
$$\leq (2a^2 e^{-\eta a/4})^x \leq (2a^2 e^{-25 \log a - 25 d \sqrt{\log a}})^x \leq (2a^{-23} e^{-25 d \sqrt{\log a}})^x$$
$$\leq (e^{-25 d \sqrt{\log a}})^x \leq (e^{-25 d \sqrt{\log a}})^{2\eta a} \leq (e^{-25 d \sqrt{\log a}})^{200 d \sqrt{\log a}} = a^{-5000}.$$

In a completely analogous way (but with slightly different numerical constants), the summand of (5.1) for $a/2 \leq x \leq a(1 + \eta)$ is at most a^{-2500}. Since there are at most a such summands, the overall sum is at most a^{-2499}. ∎

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

Proof. Let $s = a(1 + 100n)$. We say that C_v is good if $|Q(uv) \cap C_v| \ge s$ for all $u \in J(v)$. We first claim that C_v is good with probability at least $1 - a^{-100}$. Observe that $|Q(uv) \cap C_v|$ is a binomial random variable with mean $|Q(uv)| \le 2na$. Hence, $\Pr(|Q(uv) \cap C_v| \ge 100na) \le F_+(2na, 100na) \le (\frac{et}{100na})^{100na} \le 2^{-100na}$; by our assumption $na \ge 10^6 \log a$, this is at most a^{-10^8}. By a union bound over $u \in J(v)$, the probability that C_v is good is at least $1 - t^2 a^{-10^8} \ge 1 - a^{-100}$.

We next argue that, conditioned on a fixed choice of good C_v, there is a probability of at least $1 - a^{-11}$ that W_v has a t-matching. By Hall's theorem, it sufices to show that any set $X \subseteq J(v)$ has at least $|X|$ neighbors in W_v; define B_X to be the bad-event that this condition fails for X. By considering the exponential generating function for the number of neighbors of X, with is similar to a Chernoff bound argument, we can show that any such set X has

$$\Pr(B_X) \le \frac{(se^{-nx})^{s-x}}{(s-x)!} \left(\frac{s(1-e^{-nx})}{x}\right)^x \qquad \text{where } x = |X|.$$

(The details are somewhat involved, so we defer the proof of this fact to Proposition D.1.) We can take a union bound over the $\binom{t}{x}$ possible sets X of cardinality x to get

$$\sum_X \Pr(B_X) \le \sum_{x=1}^{t} a_x \qquad \text{for } a_x := \frac{(se^{-nx})^{s-x}}{(s-x)!} \left(\frac{s(1-e^{-nx})}{x}\right)^x \binom{t}{x}.$$

We will upper-bound a_x separately over two different parameter regimes as follows:

Case I: $a/3 \le x \le t$. Here, we have $se^{-nx} \le (2a)e^{na/3} \le (2a)e^{-10^6(\log a)/3} \le 2a^{-1000}$ and $\binom{t}{x} \le t^{t-x} \le (2a)^{t-x}$. Thus, we use the upper bound

$$a_x \le f_1(x) := (2a^{-1000})^{s-x}(s/x)^x(2a)^{t-x}.$$

Letting $g_1(x) = \log f_1(x)$, we compute the derivative $g_1'(x) = -1 + 999 \log a + \log s - \log(4x)$; since $x \le s$ and $a \ge 10^6$, this is at least $100 \log a$. Hence, in this range, we have $a_x \le e^{g_2(x)} \le e^{g_1(s) - 100(s-x)\log a}$. Since $g_1(s) = (t-s)\log(2a) \le 0$, we have $a_x \le a^{-100(s-x)}$, and the overall sum $\sum_{x=a/3}^{t} a_x$ is at most $2a^{-100(s-t)} \le 2a^{-100}$.

Case II: $1 \le x \le a/3$. Here $s(1-e^{-nx})/x \le s$ and $\binom{t}{x} \le t^x$, giving the upper bound:

$$a_x \le f_2(x) := (se^{-nx}/(s-x))^{s-x}(st)^x.$$

Again, letting $g_2(x) = \log f_2(x)$, we compute the derivative $g_2'(x) = 1 - ns + 2nx + \log t + \log(s-x)$. Now $x \le a/3$ while $a \le s, t \le 2a$; along with our bound $na \ge 10^6 \log a$, we get $g_2'(x) \le -100 \log a$ for $x \le 0$. Hence, in this range, we have $a_x \le e^{g_2(x)} \le e^{g_2(0) - 100x\log a} = a^{-100x}$, and so the overall sum $\sum_{x=1}^{a/3} a_x$ is at most $2a^{-100}$.

Putting the two cases together, we have shown that, conditional on a fixed good choice of C_v, we have $\sum_X \Pr(B_X) \le 4a^{-100}$. Since C_v is good with probability at least $1 - a^{-100}$, the overall probability that W_v has a t-matching is at least $1 - a^{-10}$. ∎

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

Theorem 5.4. Let G be a simple graph with arboricity α.

- If $\alpha \geq \alpha(\sqrt{\log \alpha} + \log \alpha)$, then we get an $\alpha(1+\eta)$-SFD in $O(\frac{\log^3 n}{\eta})$ rounds w.h.p.
- If $\alpha \geq \alpha(\log \alpha)$, and every edge has palette size $\alpha(1+\eta)$, we can get an LSFD in $O(\frac{\log^3 n}{\eta})$ rounds w.h.p.

Proof. By reparametrizing, it sufices to show that we can get an $\alpha(1+O(\eta))$-SFD of G or get an LSFD when every edge has palette size $(1+O(\eta))\alpha$.

After obtaining the orientation $\mathcal{J}$, we use the LLL algorithm of [17] to select $\mathcal{C}$; here, each vertex v has a bad-event that $W_v(\mathcal{C})$ has maximum matching size less than $t - 2\alpha\eta$. By Lemma 5.2, this event has probability at most $p = \alpha^{-10}$ and depends on $d = \alpha^2$ other such events (u and v can only affect each other if they have distance at most 2). Thus, the criterion $pd^2 \leq 1$ is satisfied, and the LLL algorithm runs in $O(\log n)$ rounds.

Having selected $\mathcal{C}$, we apply Proposition 5.1 to get a $(1+\eta)\alpha$-SFD of G, plus a leftover graph of pseudo-arboricity at most $2\eta\alpha$. We finish by applying Theorem 1.4 to get a $6.01\eta\alpha$-SFD of the leftover graph. Overall, we get a $(1+7.01\eta)\alpha$-SFD.

The second result is completely analogous, except we use Lemma 5.3 to obtain the matchings of W_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 [57] to obtain the t-orientation, and hence obtain the $\alpha(1+\eta)$-LSFD, in $O(\frac{\log^2 n}{\eta^2})$ rounds.

6. Lower bounds on round complexity. In this section, we show lower bounds for the round complexity of randomized LOCAL algorithms for forest decomposition, using the following construction. For given integer parameters $\alpha \geq 2$ and $t \geq 1$, we can form a multigraph G beginning with four named vertices x_1, x_2, y_1, x_2. We then put $\lceil \alpha/2 \rceil$ parallel edges from x_1 to x_2 and $\lceil \alpha/2 \rceil$ parallel edges from y_1 to y_2. We insert a path P_1 with $t+2$ vertices from x_1 to y_1, with α parallel edges between successive vertices on the path (thus, P_1 contains t vertices aside from x_1, y_1), and we insert a second path P_2 of $t+2$ vertices arranged in a line from x_2 to y_2 with α parallel edges. See Figure 4.

The graph G has arboricity α; to see this, consider coloring the edges x_1 to x_2 by $\{1, \dots, \lceil \alpha/2 \rceil\}$ and coloring the edges from y_1 to y_2 by $\{\lceil \alpha/2 \rceil + 1, \dots, 2\lceil \alpha/2 \rceil\}$, as well as coloring edges in P_1 and P_2 by $\{1, \dots, \alpha\}$. Also, G has $n = 2t+4$ vertices and maximum degree $\Delta = O(\alpha)$.

Proposition 6.1. For any $\alpha(1+\eta)$-forest decomposition on G, there are at most $2(t+1)\eta\alpha$ colors c where there is a c-colored edge between x_1, x_2 and also a c-colored edge between y_1, y_2.

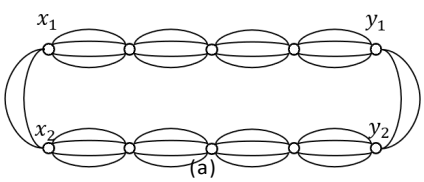

Fig. 4. An illustration of G when $\alpha = 4$ and $t = 3$.

Proof. For any two adjacent nodes u, v in G, let us say that color c appears if any of the parallel edges between u and v have color c; otherwise, color c is missing. We need to show that at most $2(t+1)n\,a$ colors appear on both x_1, x_2 and y_1, y_2.

For consecutive vertices u, v in path P_1 or P_2, the a parallel edges must receive distinct colors (otherwise, it would immediately have a cycle). Thus, u, v are missing at most na colors. Over the entire paths P_1, P_2 which have $2(t+1)$ vertices, there are at most $2(t+1)n\,a$ colors missing in total.

But now observe that if a color c appears on all consecutive vertices in the path P_1, P_2 as well as between x_1, x_2 and between y_1, y_2, then the c-colored edges would have a cycle. Hence, the only colors which can appear between x_1, x_2 and also between y_1, y_2 are the ones that are missing from some consecutive vertices in P_1, P_2, and there are at most $2(t+1)n\,a$ of them. ∎

Our lower bound will depend in a critical way on using a randomized algorithm which is oblivious, i.e., it does not use the provided vertex IDs. In particular, for an r-round oblivious randomized algorithm, the output for a given edge e is determined by the isomorphism class of $N^r(e)$.

Observation 6.2. If any randomized or deterministic LOCAL algorithm can solve a problem in r rounds, then also an oblivious randomized LOCAL algorithm can solve it in r rounds w.h.p.

Proof. Given the original algorithm A, each vertex chooses a random bit-string of length $K \log n$ and uses it as its new vertex ID for algorithm A. W.h.p., all chosen IDs are unique for K sufficiently large, and hence algorithm A succeeds (either with probability one or w.h.p., depending on whether A is randomized). ∎

Lemma 6.3. Suppose that $2(t+1)n\,a \le ra/2r$. Then any oblivious algorithm A for $a(1+\varepsilon)$-forest decomposition on G in less than $t/2$ rounds has success probability at most $\frac{a(1+\varepsilon)}{4(ra/2r - 2(t+1)n\,a)}$.

Proof. Let $\ell = a(1+\varepsilon)$. For any color $i = 1, \dots, \ell$, let X_i be the indicator function that (x_1, x_2) has an i-colored edge, and let Y_i be the indicator function 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, 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), and algorithm A is oblivious, they follow the same distribution. Thus, we denote $q_i = \mathbb{E}[X_i] = \mathbb{E}[Y_i]$ and note that $\mathbb{E}[X_i Y_i] = \mathbb{E}[X_i]\mathbb{E}[Y_i] = q_i^2$.

If A returns a forest decomposition, there are $ra/2r$ colors between x_1 and x_2 (a repeated color immediately leads to a cycle), and by Proposition 6.1 there are at most $2(t+1)n\,a$ colors i which appear between x_1, x_2 and also between y_1, y_2, i.e., colors which satisfy $X_i = Y_i = 1$. Overall, whenever A returns a forest decomposition, we have $\sum_{i=1}^{m} X_i = \sum_{i=1}^{m} Y_i = ra/2r$ and $\sum_{i=1}^{m} X_i Y_i \ge 2(t+1)n\,a$ and, in particular,

$$\sum_{i=1}^{m} X_i(1 - Y_i) \ge ra/2r - 2(t+1)n\,a.$$

Let p be the probability that A successfully returns an ℓ-forest decomposition. Taking expectations and noting that $\mathbb{E}[X_i(1 - Y_i)] = \mathbb{E}[X_i](1 - \mathbb{E}[Y_i]) = q_i(1 - q_i)$, we have

$$\sum_{i=1}^{m} q_i(1 - q_i) \ge p(ra/2r - 2(t+1)n\,a).$$

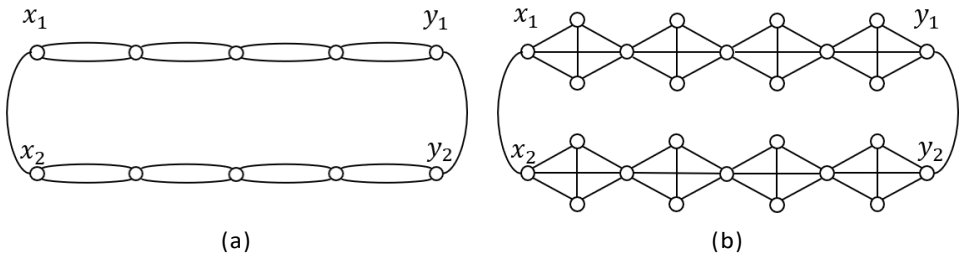

Fig. 5.

On the other hand, clearly $q_i(1 - q_i) \geq 1/4$ for all i (since $q_i \in [0, 1]$), and so

$$\sum_{i=1}^{m} q_i(1 - q_i) \geq l/4.$$

Putting the bounds together gives $p(r\alpha/2r - 2(t+1)n\alpha) \geq l/4$, which leads to the claimed bound after rearrangement. ∎

Putting these results together, we obtain the following.

Theorem 6.4. Let $\alpha, n \in Z_{\geq 2}$ and $n \in (0, 1)$ with $n\alpha \geq 1$. Any randomized algorithm for $(1 + n)\alpha$-forest decomposition on n-node graphs of arboricity α with success probability at least 0.51 requires $\Omega(\min\{n, 1/n\})$ rounds. This bound holds even on graphs of maximum degree $\Delta = O(\alpha)$.

Proof. It sufices to show this for an oblivious randomized algorithm A and where $1/n < n \leq 0.0001$. In this case, consider forming graph G with parameter $t = \lfloor 0.001/n \rfloor$; note that $t + 2 \leq 0.002/n$ due to the upper bound on n. Thus, G has at most $2t + 4 \leq 2(0.002/n) \leq n$ nodes.

Here $r\alpha/2r \geq 0.499\alpha$ since $\alpha \geq 1/n \geq 10000$, and also $2(t + 1)n\alpha \leq 2(0.002/n)n\alpha \leq r\alpha/2r$. If A runs in fewer than $t/2$ rounds, then by Lemma 6.3 it has success probability at most $\frac{\alpha(1+n)}{4(r\alpha/2r - 2(t+1)n\alpha)} \leq \frac{\alpha(1+n)}{4(0.499\alpha - 2(0.002/n)n\alpha)} \leq 0.506(1 + n) < 0.51$. ∎

We also show that it is impossible to compute α-FD in $o(n)$ rounds in simple graphs.

Proposition 6.5. In simple graphs with arboricity 2, computing a 2-forest decomposition with success probability at least 0.5 requires $\Omega(n)$ rounds.

Proof. First, construct G with parameters $\alpha = 2$ and $t = n/10$ (see Figure 5(a)). Next, replace every set of parallel edges by a copy of the complete graph K_4 (see Figure 5b). The resulting simple graph H has $6t + 8 \leq n$ nodes and arboricity 2.

Suppose now that A is an oblivious randomized algorithm which runs in $n/100$ rounds and assigns colors $\{1, 2\}$ to the edges of H. Let q denote the probability that algorithm A outputs color 1 on edge (x_1, x_2). Since the two edges are t-hops away and $n/100 < t/2$, the probability that A outputs color 1 on (y_1, y_2) is also q, independent of the color of (x_1, x_2). Therefore, with probability at least $q(1 - q) + (1 - q)q \geq 1/2$, the edges (x_1, x_2) and (y_1, y_2) receive the same color; in this case, that color does not form a forest. ∎

Appendix A. Proof of Theorem 1.4. For the first result, starting with $i = 1$, we remove vertices with degree at most $t = r(2 + n)\alpha^t r$ and add them to H_i. We continue this process, forming sets $H_1, \ldots, H_k$, until the graph is empty.

We claim that each iteration removes at least a $(n/(2+n))$ fraction of the vertices in the remaining graph. Consider the remaining graph $G^e = (V^e, E^e)$. If more than $\frac{2}{2+n}|V^e|$ vertices have degree greater than $(2+n)a^t$, then we derive a contradiction as follows:

$$2|E^e| = \sum_{v \in V^e} \deg(v) > ((2+n)a^t)|V^e| \cdot \frac{2}{2+r} = 2a^t|V^e| \geq 2(|E^e|/|V^e|) \cdot |V^e| = 2|E^e|.$$

Since the number of vertices reduces by a $(1 - a(n))$ factor in each iteration, there are at most $k = O(\log n/n)$ iterations in total.

For the second result, consider an edge $e = uv$ where $u \in H_i$ and $v \in H_j$. If $i < j$, we orient it from u to v, and likewise if $i > j$, we orient it from v to u. If $i = j$, we orient it from the vertex with a lower ID to the vertex with a higher ID. Since a vertex in H_i has at most t neighbors in $H_i \cup \cdots \cup H_k$, the outdegree is at most t. Since the edges are always oriented from a lower index partition to a higher index, with ties broken by vertex ID, the resulting orientation is acyclic.

For the third result, we arbitrarily give distinct labels to the out-edges of each vertex; this gives us a t-forest decomposition where, moreover, each tree in each forest is rooted. We can use the algorithm of [18] to get a proper 3-vertex-coloring of each tree in $O(\log^t n)$ rounds. If we assign each edge to the color of its parent, then each of the t forests decomposes into three star forests.

For the final result, consider the following process: we first fix some acyclic t-orientation. For each vertex v, we go through its out-edges $e_1, \ldots, e_t$ in some arbitrary order, and each edge e_i in turn chooses a color from its palette not already chosen by edges $e_1, \ldots, e_{i-1}$. This can be done in a greedy fashion since every edge has a palette of size t. Each vertex operates independently, so the entire process can be simulated in $O(1)$ rounds.

Appendix B. Proof of Theorem 1.5. To start, apply Theorem 1.4(1) with $n/10$ in place of n, giving partition $H_1, \ldots, H_k$ for $k = O(\frac{\log n}{r})$ where each vertex $v \in H_i$ has at most $t = r(2+n/10)a^t r$ neighbors in $H_i \cup \cdots \cup H_k$. We orient the edges from H_i to H_j if $i < j$; otherwise, we break the tie by vertex ID.

We proceed for iterations $j = k, k-1, \ldots, 1$; at iteration j, we define E_j to be the set of edges which have one endpoint in H_j and the other endpoint in $H_j \cup \cdots \cup H_k$. We also define a related line graph G^e_j as follows: each edge e in E_j corresponds to a node x_e in G^e_j. For every pair of edges e, e^e in E_j which share a common vertex in H_j, there is an edge in G^e_j between corresponding nodes x_e and x_{e^e}. Our strategy is to compute a proper list vertex-coloring of each G^e_j, where the residual palette $Q^e(x_e)$ for edge $e = uv \in E_j$ is obtained by removing from $Q(e)$ any colors chosen already by any out-edges of u or v in $E_{j+1} \cup \cdots \cup E_k$.

We first claim that this procedure gives an LSFD, where each edge is oriented toward the center of the star. Suppose edges $e = uv, e^e = u^e v$ share a color c and intersect in vertex v but that e is oriented away v. Say $v \in H_i, u \in H_j, u^e \in H_{j^e}$; necessarily $j \geq i$ and $e \in H_i$ by definitions. If $j^e \geq i$, then $e^e \in H_i$ and the graph G^e_j would have an edge between x_e and x_{e^e}, and they could not receive the same color. If $j^e < i$, then $e^e \in H_{j^e}$ and the color chosen by e would have been removed from $Q^e(x_{e^e})$ in iteration j^e. Either case is a contradiction.

Next let us argue that each graph G^e_j can be greedily colored; i.e., for each edge $e = uv$, the palette of each node x_e is larger than its degree. Suppose that u and v have a and b many neighbors in $H_{j+1} \cup \cdots \cup H_k$, respectively. Then x_e has at most $2t - 1 - (a+b)$ neighbors in G^e_j and has $|Q^e(x_e)| \geq |Q(e)| - (a+b)$. So G^e_j has maximum degree $\Delta(G^e_j) = 2t - \Delta(a^t)$, and in either case the node $x_e \in G^e_j$ satisfies

$|Q^e(x_e)| - \deg(x_e) \le |Q(e)| - (2t-1) \le r(4+n)a^t r - 2r(2+0.1n)a^t r + 1 \le 1 + r\,na^t/2r$.

Finally, let us analyze the round complexity. We use an algorithm of [21] for listvertex-coloring with palettes of size deg+na applied to each graph G^e_j. There are three parameter regimes to consider. First, if $a \le n^3$, then the algorithm of [21, Corollary 4.1], combined with the network decomposition result of [52], runs in $O(\log(1/n)) + \text{polyloglog}m$ rounds; since $m \le na \le n^4$, this is $O(\log(1/n)) + \text{polyloglog}n$.

Next, when $a > n^3$, we use the algorithm of [21, Theorem 4.1]. This algorithm requires $na(G^e_j) > (\log|V(G^e_j)|)^{1+a(1)}$. Here, $|V(G^e_j)| = m$ and $a(G^e_j) = a(a)$, and $n > 1/n$ by assumption, so $\frac{\log|V(G^e_j)|}{\log m} \le a(\frac{n}{n}\frac{\log m}{\log m}) > a(1)$. So, the algorithm runs in $O(\log^t m + \log(1/n))$ rounds.

Finally, when a is superexponentially larger than n, we can obtain an $(O(\log n), O(\log n))$-network decomposition of G^3. We then color each E_j by iterating sequentially through the classes; within each cluster, we simulate a greedy coloring of the edges. Since clusters are at distance at least 3, no edges will choose a conflicting color. The overall runtime is $O(\log^2 n)$.

Putting the three cases together, we can color each G^e_j with round complexity

$$O(\min\{\log(1/n) + \log^t m, \log(1/n) + \text{polyloglog}n, \log^2 n\}).$$

This process is repeated for iterations $j = 1, \dots, k = O(\frac{\log n}{r})$.

Appendix C. Proof of Proposition 1.6. We begin with the following algorithm to reduce diameter in a given list-forest decomposition i .

Proposition C.1. Given a multigraph G with an LFD i and $n > 0$, Algorithm C.1 runs in $O(\frac{\log n}{r})$ rounds. It partitions the edges as $E = E_0 \sqcup E_1$, where i_1 is an

Algorithm C.1 Reduce_Forest_Diameter(n , i).

Initialize E_0 w E and E^e_1, E^{ee}_1 w t.
Apply Theorem 1.4(2) to get an acyclic 3a-orientation J of G.
for each vertex v do
    Draw independent Bernoulli-1/2 random variable X_v
    if $X_v = 1$ then
        for $j = 1$ to k^e where $k^e = \lfloor na/20 \rfloor$ do
            Select an edge $e_{v,j}$ uniformly at random from the out-neighbors of v in J .
            Remove edge $e_{v,j}$ from E_0; add it to E^e_1.
            Set $i^e_1(e_{v,j}) = j$.
for each color i n C and each vertex v do
    if there is an i-colored directed path of length $10000 \log n/n$ starting from v in E_0 then
        Remove all edges of color i incident to v from E_0; add them to E^{ee}_1.
for each $j = 1, \dots, k^e$ and each vertex v do
    if there is a j-colored directed path of length $10000 \log n/n$ starting from v in E^e_1 then
        Remove all edges of color j incident to v from E^e_1; add them to E^{ee}_1.
Apply Theorem 1.4(3) to obtain a $3r\,2.01a^t$ (E^{ee}_1)r -star-forest decomposition i^{ee}_1 of E^{ee}_1.
Return E_0 and $E_1 = E^e_1 \sqcup E^{ee}_1$ along with forest decomposition $i_1 = i^e_1 \sqcup i^{ee}_1$ of E_1.

final-FD of E_1. W.h.p., both the restriction of i to E_0 and the decomposition i_1 on E_1 have diameter $D = O(\frac{\log n}{r})$.

Proof. The complexity follows from specifications of Theorem 1.4. Since the orientation is acyclic, i_1^e is a k^e-forest decomposition of E_1^e for $k^e = \lfloor a/20 \rfloor$. The two modification steps (lines 10--12 and 13--15) ensure that each forest in E_0 or E_1^e has diameter $O(\frac{\log n}{r})$, while the forests in E_1^{ee} have diameter 2. Overall, E_1 is decomposed into $k^e + 3r \cdot 2.01a^t(E_1^{ee})r$ forests; we can upper-bound this, somewhat crudely, as $k^e + 7|E_1^{ee}|$.

It remains to bound $|E_1^{ee}|$. We first claim that any edge goes into E_1^{ee} with probability at most $1/n^{24}$. Suppose that e remains in E_0 with color i. Every other color-i edge gets removed with probability at least $\frac{1}{2} \cdot \frac{\lfloor a/20 \rfloor}{3a} \geq 0.005n$, and edges at distance 2 are independent. Thus, any path of length $r \geq 10000 \log n/n$ survives with probability at most $(1 - 0.005n)^{r/2} \leq e^{-0.005 \cdot 10000 \cdot 1/2 \cdot \log n} = n^{-25}$, and there are at most n paths of color i involving any given edge.

Similarly, suppose that e goes into E_1^e with color $j \in \{1, \dots, k^e\}$. Each vertex only has deleted out-neighbors with probability 1/2. Thus, starting at the edge e, and following its directed path with respect to the j-colored edges, there is a probability of 1/2 of stopping at each vertex. Thus, the probability that e goes into E_1^{ee} is at most $2^{-10000 \log n/n} \leq n^{-24}$.

Putting these two cases together, E_1^{ee} has expected size at most $m/n^{24} \leq a/n^{23}$. By our assumption that $n > 1/n$, Markov's inequality gives $\Pr(|E_1^{ee}| \geq na/20) \leq O(\frac{a/n^{23}}{na}) \leq O(n^{-22})$. So w.h.p. E_1 uses at most $\lfloor a/20 \rfloor + 7(na/20) \leq \lfloor a \rfloor$ forests. ∎

We next show that the diameter can be reduced further in some cases. Note that Proposition C.2, in its full generality for list-forest decompositions, will not be directly required for our algorithm.

Proposition C.2. Let G be a multigraph with an LFD i. If $|C| = o(a)$ and $a \geq a \cdot \min\{\frac{\log n}{n}, \frac{o(\log a)}{n^2}\}$, there is an $O(o(\frac{\log n}{n}))$-round algorithm for obtaining an edge-set E^e such that $a^t(E^e) \leq na$ and such that w.h.p. i has diameter $D \leq O(o/n)$ in the graph $G \setminus E^e$.

Proof. First, by applying Proposition C.1 with $n/4$ in place of n, we reduce the diameter of the forests to $O(\frac{\log n}{r})$, while discarding an edge set of pseudo-arboricity at most $k^e \leq \lfloor na/4 \rfloor$. In $O(\frac{\log n}{r})$ rounds we can choose an arbitrary rooting of each remaining tree. For each color c, let U_c be the set of vertices whose depth is a multiple of $N = \lfloor 4o/n \rfloor$.

Consider the following random process: for each color c and each vertex $u \in U_c$, we independently draw an integer $J_{u,c}$ uniformly at random from $[N]$. For all vertices v of depth $J_{u,c}$ below u in the color-c tree, we remove the color-c parent edge of u. After this deletion step, every vertex $u \in U_c$ is disconnected from its distance-N descendants and is also disconnected from its distance-N ancestors. Thus, with probability one, the forests have diameter reduced to $4N \leq O(o/n)$.

It remains to bound the pseudo-arboricity of the deleted edges. For each vertex v, let B_v denote the bad-event that v has more than $k^{ee} = na/2$ deleted parent edges. If all these bad-events are avoided, then the deleted edges have pseudo-arboricity at most $k^e + k^{ee} \leq na$ from both stages.

For a vertex v and color c, let u_c be the maximum-depth ancestor of v in U_c. The color-c parent of v gets deleted if and only if $J_{u_c,c}$ is equal to the depth of v below u_c. This has probability $1/N$, so the expected number of deleted parents is at most $|C|/N \leq na/4$. Each color operates independently, so by Chernoff's bound we have $\Pr(B_v) \leq F_+(na/4, na/2) \leq e^{-na/12}$.

If $na \geq a(\log n)$, then w.h.p. none of the bad-events occur, and we are done. If $n^2 a \geq a(o \log a)$, we use the LLL algorithm of [17]. We have already calculated $p \geq e^{-na/12}$. Also, events B_v and B_w only affect each other if v and w have distance at most $2N$; hence $d \geq a^{2N} \geq a^{4o/n+2}$. So, $pd^2 \geq e^{-na/12}(a^{4o/n+2})^2 \gg 1$ for $n^2 a \geq a(o \log a)$. Each B_v depends on vertices within distance $O(o/n)$, so the LLL algorithm can be simulated in $O(\frac{o \log n}{r})$ rounds on G. ∎

To show Proposition 1.6, suppose now we are given some k-FD of G; we may assume that $k \geq O(a)$, as we can always use Theorem 1.4 to obtain a 2.01-FD. Here, we have $|C| = k$, and so $o = k/a \geq O(1)$. For the bound $D \geq O(\frac{\log n}{r})$, we can directly apply Proposition C.1. For the bound $D \geq O(\frac{1}{r})$ when a is large, we apply Proposition C.2 with $n/10$ in place of n to obtain a $k + \lfloor na/10 \rfloor$-FD, where the uncolored edges E^e have $a^t(E^e) \geq na/10$. We then use Theorem 1.4 to obtain a $3a^t(E^e)$-SFD of E^e. Overall, this gives a $k + \lfloor na \rfloor$-FD of G of diameter $O(1/n)$.

Appendix D. Concentration bound for Lemma 5.3. Here, we show the concentration bound we used in the proof of Lemma 5.3.

Proposition D.1. Let $s = a(1 + 100n), t = a(1 + n)$. Let X be any fixed subset of $J(v)$, and let $x = |X| \geq t$. Suppose the hypotheses of Lemma 5.3 hold and that C_v is fixed so that $|Q(uv) \cap C_v| \leq s$ for all $u \in X$. Then the probability that X has fewer than x neighbors in W_v is at most

$$\left(\frac{se^{-nx}}{x}\right)^{s-x}\left(\frac{s(1-e^{-nx})}{x}\right)^{x-s}.$$

Proof. For each $c \in C_v$, let z_c be the number of vertices $u \in X$ with $c \in Q(uv)$, and let Y_c be the indicator function that $c \in \bigcup_{u \in X}(Q(uv) \setminus C_u)$. Here $\Pr(Y_c = 1) = 1 - (1-n)^{z_c} \geq 1 - e^{-nz_c}$. By hypothesis we have $\sum_{c \in C_v} z_c = \sum_{u \in X} |Q(uv) \cap C_v| \leq xs$.

Consider variable $Y = \sum Y_c$, and note that X has fewer than x neighbors if and only if $Y < x$. For some parameter $a \in [0,1]$ to be determined, we define the random variable

$$i = (1-a)^{Y} = \prod_{c \in C_v} (1-a)^{Y_c}.$$

Since the colors are independent, we have

$$\text{(D.1)}\quad E[i] = \prod_{c \in C_v}(1 - a\Pr(Y_c = 1)) \geq \prod_{c \in C_v}(1 - a(1-e^{-nz_c})) = e^{\sum_{c \in C_v} \log(1-a(1-e^{-nz_c}))}.$$

Elementary calculus shows that the function $y \mapsto \log(1 - a(1 - e^{-y}))$ is negative, decreasing, and concave-up. Since $0 \leq z_c \leq x$, we thus bound it by the secant line from 0 to x, i.e.,

$$\log(1 - a(1 - e^{-nz_c})) \geq \frac{z_c}{x}\log(1 - a(1 - e^{-nx})).$$

Substituting this bound into (D.1) and using the bound $\sum z_c \leq xs$, we calculate

$$E[i] \geq e^{\sum_{c \in C_v} \frac{z_c}{x} \log(1-a(1-e^{-nx}))} = (1 - a(1 - e^{-nx}))^{\sum_{c \in C_v} z_c/x} \geq (1 - a(1 - e^{-nx}))^{s}.$$

Now by Markov's inequality applied to i, we get

$$\text{(D.2)}\qquad \Pr(Y < x) \leq E[i](1-a)^{-x} \leq (1 - a(1 - e^{-nx}))^{s}(1-a)^{-x}.$$

At this point, we set

$$a = \frac{(1 - e^{-nx})s - x}{(1 - e^{-nx})(s - x)}.$$

Clearly $a \geq 1$. We claim also that $a \geq 0$, which when substituted into (D.2) will give the claimed formula. Consider the function $f(x) = (1 - e^{-nx})s - x$. The second derivative is given by $f''(x) = -e^{-nx}n^2 s \leq 0$. Hence, the minimum value of $f(x)$ in the region $x \in [0, t]$ will occur at either 0 or t. For the former, we have $f(0) = 0$. For the latter, we have

$$f(t) = (1 - e^{-nt})s - t = (1 - e^{-na(1+n)})a(1 + 100n) - a(1 + n).$$

Now $na \geq 10^6 \log a$, so $e^{-na(1+n)} \leq a^{-10^6}$. Since $a \geq a$ and $a \geq 2$, we get

$$f(t) \geq (1 - a^{-10^6})a(1 + 100n) - a(1 + n) = a(99n - 100n/a^{10^6} - a/a^{10^6}) \geq 0$$

as desired. ∎

Appendix E. Miscellaneous observations and formulas.

Proposition E.1. For any integer $a \geq 1$ and any $n > 0$, there is a multigraph with arboricity a and $\Delta = O(1/n)$ and $a = O(a)$, for which any $a(1 + n)$-FD has diameter $a(1/n)$.

Proof. Consider the graph G with $l = \lfloor 1/n \rfloor$ vertices arranged in a path and a edges between consecutive vertices. This has maximum degree $\Delta = 2a$ and arboricity a. In any forest decomposition of diameter D, each forest must consist of consecutive subpaths each of length at most D. Thus, each color uses at most $\lfloor \frac{l}{D+1} \rfloor \cdot D \leq D(1 + l/D + 1)$ edges. There are $(l - 1)a$ total edges, so we must have $a(1 + n)D(1 + l/(d + 1)) \geq (l - 1)a$. Since $l = a(1/n)$, this implies that $D \geq a(1/n)$. ∎

Proposition E.2. For a loopless multigraph, there holds $a_{star} \leq 2a^t$.

Proof. It sufices to show that a loopless pseudo-tree can be decomposed into two star forests. This pseudo-tree can be represented as a cycle $C = \{x_1, \dots, x_t\}$ plus trees $T_1, \dots, T_t$ rooted at each x_i.

We can two-color the edges of C from $\{0, 1\}$, such that there is at most one pair of consecutive edges on C with the same color; say without loss of generality it is color 0. For each edge e in a tree T_i which is at depth d ($d = 0$ means that e has an endpoint x_i), we assign e to color $(d + 1) \bmod 2$. In particular, the child edges from root x_i have color 1, the grandchild edges have color 0, etc. ∎

Theorem E.3. For a multigraph with degeneracy a, there holds $a^{list}_{star} \leq 2a \leq \min\{4a^t, 4a - 2\}$.

Proof. It is a standard result that $a \leq 2\min\{a - 1, a^t\}$, so it sufices to show $a^{list}_{star} \leq 2a$.

Fix some acyclic a-orientation of G, and color each edge sequentially, going backward in the orientation. For each edge e oriented from u to v, we choose a color in $Q(e)$ not already chosen by any out-neighbor of u or v. Each vertex has at most a out-neighbors, so at most a colors are already used by out-neighbors of v, and $a - 1$ colors are used by out-neighbors of u. Since e has a palette of size $2a$, we can always choose a color for e. The resulting coloring at the end is a star-list-coloring, where each edge is oriented toward the center of its star. ∎

Acknowledgments. 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 notation and terminology. Thanks to the anonymous conference and journal reviewers for helpful comments and suggestions.

REFERENCES

[1] I. Algor and N. Alon, The star arboricity of graphs, Discrete Math., 43 (1989), pp. 11--22.
[2] N. Alon, C. McDiarmid, and B. Reed, Star arboricity, Combinatorica, 12 (1992), pp. 375--380.
[3] Y. Aoki, The star-arboricity of the complete regular multipartite graphs, Discrete Math., 81 (1990), pp. 115--122, https://doi.org/10.1016/0012-365X(90)90142-5.
[4] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg, Fast distributed network decompositions and covers, J. Parallel Distrib. Comput., 39 (1996), pp. 105--114.
[5] B. Bahmani, A. Goel, and K. Munagala, Eficient primal-dual graph algorithms for MapReduce, in Proc. 11th International Workshop on Algorithms and Models for the Web-Graph (WAW), Lecture Notes in Comput. Sci. 8882, Springer, 2014, pp. 59--78.
[6] B. Bahmani, R. Kumar, and S. Vassilvitskii, Densest subgraph in streaming and MapReduce, Proc. VLDB Endow., 5 (2012), pp. 454--465.
[7] L. Barenboim and M. Elkin, Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition, Distrib. Comput., 22 (2010), pp. 363--379.
[8] L. Barenboim and M. Elkin, Deterministic distributed vertex coloring in polylogarithmic time, J. ACM, 58 (2011), 23.
[9] L. Barenboim and M. Elkin, Distributed graph coloring: Fundamentals and recent developments, Synthesis Lect. Distrib. Comput. Theory, 4 (2013), pp. 1--171.
[10] S. Behnezhad, S. Brandt, M. Derakhshan, M. Fischer, M. Hajiaghayi, R. M. Karp, and J. Uitto, Massively parallel computation of matching and MIS in sparse graphs, in Proc. ACM Symposium on Principles of Distributed Computing (PODC), 2019, pp. 481--490.
[11] E. Berglin and G. S. Brodal, A simple greedy algorithm for dynamic graph orientation, Algorithmica, 82 (2020), pp. 245--259.
[12] A. Bernshteyn, A fast distributed algorithm for a + 1-edge-coloring, J. Combin. Theory Ser. B, 152 (2022), pp. 319--352.
[13] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis, Space- and time-eficient algorithm for maintaining dense subgraphs on one-pass dynamic streams, in Proc. 47th ACM Symposium on Theory of Computing (STOC), 2015, pp. 173--182.
[14] G. S. Brodal and R. Fagerberg, Dynamic representations of sparse graphs, in Proc. 6th International Workshop on Algorithms and Data Structures (WADS), Lecture Notes in Comput. Sci. 1663, Springer, 1999, pp. 342--351.
[15] Y.-J. Chang, Q. He, W. Li, S. Pettie, and J. Uitto, The complexity of distributed edge coloring with small palettes, in Proc. 29th ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2018, pp. 2633--2652, https://doi.org/10.1137/1.9781611975031.168.
[16] M. Charikar, Greedy approximation algorithms for finding dense components in a graph, in Approximation Algorithms for Combinatorial Optimization (Saarbrücken, 2000), Lecture Notes in Comput. Sci. 1913, Springer, 2000, pp. 84--95.
[17] K.-M. Chung, S. Pettie, and H.-H. Su, Distributed algorithms for the Lovász local lemma and graph coloring, Distrib. Comput., 30 (2017), pp. 261--280.
[18] R. Cole and U. Vishkin, Deterministic coin tossing with applications to optimal parallel list ranking, Inform. Comput., 70 (1986), pp. 32--53.
[19] D. Dubhashi, D. A. Grable, and A. Panconesi, Near-optimal, distributed edge colouring via the nibble method, Theoret. Comput. Sci., 203 (1998), pp. 225--251.
[20] M. Elkin and O. Neiman, Distributed strong diameter network decomposition, Theoret. Comput. Sci. 922 (2022), pp. 150--157.
[21] M. Elkin, S. Pettie, and H.-H. Su, (2a - 1)-edge-coloring is much easier than maximal matching in the distributed setting, in Proc. 26th ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2015, pp. 355--370, https://doi.org/10.1137/1.9781611973730.26.
[22] H. Esfandiari, M. Hajiaghayi, and D. P. Woodruff, Applications of uniform sampling: Densest subgraph and beyond, in Proc. 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2016, pp. 397--399.

[23] M. Fischer, M. Ghaffari, and F. Kuhn, Deterministic distributed edge-coloring via hypergraph maximal matching, in Proc. 58th IEEE Symposium on Foundations of Computer Science (FOCS), 2017, pp. 180--191.
[24] H. N. Gabow and M. Stallmann, Eficient algorithms for graphic matroid intersection and parity, in Proc. 12th International Colloquium on Automata, Languages and Programming (ICALP), Lecture Notes in Comput. Sci. 194, Springer, 1985, pp. 210--220.
[25] H. N. Gabow and H. H. Westermann, Forests, frames, and games: Algorithms for matroid sums and applications, Algorithmica, 7 (1992), pp. 465--497.
[26] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan, A fast parametric maximum flow algorithm and applications, SIAM J. Comput., 18 (1989), pp. 30--55, https://doi.org/10.1137/0218003.
[27] M. Ghaffari, D. G. Harris, and F. Kuhn, On derandomizing local distributed algorithms, in Proc. 59th IEEE Symposium on Foundations of Computer Science (FOCS), 2018, pp. 662--673.
[28] M. Ghaffari, J. Hirvonen, F. Kuhn, and Y. Maus, Improved distributed a -coloring, in Proc. 37th ACM Symposium on Principles of Distributed Computing (PODC), 2018, pp. 427--436.
[29] M. Ghaffari, F. Kuhn, and Y. Maus, On the complexity of local distributed graph problems, in Proc. 49th ACM Symposium on Theory of Computing (STOC), 2017, pp. 784--797.
[30] M. Ghaffari, F. Kuhn, Y. Maus, and J. Uitto, Deterministic distributed edge-coloring with fewer colors, in Proc. 50th ACM Symposium on Theory of Computing (STOC), 2018, pp. 418--430.
[31] M. Ghaffari, S. Lattanzi, and S. Mitrović, Improved parallel algorithms for density-based network clustering, in Proc. 36th International Conference on Machine Learning (ICML), PMLR 97, 2019, pp. 2201--2210.
[32] M. Ghaffari and H.-H. Su, Distributed degree splitting, edge coloring, and orientations, in Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2017, pp. 2505--2523, https://doi.org/10.1137/1.9781611974782.166.
[33] A. V. Goldberg, Finding a Maximum Density Subgraph, Tech. report UCB/CSD-84-171, EECS Department, University of California, Berkeley, 1984.
[34] A. D. Gore, A. Karandikar, and S. Jagabathula, On high spatial reuse link scheduling in STDMA wireless ad hoc networks, in Proc. IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference, 2007, pp. 736--741.
[35] A. Gupta, A. Kumar, and C. Stein, Maintaining assignments online: Matching, scheduling, and flows, in Proc. 25th ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2014, pp. 468--479, https://doi.org/10.1137/1.9781611973402.35.
[36] S. L. Hakimi, On the degrees of the vertices of a directed graph, J. Franklin Inst., 279 (1965), pp. 290--308.
[37] D. G. Harris, Distributed local approximation algorithms for maximum matching in graphs and hypergraphs, SIAM J. Comput., 49 (2020), pp. 711--746, https://doi.org/10.1137/19M1279241.
[38] H. Imai, Network-flow algorithms for lower-truncated transversal polymatroids, J. Oper. Res. Soc. Japan, 26 (1983), pp. 186--211.
[39] S. Khuller and B. Saha, On finding dense subgraphs, in Proc. 36th International Colloquium on Automata, Languages and Programming (ICALP), Part I, Lecture Notes in Comput. Sci. 5555, Springer, 2009, pp. 597--608.
[40] T. Kopelowitz, R. Krauthgamer, E. Porat, and S. Solomon, Orienting fully dynamic graphs with worst-case time bounds, in Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP), Part II, Lecture Notes in Comput. Sci. 8573, Springer, 2014, pp. 532--543.
[41] Ł. Kowalik, Approximation scheme for lowest outdegree orientation and graph density measures, in Proc. 17th International Conference on Algorithms and Computation (ISAAC), Lecture Notes in Comput. Sci. 4288, Springer, 2006, pp. 557--566.
[42] F. Kuhn, Faster deterministic distributed coloring through recursive list coloring, in Proc. 31st ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2020, pp. 1244--1259, https://doi.org/10.1137/1.9781611975994.76.
[43] N. Linial, Locality in distributed graph algorithms, SIAM J. Comput., 21 (1992), pp. 193--201, https://doi.org/10.1137/0221015.
[44] N. Linial and M. E. Saks, Low diameter graph decompositions, Combinatorica, 13 (1993), pp. 441--454.
[45] A. McGregor, D. Tench, S. Vorotnikova, and H. T. Vu, Densest subgraph in dynamic graph streams, in Mathematical Foundations of Computer Science 2015, Part II, Lecture Notes in Comput. Sci. 9235, Springer, 2015, pp. 472--482.

[46] G. L. Miller, R. Peng, and S. C. Xu, Parallel graph decompositions using random shifts, in Proc. 25th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2013, pp. 196--203.
[47] C. S. A. Nash-Williams, Decomposition of finite graphs into forests, J. London Math. Soc., s1-39 (1964), pp. 12--12.
[48] A. Panconesi and A. Srinivasan, Randomized distributed edge coloring via an extension of the Chernoff--Hoeffding bounds, SIAM J. Comput., 26 (1997), pp. 350--368, https://doi.org/10.1137/S0097539793250767.
[49] J.-C. Picard and M. Queyranne, A network flow solution to some nonlinear 0-1 programming problems, with applications to graph theory, Networks, 12 (1982), pp. 141--159.
[50] S. Ramanathan and E. L. Lloyd, Scheduling algorithms for multihop radio networks, IEEE/ACM Trans. Netw., 1 (1993), pp. 166--177.
[51] J. Roskind and R. E. Tarjan, A note on finding minimum-cost edge-disjoint spanning trees, Math. Oper. Res., 10 (1985), pp. 701--708.
[52] V. Rozhoň and M. Ghaffari, Polylogarithmic-time deterministic network decomposition and distributed derandomization, in Proc. 52nd ACM Symposium on Theory of Computing (STOC), 2020, pp. 350--363.
[53] A. D. Sarma, A. Lall, D. Nanongkai, and A. Trehan, Dense subgraphs on dynamic networks, in DISC, Lecture Notes in Comput. Sci. 7611, Springer, 2012, pp. 151--165.
[54] S. Sawlani and J. Wang, Near-optimal fully dynamic densest subgraph, in Proc. 52nd ACM Symposium on Theory of Computing (STOC), 2020, pp. 181--193.
[55] P. D. Seymour, A note on list arboricity, J. Combin. Theory Ser. B, 72 (1998), pp. 150--151.
[56] J. Shi, L. Dhulipala, and J. Shun, Parallel clique counting and peeling algorithms, in Proc. SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21), SIAM, 2021, pp. 135--146, https://doi.org/10.1137/1.9781611976830.13.
[57] H.-H. Su and H. T. Vu, Distributed dense subgraph detection and low outdegree orientation, in Proc. 34th International Symposium on Distributed Computing (DISC), LIPIcs. Leibniz Int. Proc. Inform. 179, Schloss Dagstuhl. Leibniz-Zent. Inform., 2020, p. 15.
[58] H.-H. Su and H. T. Vu, Towards the locality of Vizing's theorem, in Proc. 51st ACM Symposium on Theory of Computing (STOC), 2019, pp. 355--364.
[59] V. G. Vizing, On an estimate of the chromatic class of a p-graph, Diskret. Analiz, 1964, no. 3, pp. 25--30 (in Russian).