58th Annual IEEE Symposium on Foundations of Computer Science

Deterministic Distributed Edge-Coloring via Hypergraph Maximal Matching

Manuela Fischer and Mohsen Ghaffari
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
manuela.fischer@inf.ethz.ch and ghaffari@inf.ethz.ch

Fabian Kuhn
Department of Computer Science
University of Freiburg
79110 Freiburg, Germany
kuhn@cs.uni-freiburg.de

***Abstract*—We present a deterministic distributed algorithm that computes a $(2\Delta-1)$-edge-coloring, or even list-edge-coloring, in any n-node graph with maximum degree Δ, in $O(\log^8 \Delta \cdot \log n)$ rounds. This answers one of the long-standing open questions of *distributed graph algorithms* from the late 1980s, which asked for a polylogarithmic-time algorithm. See, e.g., Open Problem 4 in the Distributed Graph Coloring book of Barenboim and Elkin. The previous best round complexities were $2^{O(\sqrt{\log n})}$ by Panconesi and Srinivasan [STOC'92] and $\tilde{O}(\sqrt{\Delta}) + O(\log^* n)$ by Fraigniaud, Heinrich, and Kosowski [FOCS'16]. A corollary of our deterministic list-edge-coloring also improves the randomized complexity of $(2\Delta-1)$-edge-coloring to poly$(\log\log n)$ rounds.**

The key technical ingredient is a deterministic distributed algorithm for *hypergraph maximal matching*, which we believe will be of interest beyond this result. In any hypergraph of rank r — where each hyperedge has at most r vertices — with n nodes and maximum degree Δ, this algorithm computes a maximal matching in $O(r^5 \log^{6+\log r} \Delta \cdot \log n)$ rounds.

This hypergraph matching algorithm and its extensions also lead to a number of other results. In particular, we obtain a polylogarithmic-time deterministic distributed maximal independent set (MIS) algorithm for graphs with bounded neighborhood independence, hence answering Open Problem 5 of Barenboim and Elkin's book, a $\big((\log\Delta/\varepsilon)^{O(\log 1/\varepsilon)}\big)$-round deterministic algorithm for $(1+\varepsilon)$-approximation of maximum matching, and a quasi-polylogarithmic-time deterministic distributed algorithm for orienting λ-arboricity graphs with out-degree at most $\lceil(1+\varepsilon)\lambda\rceil$, for any constant $\varepsilon > 0$, hence partially answering Open Problem 10 of Barenboim and Elkin's book.

***Keywords*-distributed graph algorithms; local algorithms; deterministic distributed algorithms; edge-coloring; hypergraph; maximal matching; rounding linear programs**

I. Introduction

Distributed graph algorithms have been studied extensively over the past 30 years, since the seminal work of Linial [Lin87]. Despite this, determining whether there are efficient deterministic distributed algorithms for the most classic problems of the area remains a long-standing open question.

Distributed graph algorithms are typically studied in a standard synchronous message passing model known as the LOCAL model [Lin87], [Pel00]: the network is abstracted as an undirected graph $G = (V, E)$, $n = |V|$, with maximum degree Δ, and where each node has a $\Theta(\log n)$-bit unique identifier. Initially, nodes have knowledge of their neighbors in G as well as $\log n$ and $\log\Delta$ up to constants.[1] At the end, each node should know its own part of the solution, e.g., the colors of its edges in edge-coloring. Communication happens in synchronous rounds, where in each round each node sends a message to each of its neighbors.[2] The main complexity measure is the number of rounds needed for solving a given graph problem. The four classic local distributed graph problems are *Maximal Independent Set* (MIS), $(\Delta+1)$-vertex-coloring, $(2\Delta-1)$-edge-coloring, and maximal matching [PR01], [BE13]. All these problems have trivial greedy sequential algorithms, as well as simple $O(\log n)$-round randomized distributed algorithms [Lub86], [ABI86], and even some faster ones [BEPS12], [EPS15], [Gha16], [HSS16]. But the deterministic distributed complexity remains widely open, despite extensive interest (see, e.g., the first five open problems of [BE13]). Particularly, with regards to MIS — which is the hardest of the four problems, as the other three can be reduced to MIS locally [Lin87] — Linial [Lin92] asked

> *"can it [MIS] always be found [deterministically] in polylogarithmic time?"*.

This remains the most well-known open question of the area. The best known round complexity is $2^{O(\sqrt{\log n})}$, due to Panconesi and Srinivasan [PS92]. Panconesi and Rizzi pointed out in the year 2001 [PR01] that "*while maximal matchings can be computed in polylogarithmic time, in n, in the distributed model [HKP98], it is a decade old open problem whether the same running time is achievable for the remaining 3 structures.*" The status remains the same as of today, after almost two more decades. In particular, for edge-coloring which is our main target, Barenboim and Elkin stated the following problem in their recent Distributed

[1] If the latter is not the case, it is enough to try exponentially increasing estimates.

[2] In the LOCAL model, messages might be of arbitrary size. A variant of the model where the messages have to be of bounded size is known as the CONGEST model [Pel00]. Our edge-coloring algorithms and our MIS and vertex-coloring algorithms for graphs of bounded neighborhood independence in fact work with small $O(\log n)$-bit-size messages. Though, for the sake of the readability, we avoid explicitly discussing the details of this aspect.

0272-5428/17 $31.00 © 2017 IEEE
DOI 10.1109/FOCS.2017.25

Graph Coloring book [BE13]:

> OPEN PROBLEM 11.4 [BE13]: Devise or rule out a deterministic $(2\Delta - 1)$-edge-coloring algorithm that runs in polylogarithmic time.

A. Our Contributions

Deterministic Edge-Coloring Algorithm: One of our main end results is a positive resolution of the above question.

Theorem I.1. *There is a deterministic distributed algorithm that computes a $(2\Delta-1)$-edge-coloring in $O(\log^8 \Delta \cdot \log n)$ rounds, in any n-node graph with maximum degree Δ. Moreover, the same algorithm solves list-edge-coloring, where each edge $e \in E$ must get a color from an arbitrary given list L_e of colors with $|L_e| = d_e + 1$, where d_e denotes the number of edges incident to e.*

For list-edge-coloring, the previously best known round complexity was $2^{O(\sqrt{\log n})}$, by a classic network decomposition of Panconesi and Srinivasan [PS92], which itself improved on an $2^{O(\sqrt{\log n \cdot \log\log n})}$-round algorithm of Awerbuch et al. [ALGP89]. For low-degree graphs, the best known is an $(\tilde{O}(\sqrt{\Delta}) + O(\log^* n))$-round algorithm of Fraigniaud, Heinrich, and Kosowski [FHK16], which is more general and applies also to $(\Delta+1)$-list-vertex-coloring. There are some known $\operatorname{poly}\log n$-round deterministic edge-coloring algorithms, but all have two major shortcomings: (A) they require more colors, and (B) they are quite restricted and do not work for list-coloring. These algorithms are as follows: (I) a $((2+o(1))\Delta)$-edge-coloring by Ghaffari and Su [GS17]; (II) a $\Delta \cdot 2^{O\left(\frac{\log \Delta}{\log\log \Delta}\right)}$-edge-coloring by Barenboim and Elkin [BE11]—see also [GS17, Appendix B] for a short proof; and (III) an $O(\Delta \log n)$-edge-coloring by Czygrinow et al. [CHK01]. See also [BE11, Chapter 8].

We also note that a recent work of Barenboim, Elkin, and Maimon [BEM17] presents an efficient deterministic algorithm for computing a $(\Delta + o(\Delta))$-edge-coloring in graphs with arboricity $a \leq \Delta^{1-\delta}$, for some constant $\delta > 0$. In Corollary IV.1, we sketch how a simple combination of the list-edge-coloring algorithm of Theorem I.1 with H-partitionings [BE13, Chapter 5.1] significantly extends their result.

Improved Randomized Edge-Coloring Algorithm: The deterministic list-edge-coloring of Theorem I.1, in combination with some randomized edge-coloring algorithms of [EPS15], [BEPS12], also improves the complexity of randomized algorithms for $(2\Delta - 1)$-edge-coloring, making it the first among the four classic problems whose randomized complexity falls down to $\operatorname{poly}(\log\log n)$.

Corollary I.2. *There is a randomized distributed algorithm that computes a $(2\Delta - 1)$-edge-coloring in $O(\log^9 \log n)$ rounds, with high probability, in any n-node graph with maximum degree Δ.*

The previous (worst-case) complexity for randomized $(2\Delta - 1)$-edge-coloring was $2^{O(\sqrt{\log\log n})}$ rounds[3], due to Elkin, Pettie, and Su [EPS15]. By improving this, Corollary I.2 widens the provable gap between the complexity of $(2\Delta-1)$-edge-coloring, which is now in $\operatorname{poly}(\log\log n)$ rounds, and the complexity of maximal matching, which needs $\Omega(\sqrt{\log n/\log\log n})$ rounds [KMW16].

Unified Formulation as Hypergraph Maximal Matching: Our first step towards proving Theorem I.1 is a simple unification of all the aforementioned four classic problems: MIS, $(\Delta + 1)$-vertex-coloring, $(2\Delta - 1)$-edge-coloring, and maximal matching. We can cast each of these problems as a maximal matching problem on hypergraphs[4] of some rank r, which depends on the problem, and increases as we move from maximal matching to maximal independent set. In other words, the hypergraph maximal matching problem can be used to obtain a smooth interpolation between maximal matching in graphs and maximal independent set in graphs. Recall that the rank of a hypergraph is the maximum number of vertices in any of its hyperedges. Moreover, a matching in a hypergraph is a set of hyperedges, no two of which share an endpoint.

We present a reduction from $(2\Delta - 1)$-edge-coloring to maximal matching in rank-3 hypergraphs, as we sketch next in Lemma I.3. A similar reduction can be used for list-edge-coloring, as formalized in Lemma III.1. We note that these reductions are inspired by the well-known reduction of Luby from $(\Delta + 1)$-vertex-coloring to MIS [Lub86], [Lin87].

Lemma I.3. *Given a deterministic distributed algorithm $\mathcal{A}$ that computes a maximal matching in N-vertex hypergraphs of rank 3 and maximum degree d in $T(N, d)$ rounds, there is a deterministic distributed algorithm $\mathcal{B}$ that computes a $(2\Delta-1)$-edge-coloring of any n-node graph $G = (V, E)$ with maximum degree Δ in at most $T(3n\Delta, 2\Delta - 1)$ rounds.*

Proof Sketch: To edge-color $G = (V, E)$, we generate a hypergraph H: Take $2\Delta-1$ copies of G. For each edge $e \in E$, let e_1 to $e_{2\Delta-1}$ be its copies. For each $e \in E$, add one extra vertex w_e to H and then, change all edge copies e_1 to $e_{2\Delta-1}$ to 3-hyperedges by adding w_e to them. Algorithm $\mathcal{B}$ runs the maximal matching algorithm $\mathcal{A}$ on H. Then, for each $e \in E$, if the copy e_i of e is in the computed maximal matching, $\mathcal{B}$ colors

[3]It is worth noting that the randomized algorithm of [EPS15], as well as its predecessors [PS97], [DGP98], can also obtain better colorings, even as good as $((1+\varepsilon)\Delta)$-edge-coloring. Though this becomes slow in low-degree graphs.

[4]In the LOCAL model, when communicating on a hypergraph, per round each node v can send a message on each of its hyperedges, which then gets delivered to all the other endpoints of that hyperedge. The variant of the model with bounded-size messages can be specialized in a few different ways, see e.g. [KNPR14]. For our purposes, hypergraphs are mainly used for formulating the requirements of the problem, and the real communication happens on the base graph.

e with color i. One can see that each G-edge e must have exactly one copy e_i in the maximal matching, and thus we get a $(2\Delta - 1)$-edge-coloring. ■

Besides edge-coloring that gets reduced to hypergraph maximal matching for rank $r = 3$, and graph maximal matching which trivially is the special case of $r = 2$, we can also formulate MIS and $(\Delta+1)$-vertex-coloring as maximal matching in hypergraphs. For instance, to translate MIS on a graph G to maximal matching on a hypergraph H, view each G-edge as one H-vertex and each G-node v as one H-edge on the H-vertices corresponding to the G-edges incident to v. However, unfortunately, in this naive formulation, the rank becomes Δ. As such, we do not obtain any improvement over the known algorithms for these problems, in the general case. It remains an intriguing open question whether any alternative formulation, perhaps in combination with other ideas, can help. However, as we shall discuss soon, using some more involved ideas, we obtain improvements for some special cases, which lead to answers for a few other open problems.

Our Hypergraph Maximal Matching Algorithm: Our main technical contribution is an efficient deterministic algorithm for maximal matching in low-rank hypergraphs. In combination with the reduction of Lemma I.3, this leads to our edge-coloring algorithm stated in Theorem I.1.

Theorem I.4. *There is a deterministic distributed algorithm that computes a maximal matching in $O(r^5 \log^{6+\log r} \Delta \cdot \log n)$ rounds[5], in any n-node hypergraph with maximum degree Δ and rank r, i.e., where each hyperedge contains at most r vertices.*

This result has a number of other implications, as we overview in Section I-B. Besides those, it also supplies an alternative $\operatorname{poly}\log n$-round deterministic algorithm for maximal matching in graphs, where $r = 2$. We remark that $\operatorname{poly}\log n$ deterministic algorithms for graph maximal matching have been known for about two decades, due to the breakthroughs of Hańćkowiak, Karonski, and Panconesi [HKP98], [HKP99]. Moreover, a faster algorithm was recently presented in [Fis17]. However, the methods of [HKP98], [HKP99], [Fis17], or their natural extensions, inherently rely on rank $r = 2$ in a seemingly crucial manner, and they do not extend to hypergraphs of rank 3 or higher. The method we develop for Theorem I.4 is quite different and significantly more flexible. We overview this method in Section I-C, and contrast it with the previously known techniques. The difference and the generality of our method becomes more discernible when considering another closely related open problem which did not seem solvable using the methods of [HKP98], [HKP99] and which can now be solved using a natural, though non-trivial, extension of Theorem I.4.

[5]Throughout this paper, all logarithms are to base 2. Moreover, we stress that one does not need a ceiling sign for $\log r$ in this complexity bound.

Extension to MIS in Graphs with Bounded Neighborhood Independence: Consider the problem of computing an MIS in graphs with *neighborhood independence* bounded by an integer r, i.e., where the number of mutually non-adjacent neighbors of each node is at most r. Notice that maximal matching in graphs is the same as MIS in the corresponding line graph, which is a graph of neighborhood independence $r = 2$. It is not clear how to extend the methods of [HKP98], [HKP99] to MIS in such graphs[6], even for $r = 2$. As an open question alluding to this point, and as "*a good stepping stone towards the MIS problem in general graphs*", Barenboim and Elkin asked in their book [BE13]:

> OPEN PROBLEM 11.5 [BE13] Devise or rule out a deterministic polylogarithmic algorithm for the MIS problem in graphs with neighborhood independence bounded by 2.

Our method for Theorem I.4 generalizes to MIS in graphs with bounded neighborhood independence, as we state formally next, hence positively answering this open question.

Theorem I.5. *There is a deterministic distributed algorithm that computes a maximal independent set in $O(r^5 \log^{6+\log r} \Delta \cdot \log n)$ rounds, in any n-node graph with maximum degree Δ and neighborhood independence bounded by r.*

Moreover, since Luby's reduction of $(\Delta + 1)$-vertex-coloring to MIS [Lub86], [Lin87] increases the neighborhood independence by at most 1, we also get efficient algorithms for $(\Delta + 1)$-vertex-coloring in graphs with bounded neighborhood independence.

Corollary I.6. *There is a deterministic distributed algorithm that computes a $(\Delta + 1)$-vertex-coloring in $O(r^5 \log^{6+\log(r+1)} \Delta \cdot \log n)$ rounds, in any n-node graph with maximum degree Δ and neighborhood independence bounded by r. Moreover, the same algorithm solves list-vertex-coloring, where each node $v \in V$ must get a color from an arbitrary given list L_v of colors with $|L_v| \geq \deg(v) + 1$.*

The proofs of these results are in the full version [FGK17].

B. Other Implications

Our hypergraph maximal matching algorithm enables us to obtain answers and improvements for some other problems. A family of improvements comes for graph problems in which the main technical challenge is to find a maximal set of disjoint augmenting paths of short length ℓ. These problems can be phrased as maximal matching in hypergraphs with rank $r = \Theta(\ell)$, essentially by viewing each augmenting path as one hyperedge on its elements (depending on the required disjointness). We next mention

[6]For a more strict definition of bounded neighborhood, MIS turns out to be much easier. See [KMNW05], [SW10].

the results that we obtain based on this connection. The proofs appear in the full version [FGK17].

Maximum Matching Approximation: By integrating our hypergraph maximal matching into the framework of Hopcroft and Karp [HK73], we can compute a $(1+\varepsilon)$-approximation of maximum matching in graphs in $\left((\log\Delta/\varepsilon)^{O(\log 1/\varepsilon)}\right)$ rounds. For that, we mainly need to find maximal sets of vertex-disjoint augmenting paths of length at most $\ell = O(1/\varepsilon)$. This is faster than the previously best known deterministic algorithm for $(1+\varepsilon)$-approximation in bipartite graphs, which required $\log^{O(1/\varepsilon)} n$ rounds [CH03]. We remark that an $O(\log n/\varepsilon^3)$-round randomized $(1+\varepsilon)$-approximation algorithm was presented by Lotker et al. [LPSP08], mainly by computing this maximal set of vertex-disjoint augmenting paths using Luby's randomized MIS algorithm [Lub86].

Low-Out-Degree Orientation and (Pseudo-)Forest Decomposition: By integrating our hypergraph maximal matching into the low-out-degree orientation framework of Ghaffari and Su [GS17], we can compute orientations with out-degree at most $\lceil(1+\varepsilon)\lambda\rceil$, for any $0<\varepsilon<1$, in graphs with arboricity λ. For that, we mainly need to find maximal sets of disjoint augmenting paths of length $\ell = O(\log n/\varepsilon)$. This low-out-degree orientation directly implies a decomposition into $\lceil(1+\varepsilon)\lambda\rceil$ edge-disjoint pseudo-forests. For constant ε and even $\varepsilon = \Omega(1/\operatorname{poly}\log n)$, the round complexity of the resulting algorithm is quasi-polylogarithmic—that is, $2^{O(\log^2\log n)}$. Although this is not a polylogarithmic complexity, it gets close and it is almost exponentially faster than the previously best known $2^{O(\sqrt{\log n})}$ deterministic algorithm [GS17], [PS92]. This improvement can be viewed as partial solution for Open Problem 11.10 of Barenboim and Elkin [BE13], which asks for an efficient deterministic distributed algorithm for decomposing the graph into less than 2λ forests.

C. Our Method for Hypergraph Maximal Matching, in a Nutshell

The main ingredient in our results is our hypergraph maximal matching algorithm. Here, we present a brief overview of this algorithm. Before the overview, we note that the key technical novelty in our hypergraph maximal matching algorithm is developing an efficient deterministic distributed *rounding* method, which transforms *fractional* hypergraph matchings to *integral* hypergraph matchings. This becomes more instructive when viewed in the context of the recent results of Ghaffari, Kuhn, and Maus [GKM17], which show that deterministically rounding fractional solutions of certain linear programs into integral solutions while approximately preserving some linear constraints is the *"the only obstacle"* for efficient deterministic distributed algorithms. In other words, if we find an efficient deterministic method for approximately rounding certain linear programs, we would get efficient algorithms for essentially all the classic local graph problems, including MIS. See [GKM17] for the precise statement.

Our rounding for hypergraph matchings can be seen as a drastic generalization of the rounding methods Fischer presented recently for matching in normal graphs [Fis17]. The methods of [Fis17] do not extend to hypergraphs (even for rank $r=3$), for reasons that we will discuss soon. The new rounding method we present is more general and significantly more flexible. As such, we are hopeful that this deterministic rounding method will prove useful for a wider range of problems, and may potentially serve as a stepping stone towards a poly $\log n$-time deterministic MIS algorithm.

We next present a high-level overview of our hypergraph maximal matching algorithm.

Matching Approximation: The core of our maximal matching algorithm is an algorithm that computes a matching whose size is within an factor $O(r^3)$ of the maximum matching. Once given such an approximation algorithm, we can easily find a maximal matching in $O(r^3\log n)$ iterations, by repeated applications of this approximation algorithm, each time adding the found matching to the output matching, and then removing the found matching and its incident edges from the hypergraph.

Fractional Matchings: In approximating the maximum matching, the main challenge is finding an *integral* matching with such an approximation guarantee. Finding a *fractional* matching — where each edge e has a value $x_e \in [0,1]$ such that for each vertex v we have $\sum_{e\in E(v)} x_e \le 1$ — with such an approximation is trivial, and can be done in $O(\log\Delta)$ rounds: initially, set $x_e = 1/\Delta$ for all edges e. Then, for $\log\Delta$ iterations, each time double the values x_e of all the edges e for which all vertices $v\in e$ have $\sum_{e\in E(v)} x_e \le 1/2$. One can see that this produces a $(2r)$-approximation. See [KY09], [KMW06], [GKM17], which discuss other distributed aspects of linear programs. The challenge thus is in *rounding* fractional matchings to integral matchings, without losing much in the size $\sum_{e\in E} x_e$.

Known Rounding Methods for Graphs, and Their Shortcomings: In graphs with rank $r=2$, this rounding can be done essentially with no loss. Indeed, this is the core part of the recent maximal matching algorithm of Fischer [Fis17], which finds a maximal matching in $O(\log^2\Delta\log n)$ rounds. The method of [Fis17] rounds any fractional matching in graphs in $O(\log\Delta)$ iterations, in each iteration moving a factor 2 closer to integrality while decreasing the matching size only negligibly, by a factor $(1-\frac{\varepsilon}{\Theta(\log\Delta)})$. Hence, even after all the rounding iterations, the overall loss is a negligible factor ε, for a desirably small $\varepsilon>0$. Although the algorithms of [HKP98], [HKP99] are not explicitly phrased in this rounding framework, one can see that the principle behind them is the same. The reader familiar with [HKP98], [HKP99] might recall that the key component is, roughly speaking, to decompose edges of any regular graph into two

groups, say red and blue, so that *almost all* nodes see a fair split of their edges into the two colors. This is a special case of rounding for regular graphs.

This whole methodology of rounding without more than a factor $o(1)$ loss in the size seems to be quite limited, and it certainly gets stuck at rank $r = 2$. For the interested reader, we briefly sketch the obstacle: all of those matching rounding methods [Fis17], [HKP98], [HKP99] decompose the edges of the graph into bipartite low-diameter degree-2 graphs (i.e., short even-length cycles) — aside from a smaller portion of some not-so-nice parts, which are handled separately — and then 2-color edges of each short cycle so that each node has half of its edges in each color. Then, in rounding, one color is raised by a factor 2 while the other is dropped to zero. Unfortunately, this type of locally-balanced splittings of edges does not seem within reach for hypergraphs, as of now. Indeed, if we could solve that, we would get far more consequential results: Ghaffari et al. [GKM17] recently proved that this splitting problem for hypergraphs is *'complete'*, meaning that if one can do such a splitting in polylogarithmic time for all hypergraphs, we get polylogarithmic deterministic algorithms for all the classic local problems, including MIS.

Challenges in Rounding for Hypergraphs: When trying to deterministically round fractional matchings in hypergraphs, we face essentially two challenges: (1) It is not clear how to efficiently perform any slight rounding— e.g., rounding all fractional values so that the minimum moves from at least $1/d$ to at least $2/d$ without violating the constraints — without a considerable loss in the matching size. (2) An even more crucial issue comes from the need to do many levels of rounding. Even once we have an efficient solution for a single iteration of rounding, which moves say a constant factor closer to integrality, a factor-$\Theta(r)$ reduction of the matching size seems inevitable. However, if we do this repeatedly, and our matching size drops by a factor $\Theta(r)$ in each rounding iteration, the matching size would become too small. Notice that we need about $O(\log \Delta)$ levels of factor-2 roundings. If we decrease by a factor $\Omega(r)$ per iteration, we would be left with a matching of size a factor $1/r^{\Theta(\log \Delta)} = 1/\operatorname{poly}(\Delta)$ of the maximum matching, which is essentially useless. We next mention a bird's eye view of our rounding method.

Our Rounding Method for Matchings in Hypergraphs: We devise a rounding procedure for hypergraph matchings which rounds the fractional matching by a factor L— i.e., raising fractional values by factor L—while reducing the matching size only by a factor $\Theta(r)$. On a high level, this rounding is by recursion on L. The base level of the recursion is an algorithm that rounds the fractional matching by a constant factor, for $L = O(1)$, with only a factor $\Theta(r)$ decrease of the matching size. This part is somewhat simpler and is performed efficiently using defective coloring results of [Kuh09]. This is a solution for the first challenge above. To overcome the second challenge, our method interleaves some iterations of *rounding* with *refilling* the fractional matching. In particular, suppose that we would like to do a factor-L rounding of a given fractional matching $\vec{x}$, thus producing an output fractional matching $\vec{y}$ with fractional values raised by a factor L compared to $\vec{x}$. We do this in $\Theta(r)$ iterations, using a number of factor-$\sqrt{2L}$ rounding procedures. Concretely, per iteration, we first '*remove*' the current output fractional matching $\vec{y}$ from the input fractional matching $\vec{x}$, in a sense to be made precise, and then we apply two successive factor-$(\sqrt{2L})$ rounding operations on the left-over fractional matching. This creates a fractional matching which is rounded by a factor of $2L$, but may be a factor $1/\Theta(r^2)$ smaller than $\vec{x}$. We add (a half of) this to the current fractional matching $\vec{y}$, in a sense to be made precise. The removal and also the addition are done carefully, so as to ensure that the size of the output fractional matching grows by about a factor $(1/\Theta(r^2))$ of the size of $\vec{x}$ while the fractionality is by a factor L better than the one of $\vec{x}$. After $\Theta(r)$ such iterations, we get that the output fractional matching is a $\Theta(r)$-approximation of the input.

Extension to MIS in Graphs with Bounded Neighborhood Independence: When moving from matchings in hypergraphs to independent sets in graphs of neighborhood independence at most r, it is not directly clear how to define a fractional solution of an MIS in such graphs. Note that the integrality gap of the natural LP relaxation of maximum independent set might be linear in Δ. However, any MIS is within a factor r of a maximum independent set, and this can in fact be generalized to maximal fractional solutions of the following kind. We start by setting the fractional values of all nodes to 0 and then, we iteratively increment the value of some nodes. As long as right after incrementing the value of a node v the total value in the 1-neighborhood of v does not exceed 1, the total value of the resulting fractional solution is guaranteed to be within a factor r of a maximum independent set. We call such a fractional solution a *greedy packing* and show that our rounding scheme for hypergraph matching can be adapted to greedy packings of graphs of bounded neighborhood independence.

Integral greedy packings are exactly independent sets. Thus, integral greedy packings of the line graph of a hypergraph H correspond to matchings of H. However, we note that a fractional greedy packing of the line graph of H is not the same as a fractional matching of H. We believe that this stresses the robustness of our approach. For example, when running the MIS algorithm for graphs of bounded neighborhood independence on the line graph of a bounded rank hypergraph H, we get a slightly different but equally efficient algorithm for computing a maximal matching of H.

II. Maximal Matching and Edge-Coloring in Hypergraphs

In this section, we present our hypergraph maximal matching algorithm, thus proving Theorem I.4. Then, at the end of Section III, we use this hypergraph maximal matching algorithm to prove our edge-coloring results, including Theorem I.1 and Corollary I.2.

For our hypergraph maximal matching algorithm, the key part is a *maximum matching approximation* procedure that finds a matching whose size is at least a factor $1/(32r^3)$ of the maximum matching.

Lemma II.1. *There is a deterministic distributed algorithm that computes a $(32r^3)$-approximate maximum matching in $O\Big(r^2 \log^{6+\log r} \Delta\Big)$ rounds, given an $O(r^2\Delta^2)$-edge-coloring.*

Once we have this approximation algorithm, we can find a maximal matching by iteratively applying this matching approximation procedure to the remainder hypergraph, for $O(r^3 \log n)$ iterations, each time removing the found matching and all its incident hyperedges. This is formalized in the proof of Theorem I.4 in Section II-E.

Over the next two subsections, we discuss the matching approximation procedure of Lemma II.1. We note that finding a *fractional* matching with size close to the maximum matching is straightforward, as we soon overview in Section II-A. The challenge is in finding an integral matching with the same guarantee. In other words, the core technical component of our method is an algorithm for *rounding* fractional hypergraph matchings to integral matchings, without losing much in the size. In particular, we present our deterministic rounding technique for hypergraph matchings in Section II-B.

A. Fractional Matching Approximation

In the following, we present a simple $O(\log \Delta)$-round algorithm that computes a $(2r)$-approximate fractional matching.

Some Notions and Terminology for Fractional Matchings: Given a hypergraph $H = (V, E)$, a fractional matching of H is an assignment of values $\vec{x} \in [0,1]^{|E|}$ to edges such that for each vertex $v \in V$, we have $\sum_{e \in E(v)} x_e \leq 1$. Here, $E(v) := \{e \in E \colon v \in e\}$ is the set of edges incident to v. We say a vertex v is *half-tight* in the given fractional matching $\vec{x}$ if $\sum_{e \in E(v)} x_e \geq \frac{1}{2}$. Moreover, we say $\vec{x}$ is a $(1/d)$-fractional matching if each edge $e \in E$ has $x_e \geq 1/d$ or $x_e = 0$.

Greedy Fractional Matching Algorithm: Initially, we set $x_e = \frac{1}{\Delta}$ for all edges e. This obviously is a valid fractional matching. Then, for $\log \Delta$ iterations, in each iteration, we freeze all the edges that have at least one half-tight vertex and then raise the value of all unfrozen edges by a factor 2.

This way, we always keep a valid fractional matching, since only the values of edges incident to non-half-tight vertices are increased. Moreover, within $O(\log \Delta)$ iterations all edges will be frozen. We next show that this property already implies an approximation ratio $2r$.

Lemma II.2. *The greedy algorithm described above computes a $(2r)$-approximate fractional matching. Moreover, any (fractional) matching $\vec{x}$ with the property that each edge has at least one half-tight endpoint is a $(2r)$-approximation.*

Proof: We show that $\vec{x}$ must have size at least a factor $1/(2r)$ of a maximum matching M^* employing an argument based on counting in two ways. To that end, we give 1 dollar to each edge $e \in M^*$ and ask it to redistribute this money among edges in such a way that no edge e' receives more than $2rx_{e'}$ dollars. This can be achieved as follows. Each edge $e \in M^*$ asks a half-tight vertex, say $v \in e$, to distribute e's dollar on e's behalf. Vertex v does so by splitting this money among its incident edges $e' \in E(v)$ proportionally to the edge values $x_{e'}$. In this way, every edge $e' \in E(v)$ receives no more than $2x_{e'}$ dollars from v. This is because v is half-tight and because it cannot have more than one incident edge in M^*, hence does not receive more than 1 dollar. Since an edge can receive money only from its vertices, every edge e' receives at most $2rx_{e'}$ dollars in total. ∎

B. Rounding Fractional Matchings in Hypergraphs

Our method for rounding fractional matchings is recursive, and parametrized mainly by a parameter L which captures the extent of the performed rounding.

Definition II.3 (factor-L rounding)**.** *Given a $(1/d)$-fractional matching $\vec{x} \in [0,1]^{|E|}$ — i.e. where for each $e \in E$, we have $x_e \geq 1/d$ or $x_e = 0$ — a factor-L rounding of $\vec{x}$ is a (L/d)-fractional matching $\vec{y} \in [0,1]^{|E|}$ where $x_e = 0$ implies $y_e = 0$.*

We will provide a method that performs an factor-L rounding of $\vec{x}$, and will refer to it as $\mathsf{round}(\vec{x}, L)$. Given as input a fractional matching $\vec{x} \in [0,1]^{|E|}$, the method $\mathsf{round}(\vec{x}, L)$ outputs a L/d-fractional matching $\vec{y} \in [0,1]^{|E|}$ of size at least a factor $1/(4r)$ of the input fractional matching, that is, $\sum_{e \in E} y_e \geq \frac{1}{4r} \sum_{e \in E} x_e$.

Remark II.4. *The method requires some condition on the values of L and d. Since L/d refers to the fractionality, the statement is meaningful only when $L \leq d$. Due to some small technicalities, we will perform the recursive parts of rounding only for values of L that satisfy a slightly stronger condition of $L \log^2 L \leq d$. For the remaining cases, we resort to our basic rounding.*

We explain our rounding method in two main parts. The first part, explained in Section II-C, is a procedure that we use as the base case, to round the matching by a constant

factor $L = O(1)$ in $O(r^2 + \log \Delta)$ rounds. The second part, discussed in Section II-D, is the recursive step which explains how our factor-L rounding works by making a few calls to factor-$\sqrt{2L}$ rounding procedures, and a few smaller steps. Finally, in Section II-E, we combine these rounding procedures with the previously seen algorithm of Section II-A for fractional matchings to obtain our matching approximation procedure of Lemma II.1.

C. Basic Rounding

In this subsection, we explain our base case rounding procedure for small rounding parameters, i.e., $L = O(1)$. Throughout, we will assume that the base hypergraph already has an $O(r^2\Delta^2)$-edge-coloring, which can be computed easily using Linial's algorithm [Lin87], in $O(\log^* n)$ rounds.

Lemma II.5 (Basic Rounding)**.** *There is an $O(L^2r^2 + \log \Delta)$-round deterministic distributed algorithm that turns a $(1/d)$-fractional matching $\vec{x}$ into an (L/d)-fractional matching $\vec{y}$ with $\sum_{e\in E} y_e \geq \frac{1}{2r}\sum_{e\in E} x_e$, for any $L \leq d$.*

Algorithm Outline and Intuitive Discussions: Let E_x be the set of all edges e for which $x_e > 0$, and let $H_x = (V, E_x)$ be the subgraph of H with this edge set. Notice that H_x has degree at most d, because $\vec{x}$ is a $(1/d)$-fractional matching. Our goal is to compute a fractional matching $\vec{y}$, supported on the edge set E_x, such that for each edge $e \in E_x$, at least one of its endpoints $v \in e$ is half-tight in $\vec{y}$, meaning that $\sum_{e'\in E_x(v)} y_{e'} \geq 1/2$. One can easily see that such a fractional matching is a $(2r)$-approximation of $\vec{x}$, i.e., $\sum_{e\in E} y_e \geq \frac{1}{2r}\sum_{e\in E} x_e$. Thus, the goal is to find a fractional matching $\vec{y}$ such that for each edge $e \in E_x$, at least one of its endpoints is *half-tight* in $\vec{y}$. Furthermore, we want $\vec{y}$ to be (L/d)-fractional, meaning that all the non-zero y_e-values must be greater than or equal to L/d.

If we had no concern for the time complexity, we could go through the color classes of edges one by one, each time setting $y_e = 1$ for all edges of that color, and then removing edges of E_x that have half-tight vertices. This would ensure that, at the end, all edges in E_x have at least one half-tight endpoint. However, this would require time proportional to the number of colors. Even if we were given an ideal edge-coloring for free, that would be $\Omega(d)$ rounds, which is too slow for us.

To speed up the process, we use a relaxed notion of edge-coloring, namely *defective edge-coloring*, which allows us to have much less colors, while each color class has a bounded number of edges incident to each vertex, say k. Now, we cannot raise the y_e-values of all the edges of the same color at the same time to $y_e = 1$, because that would be too fast and could violate the condition $\sum_{e'\in E_x(v)} y_{e'} \leq 1$. However, we can raise each of these edge values to say $y_e = \frac{1}{2k}$ and still be sure that the summation $\sum_{e'\in E_x(v)} y_{e'}$ for each node does not increase faster than an additive $1/2$. That is because there are only k edges incident to each node, per color class. If we freeze and set to 0 the values of all edges that now have one half-tight vertex, these fractional value raises would never violate the condition $\sum_{e'\in E_x(v)} y_{e'} \leq 1$, thus always lead to a valid fractional matching.

The Basic Rounding Algorithm: To materialize the above intuitive approach, we first compute a *defective edge-coloring* with $O\big(r^2L^2\big)$ colors and defect $k = \lfloor d/(2L)\rfloor$. Then, we go through the colors, one by one, applying the above fractional-value increases. This ensures that all the non-zero fractional values y_e are at least $\frac{1}{2k} \geq L/d$. At the very end, we perform $O(\log d/L)$ doubling steps to ensure that each edge has at least one half-tight endpoint. We next explain the steps of this algorithm, and then provide the related analysis.

Part I, Defective Edge-Coloring Algorithm: We compute a defective edge-coloring of H_x with $O(L^2r^2)$ colors and defect — that is, maximum degree induced by edges of the same color — at most $d/(2L)$, as follows. Let $F = (V_F, E_F)$ be the line graph of H_x, that is, the graph which has a vertex $v_e \in V_F$ for every edge $e \in E_x$ and an edge $\{v_e, v_{e'}\} \in E_F$ if e and e' are incident, thus $e \cap e' \neq \emptyset$. Note that F has maximum degree at most $r \cdot d$, since H_x's maximum degree is bounded by d. With the defective coloring algorithm of Kuhn [Kuh09], we can compute a $(d/(2L) - 1)$-defective vertex-coloring of F with $O\big((\frac{r\cdot d}{(d/(2L)-1)})^2\big) = O\big(L^2r^2\big)$ colors[7]. Exploiting the given $O\big(r^2\Delta^2\big)$-edge-coloring of H, and thus H_x, which is an $O\big(r^2\Delta^2\big)$-vertex-coloring of the line graph F, we can make this algorithm run in $O(\log^*(r\Delta))$ rounds. The vertex-coloring of the line graph with defect $d/(2L)-1$ is an edge-coloring of H_x where every edge has at most $d/(2L) - 1$ incident edges of the same color, resulting in at most $d/(2L)$ many edges of the same color incident to each vertex.

Part II, Fractional Matching Computation via Defective Coloring: We process the colors of the $d/(2L)$-defective coloring one by one, in $O\big(L^2r^2\big)$ iterations. In the i^{th} iteration, for each non-frozen edge e with color i, we raise y_e from $y_e = 0$ to $y_e = L/d$. Then for each node v that is already half-tight, meaning that $\sum_{e\in E_x(v)} y_e \geq 1/2$, we freeze all the edges incident to v. This means the fractional value of these edges will not be raised in the future. Notice that since we raise values only incident to nodes that are not already half-tight, and as for each such node the summation goes up by at most $\frac{d}{2L} \cdot \frac{L}{d} = 1/2$, the vector $\vec{y}$ always remains a fractional matching, meaning that we always have $\sum_{e\in E_x(v)} y_e \leq 1$ for each node v.

[7]If we happen to have $d/(2L) \leq 1$, then $(d/(2L)-1)$-defective coloring becomes a degenerate case of the definition, as $(d/(2L)-1) \leq 0$, and then by convention, this simply means proper coloring. In that case the algorithm of Kuhn [Kuh09] provides a proper coloring with $O(d^2r^2) = O(L^2r^2)$ colors.

At the very end, once we are done with processing all colors, some edges in E_x may remain without any half-tight endpoint. Though any such edge e would itself have $y_e = L/d$. We perform $\log(d/L)$ iterations of doubling, where in each iteration, we double all the fractional values y_e for all edges that do not have a half-tight endpoint. At the end, we are ensured that each edge has at least one half-tight endpoint, and moreover, each non-zero fractional value y_e is at least L/d.

Lemma II.6. *The above algorithm computes an (L/d)-fractional matching $\vec{y}$ such that $\sum_{e\in E} y_e \geq \frac{1}{2r}\sum_{e\in E} x_e$, in $O(L^2r^2 + \log(d/L) + \log^*(r\Delta)) = O(L^2r^2 + \log\Delta)$ rounds.*

Proof: The round complexity of the algorithm comes from the $O(\log^*(r\Delta))$ rounds spent for computing the defective edge-coloring, $O(L^2r^2)$ rounds for processing the colors of the defective coloring one by one, and then $O(\log(d/L))$ rounds for the final doubling steps.

It is clear by construction that the computed vector $\vec{y}$ is a fractional matching, because we always have $\sum_{e\in E_x(v)} y_e \leq 1$, and that it is (L/d)-fractional, because the smallest non-zero y_e value that we use is L/d. What remains to be proved is that $\sum_{e\in E} y_e \geq \frac{1}{2r}\sum_{e\in E} x_e$. For that, we use the property that the fractional matching $\vec{y}$ that we compute is such that for each $e \in E_x$, at least one of the vertices $v \in e$ must be half-tight, meaning that $\sum_{e\in E(v)} y_e \geq 1/2$. We use this property to argue that the fractional matching $\vec{y}$ has size at least a factor $1/(2r)$ of $\vec{x}$. This is done via a blaming argument along the same lines as the proof of Lemma II.2. We let every edge $e \in E_x$ put x_e dollars on edges $e' \in E_y$ as follows. Each edge e passes its x_e dollars to one of its half-tight vertices $v \in e$. Then, the half-tight vertex v distributes these x_e dollars among all its incident edges $e' \in E_x(v)$ proportionally to the values $y_{e'}$. As $\vec{x}$ is a fractional matching, in this way, v cannot receive more than 1 dollar in total from its incident edges in E_x. Therefore, and since v is half-tight, no edge e' incident to v receives more than $2y_{e'}$ dollars from $v \in e'$. In total, an edge $e' \in E_y$ can receive at most $2ry_{e'}$ dollars from edges in E_x, at most $2y_{e'}$ from each of its endpoints. Therefore, $\sum_{e\in E} x_e \leq 2r\sum_{e\in E} y_e$. ∎

D. Recursive Rounding

We explain a recursive method $\mathsf{round}(\vec{x}, L)$ that given a $(1/d)$-fractional matching $\vec{x}$ computes an (L/d)-fractional matching $\vec{y}$ such that $\sum_{e\in E} y_e \geq \frac{1}{4r}\sum_{e\in E} x_e$. This procedure will be applied when L is greater than some fixed constant. The procedure works mainly by a number of recursive calls to factor-$\sqrt{2L}$ rounding procedures, and a few additional steps.

Lemma II.7 (Recursive Rounding)**.** *There is an $O\Big((r^2 + \log\Delta)\log^{5+\log r} L\Big)$-round deterministic distributed algorithm that turns a $(1/d)$-fractional matching $\vec{x}$ into an (L/d)-fractional matching $\vec{y}$ with $\sum_{e\in E} y_e \geq \frac{1}{4r}\sum_{e\in E} x_e$, for any L such that $L\log^2 L \leq d$.*

The Recursive Rounding Algorithm: The method $\mathsf{round}(\vec{x}, L)$ consists of $16r$ iterations. Initially, we set $y_e = 0$ for all edges. Then, we gradually grow $\vec{y}$ while keeping it (L/d)-fractional.

The process in each iteration is as follows:

- We first generate a fractional matching $\vec{z}$ by initially setting it equal to $\vec{x}$, and then setting the value of each edge e that is incident to at least one half-tight vertex of $\vec{y}$ to 0. In other words, for each vertex v such that $\sum_{e\in E(v)} y_e \geq 1/2$, we set $z_e = 0$ for all e with $v \in e$; for all other edges, we set $z_e = x_e$.
- We first perform $\mathsf{round}(\vec{z}, \sqrt{2L})$, producing some intermediate $(\sqrt{2L}/d)$-fractional matching $\vec{z'}$. Then we perform $\mathsf{round}(\vec{z'}, \sqrt{2L})$. This creates a $(2L/d)$-fractional matching $\vec{z''}$ whose size is at least a factor $(\frac{1}{4r})\cdot(\frac{1}{4r}) = \frac{1}{16r^2}$ of the size of $\vec{z}$.
- We divide the values of this fractional matching $\vec{z''}$ by a factor 2, creating an (L/d)-fractional matching, and we add the result to $\vec{y}$. That is, we update $\vec{y} \leftarrow \vec{y} + \vec{z''}/2$.

Remark II.8. *Recall the promise from Remark II.4 that we will apply the rounding method only for values such that $L\log^2 L \leq d$. The main reason for this stronger condition, compared to the more natural condition of $L \leq d$, is the factor 2 that we have in the recursive rounding call. For instance, the matching $\vec{z''}$ is a $(2L/d)$-fractional matching and thus, for this to be meaningful, we need $2L \leq d$. However, with the stronger condition that $L\log^2 L \leq d$, we can say that the promise is satisfied throughout the recursive calls. For instance, in the second call to $\mathsf{round}(\vec{z'}, \sqrt{2L})$, the new condition would be $\sqrt{2L}(\log\sqrt{2L})^2 \leq d/\sqrt{2L}$, which is readily satisfied given that $L\log^2 L \leq d$ and $L \geq 8$.*

Analysis of the Recursive Rounding: We next provide the related analysis. In particular, Lemma II.9 proves that the generated fractional matching $\vec{y}$ is valid, Lemma II.10 proves that it is a good approximation of $\vec{x}$, and Lemma II.11 analyzes the running time of this recursive procedure.

Lemma II.9. *The fractional matching $\vec{y}$ is valid, meaning that $\sum_{e\in E(v)} y_e \leq 1$ for all vertices v.*

Proof: We show by induction on i that the fractional matching $\vec{y}$ in iteration i does not violate the constraints $\sum_{e\in E(v)} y_e \leq 1$ for all v. At the beginning, the condition is trivially satisfied. If v is half-tight at the beginning of an iteration, then $z_e = 0$ and hence $z''_e = 0$ for all $e \in E(v)$, thus no value is added to $\sum_{e\in E(v)} y_e$ in this iteration. If v is not half-tight at the beginning of an iteration, we add at most half of a fractional matching to edges incident to v, thus at most a value $1/2$ to the summation $\sum_{e\in E(v)} y_e$. More

formally, we have $\sum_{e\in E(v)} z''_e \le 1$, thus $\sum_{e\in E(v)} z''_e/2 \le \frac{1}{2}$, which results in a new value of at most $\sum_{e\in E(v)}(y_e + z''_e/2) \le 1$. ∎

Lemma II.10. *At the end of $16r$ iterations, we have $\sum_{e\in E} y_e \ge \frac{1}{4r}\sum_{e\in E} x_e$.*

Proof: Consider one iteration and suppose that $\sum_{e\in E} y_e \le \frac{1}{4r}\sum_{e\in E} x_e$. We first show that then $\sum_{e\in E} z_e \ge \frac{1}{2}\sum_{e\in E} x_e$ by a blaming argument along the same lines as the proof of Lemma II.2. For that, we let every edge $e \in E$ which is incident to a half-tight vertex in $\vec{y}$ put x_e dollars on edges in a manner that each edge e' receives at most $2ry_{e'}$ dollars. This can be done by sending those x_e dollars of e to (one of) its $\vec{y}$-half-tight vertex v, and then letting v distribute these x_e dollars among its incident edges $e' \in E(v)$ proportionally to the values $y_{e'}$. Since $\vec{x}$ is a matching, each vertex v in total receives at most 1 dollar from its incident edges. Then, since v is $\vec{y}$-half-tight, it can distribute this dollar among its incident edges such that no edge e' receives more than $2y_{e'}$ dollars from one of its endpoints v. Now, an edge $e' \in E$ can possibly receive $2y_{e'}$ dollars from each of its (half-tight) endpoint vertices, thus in total at most $2ry_{e'}$ dollars. Therefore, indeed the sum over the edge values for all $e \in E$ that have at least one half-tight vertex $v \in e$ is at most $2r\sum_{e\in E} y_e \le \frac{2r}{4r}\sum_{e\in E} x_e = \frac{1}{2}\sum_{e\in E} x_e$. It follows that if $\sum_{e\in E} y_e \le \frac{1}{4r}\sum_{e\in E} x_e$, then $\sum_{e\in E} z_e \ge \frac{1}{2}\sum_{e\in E} x_e$. Thus, in each such iteration, $\vec{y}$ grows by at least

$$\sum_{e\in E} z''_e/2 \ge \frac{1}{2}\cdot\frac{1}{16r^2}\sum_{e\in E} z_e \ge \frac{1}{2}\cdot\frac{1}{16r^2}\cdot\frac{1}{2}\sum_{e\in E} x_e.$$

Hence, after $16r$ iterations, $\sum_{e\in E} y_e \ge \frac{1}{4r}\sum_{e\in E} x_e$. ∎

Lemma II.11. *The algorithm* $\mathsf{round}(\vec{x}, L)$ *has round complexity*[8] $O((r^2+\log\Delta)\log^{5+\log r} L)$.

Proof: The complexity $R(L)$ of the rounding algorithm $\mathsf{round}(\vec{x}, L)$ follows the recursive inequality $R(L) \le 16r(R(\sqrt{2L}) + R(\sqrt{2L}) + O(1))$. Furthermore, we have the base case solution of $R(L) = O(L^2r^2 + \log\Delta)$ for $L = O(1)$. The claim can now be proved by an induction on L, as formalized in the full version [FGK17]. Here, instead of the formal calculations, we mention an intuitive explanation: the complexity gets multiplied by roughly $32r$ as we move from L to $\sqrt{2L}$. There are, roughly, $\log\log L$ such moves, and hence the complexity gets multiplied by $(32r)^{\log\log L} < \log^{5+\log r} L$ until we reach the base case of $L = O(1)$, where the base complexity is $O(r^2 + \log\Delta)$ by Lemma II.5. ∎

[8]We remark that we have not tried to optimize the constant that appears in the exponent of the round complexity $O((r^2+\log\Delta)\log^{5+\log r} L)$. This constant mainly comes from the constant in the number of iterations in our recursive rounding, which is currently set to $16r$, for simplicity. Optimizing this constant may be of interest specially for small values of r. We refer to the full version [FGK17] for a more thorough discussion.

E. Wrap-up

We now use our rounding procedure to find the approximate maximum matching of Lemma II.1.

Proof of Lemma II.1: First, we compute a $(1/\Delta)$-fractional $(2r)$-approximate matching $\vec{x}$ in $O(\log\Delta)$ rounds, by the greedy algorithm described in Section II-A. Then, we apply the recursive rounding from Lemma II.7 for $L = \Delta/\log^2\Delta$. This produces a $(1/\log^2\Delta)$-fractional matching $\vec{x'}$ whose size is a factor $1/(8r^2)$ of the maximum fractional matching of the hypergraph, in $O\Big(\log^{5+\log r}\Delta(r^2+\log\Delta)\Big)$ rounds. To finish up the rounding, we apply the basic rounding of Lemma II.5 for $L = \log^2\Delta$, which runs in $O(r^2\log^2\Delta)$ and produces a 1-fractional — i.e. integral — matching $\vec{x''}$ whose size is at least a factor $1/(4r)$ of the size of $\vec{x'}$. Hence, the final produced integral matching is a $(32r^3)$-approximation of the maximum matching. ∎

We compute a maximal matching via iterative applications of this matching approximation.

Proof of Theorem I.4: First, we pre-compute an $O(r^2\Delta^2)$-edge-coloring of H in $O(\log^* n)$ rounds by Linial's algorithm [Lin87]. Then, iteratively, we apply the maximum matching approximation procedure of Lemma II.1 to the remaining hypergraph. We add the found matching M to the matching that we will output at the end, and remove M along with its incident edges from the hypergraph.

In each iteration, the size of the maximum matching of the remaining hypergraph goes down to at least a factor of $1 - 1/(32r^3)$ of the previous size. This is because otherwise we could combine the matching computed so far with the maximum matching in the remainder hypergraph to obtain a matching larger than the maximum matching in H. After $O(r^3\log n)$ repetitions, the remaining maximum matching size is 0, which means the remaining hypergraph is empty. Hence, we have found a maximal matching in $O\left(\log^* n + r^3\log n\left(r^2\log^{6+\log r}\Delta\right)\right) = O(r^5\log^{6+\log r}\Delta\cdot\log n)$ rounds. ∎

III. Edge-Coloring

In this section, we show how the hypergraph maximal matching algorithm can be employed for list-edge-coloring. In the list-edge-coloring problem, each edge e must choose its color from an arbitrary given list L_e of colors with $|L_e| = d_e + 1$, where d_e is the number of edges adjacent to e.

Lemma III.1. *Given a deterministic distributed algorithm $\mathcal{A}$ that computes a maximal matching in N-vertex hypergraphs of rank 3 and maximum degree d in $T(N,d)$ rounds, there is a deterministic distributed algorithm $\mathcal{B}$ that solves list-edge-coloring of any n-node graph $G = (V,E)$ with maximum degree Δ in at most $T(2n\Delta^2, 2\Delta-1)$ rounds.*

Proof Sketch: To edge-color $G = (V,E)$, we generate a hypergraph H: For each edge $e \in E$ with list-color L_e,

we take $|L_e|$ copies of $e = \{v, u\}$ as follows: For each color $i \in L_e$, we take one copy e_i of e which is put incident to copies v_i and u_i of v and u. Thus, if two adjacent edges $e = \{v, u\}$ and $e' = \{v, u'\}$ have a common color $i \in L_e \cap L_{e'}$, then their i^{th} copies e_i and e'_i will be present and will both be incident to v_i. Notice that for each vertex v, at most $2\Delta^2$ copies of it will be used because for each of the edges e incident to v, at most $|L_e| < 2\Delta$ additional copies of v are added.

Algorithm $\mathcal{B}$ runs the maximal matching algorithm $\mathcal{A}$ on H, and then, for each edge $e \in E$, if the copy e_i of e is in the computed maximal matching, $\mathcal{B}$ colors e with color i. One can verify that each G-edge e has exactly one copy e_i in the maximal matching. ■

Theorem I.1. *There is a deterministic distributed algorithm that computes a $(2\Delta-1)$-edge-coloring in $O(\log^8 \Delta \cdot \log n)$ rounds, in any n-node graph with maximum degree Δ. Moreover, the same algorithm solves list-edge-coloring, where each edge $e \in E$ must get a color from a given list L_e of colors with $|L_e| = d_e + 1$, where d_e denotes the number of edges adjacent to e.*

Proof: Follows from Lemma III.1 and Theorem I.4. ■

Remark III.2. *We note that Lemma III.1, and hence also Theorem I.1, can be easily extended from graphs to hypergraphs. In particular, list-edge-coloring of hypergraphs of rank r can be reduced to maximal matching in hypergraphs of rank $r+1$. Thus, we can obtain a deterministic list-edge-coloring algorithm for hypergraphs of rank r with round complexity $O(r^5 \log^{6+\log(r+1)} \Delta \log n)$ rounds.*

As stated before, the deterministic list-edge-coloring algorithm of Theorem I.1, in combination with known randomized algroithms of Elkin, Pettie, and Su [EPS15] and Johansson [Joh99], leads to a $\text{poly}(\log\log n)$ randomized algorithm for $(2\Delta - 1)$-edge-coloring.

Corollary I.2. *There is a randomized distributed algorithm that computes a $(2\Delta - 1)$-edge-coloring in $O(\log^9 \log n)$ rounds, with high probability.*

Proof of Corollary I.2: For $\Delta = \Omega(\log^2 n)$, we run the $O\Big(\log^* \Delta + \frac{\log n}{\Delta^{1-o(1)}}\Big)$-round algorithm by Elkin, Pettie, and Su [EPS15] for $((1+\varepsilon)\Delta)$-edge-coloring, in $O(\log^* \Delta)$ rounds. For $\Delta = o(\log^2 n)$, we first apply the simple randomized coloring algorithm of Johansson [Joh99] for $O(\log \Delta) = O(\log\log n)$ rounds. In particular, in each iteration, every remaining edge e independently picks a color q_e from its remaining palette uniformly at random. If there is no incident edge that picked the same color q_e, then the edge e is colored with this color q_e and removed from the graph. Moreover, the color q_e gets deleted from the palettes of every incident edge. As proved in e.g. [BEPS12], after $O(\log \Delta)$ rounds, this procedure leaves us with a graph where each connected component of remaining edges has size at most $N = \text{poly}\log n$. On these components, we then run the list-edge-coloring algorithm of Theorem I.1 to complete the partial coloring. This takes at most $O(\log^9 N) = O(\log^9 \log n)$ rounds. Hence, including the $O(\log\log n)$ initial rounds, the overall complexity is $O(\log^9 \log n)$ rounds. ■

IV. Open Problems

We believe that our techniques and results open the road for further progress on deterministic distributed graph algorithms, with clear consequences also on randomized algorithms, as exemplified by Corollary I.2. As Barenboim and Elkin suggested when discussing their Open Problem 5, perhaps these will serve as a "*good stepping stone*" towards obtaining an efficient deterministic algorithm for MIS, thus resolving Linial's long-standing question [Lin87]. As concrete steps on this path, we point out two smaller problems, which appear to be the immediate next steps.

Hypergraph Maximal Matching with Better Rank Dependency: For our hypergraph maximal matching algorithm, we have been more focused on the case of smaller ranks r, and the current complexity has a factor of $\log^{\log r} \Delta$ in it. Can we improve this to $\text{poly}(r \log n)$, for instance? Notice that the case of $r = \text{poly}\log n$ captures a range of problems of interest, see e.g. the full version of this paper [FGK17], and this improvement would give a $\text{poly}\log n$-time algorithm for these cases, including a resolution of Open Problem 10 of Barenboim and Elkin's book [BE13].

Better than $(2\Delta - 1)$-Edge-Coloring: We obtained a polylogarithmic-time algorithm for $(2\Delta-1)$-edge-coloring, as formalized in Theorem I.1. This value of $2\Delta - 1$ is a natural threshold, because this is what greedy sequential arguments obtain, which made it the classic target of (deterministic) distributed algorithms. However, as Vizing's theorem [Viz64] shows, every graph has a $(\Delta + 1)$-edge-coloring. How close can we get to this, while remaining with polylogarithmic-time LOCAL algorithms?

We are confident that by combining Theorem I.1 with ideas of Panconesi and Srinivasan [PS95] for Δ-vertex-coloring, we can obtain a polylogarithmic-time algorithm for $(2\Delta-2)$-edge-coloring. But how about $(2\Delta-3)$-edge-coloring, or even $(3\Delta/2)$-edge-coloring?

Interestingly, we can already make some progress on this question for graphs with small arboricity. This result is achieved by combining our list-edge-coloring algorithm of Theorem I.1 with an H-partitioning method of Barenboim and Elkin [BE13, Chapter 5.1]. This significantly generalizes the $(\Delta + o(\Delta))$-edge-coloring results of [BEM17], which worked for $a \leq \Delta^{1-\delta}$ for some constant $\delta > 0$. See [CV10] for a study of distributed edge-coloring in high-girth graphs.

Corollary IV.1. *There is a deterministic distributed algorithm that computes an edge-coloring with $\Delta + (2+\varepsilon)a - 1$ colors in $O(\frac{1}{\varepsilon} \log^8 \Delta \log^2 n)$ rounds, on any n-node graph $G = (V, E)$ with maximum degree Δ and arboricity a.*

Notice that any graph has arboricity $a \leq \lfloor(\Delta+1)/2\rfloor$. The above corollary shows that we start seeing savings in the number of colors as soon as the arboricity goes slightly below this upper bound, e.g., for $a < \Delta(1-\varepsilon)/2$, we already get colorings with less than $2\Delta-2$ colors (for $\Delta \geq 2$).

Proof of Corollary IV.1: First, we compute an H-partitioning [BE13, Chapter 5.1] in $O(\log n/\varepsilon)$ rounds. This decomposes V into disjoint vertex sets $H_1, H_2, \ldots, H_\ell$, for $\ell = O(\log n/\varepsilon)$, with the property that each node in H_i has degree at most $(2+\varepsilon)a$ in the graph $G[\cup_{j=i}^{\ell} H_j]$. To compute this decomposition, one just needs to iteratively peel vertices of degree at most $(2+\varepsilon)a$ from the remaining graph.

Having this partitioning, we compute a $(\Delta+(2+\varepsilon)a-1)$-edge-coloring by gradually moving backwards in this partition, from H_ℓ towards H_1. Each step is as follows. Suppose we already have a coloring of edges of $G[\cup_{j=i+1}^{\ell} H_j]$. We now introduce the vertices of H_i and also their edges whose other endpoint is in $\cup_{j=i}^{\ell} H_j$. Each such edge e has at most $(2+\varepsilon)a-1$ other incident edges on the side of its H_i-endpoint and at most $\Delta-1$ other incident edges on the other endpoint. If we take away the colors of $\{1,2,\ldots,\Delta+(2+\varepsilon)a-1\}$ that are already used by neighboring edges e' whose both endpoints are in $\cup_{j=i+1}^{\ell} H_j$, the edge e would still have at least d_e+1 remaining colors in its palette, where d_e is the number of edges in $G[\cup_{j=i}^{\ell} H_j]$ incident on e who remain uncolored. Hence, we can color all these edges by applying the list-edge-coloring algorithm of Theorem I.1, in $O(\log^8 \Delta \log n)$ rounds. This is the round complexity needed for coloring new edges after introducing each layer H_i. Hence, the overall complexity until we go through all the ℓ layers and finish the edge-coloring of $G = G[\cup_{j=1}^{\ell} H_j]$ is $\ell \cdot O(\log^8 \Delta \log n) = O(\frac{1}{\varepsilon}\log^8 \Delta \log^2 n)$. ∎

ACKNOWLEDGMENT

We are grateful to Moab Arar and Shiri Chechik for sharing with us their manuscript about distributed matching approximation in graphs [AC17]. We note that we arrived at a prior (and slower) version of Lemma II.5 (for rank-3 hypergraphs) inspired by a concept they use, called the *kernel of a graph*, which itself is borrowed from [BHN16].

REFERENCES

[ABI86] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms*, 7(4):567–583, 1986.

[AC17] Moab Arar and Shiri Chechik. A distributed deterministic $(2+\epsilon)$-approximation for maximum matching in $O(\Delta^{o(1)})$ rounds. *Manuscript*, 2017.

[ALGP89] Baruch Awerbuch, Michael Luby, Andrew V. Goldberg, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 364–369. IEEE, 1989.

[BE11] Leonid Barenboim and Michael Elkin. Distributed deterministic edge coloring using bounded neighborhood independence. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 129–138, 2011.

[BE13] Leonid Barenboim and Michael Elkin. Distributed graph coloring: Fundamentals and recent developments. *Synthesis Lectures on Distributed Computing Theory*, 4(1):1–171, 2013.

[BEM17] Leonid Barenboim, Michael Elkin, and Tzalik Maimon. Deterministic distributed (Δ+ o(Δ))-edge-coloring, and vertex-coloring of graphs with bounded diversity. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 175–184, 2017.

[BEPS12] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 321–330, 2012.

[BHN16] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. New deterministic approximation algorithms for fully dynamic matching. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 398–411. ACM, 2016.

[CH03] Andrzej Czygrinow and Michał Hańćkowiak. Distributed algorithm for better approximation of the maximum matching. In *International Computing and Combinatorics Conference*, pages 242–251, 2003.

[CHK01] Andrzej Czygrinow, M Hańćkowiak, and M Karoński. Distributed $O(\Delta \log n)$-edge-coloring algorithm. In *European Symposium on Algorithms*, pages 345–355. Springer, 2001.

[CV10] Flavio Chierichetti and Andrea Vattani. The local nature of list colorings for graphs of high girth. *SIAM Journal on Computing*, 39(6):2232–2250, 2010.

[DGP98] Devdatt Dubhashi, David A Grable, and Alessandro Panconesi. Near-optimal, distributed edge colouring via the nibble method. *Theoretical Computer Science*, 203(2):225–251, 1998.

[EPS15] Michael Elkin, Seth Pettie, and Hsin-Hao Su. (2Δ-1)-edge-coloring is much easier than maximal matching in the distributed setting. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 355–370. Society for Industrial and Applied Mathematics, 2015.

[FGK17] Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. Deterministic distributed edge-coloring via hypergraph maximal matching. *CoRR*, abs/1703.00900, 2017. http://arxiv.org/abs/1704.02767.

[FHK16] Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local conflict coloring. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 625–634. IEEE, 2016.

[Fis17] Manuela Fischer. Improved deterministic distributed matching via rounding. *Proceedings of the International Symposium on Distributed Computing (DISC)*, 2017. http://arxiv.org/abs/1703.00900.

[Gha16] Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 270–277, 2016.

[GKM17] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 784–797, 2017.

[GS17] Mohsen Ghaffari and Hsin-Hao Su. Distributed degree splitting, edge coloring, and orientations. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2505–2523, 2017.

[HK73] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[HKP98] Michał Hańćkowiak, Michał Karonski, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 219–225, 1998.

[HKP99] Michał Hańćkowiak, Michał Karoński, and Alessandro Panconesi. A faster distributed algorithm for computing maximal matchings deterministically. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 219–228, 1999.

[HSS16] David G. Harris, Johannes Schneider, and Hsin-Hao Su. Distributed (Δ+ 1)-coloring in sublogarithmic rounds. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 465–478, 2016.

[Joh99] Öjvind Johansson. Simple distributed $(\Delta + 1)$-coloring of graphs. *Information Processing Letters*, 70(5):229–232, 1999.

[KMNW05] Fabian Kuhn, Thomas Moscibroda, Tim Nieberg, and Roger Wattenhofer. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. *Distributed Computing*, pages 273–287, 2005.

[KMW06] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 980–989, 2006.

[KMW16] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *J. ACM*, 63(2):17:1–17:44, March 2016.

[KNPR14] Shay Kutten, Danupon Nanongkai, Gopal Pandurangan, and Peter Robinson. Distributed symmetry breaking in hypergraphs. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, pages 469–483. Springer, 2014.

[Kuh09] Fabian Kuhn. Weak graph colorings: distributed algorithms and applications. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 138–144. ACM, 2009.

[KY09] Christos Koufogiannakis and Neal E. Young. Distributed fractional packing and maximum weighted b-matching via tail-recursive duality. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, pages 221–238, 2009.

[Lin87] Nathan Linial. Distributive graph algorithms - global solutions from local data. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 331–335, 1987.

[Lin92] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[LPSP08] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 129–136, 2008.

[Lub86] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986.

[Pel00] David Peleg. *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[PR01] Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. *Distributed computing*, 14(2):97–100, 2001.

[PS92] Alessandro Panconesi and Aravind Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 581–592. ACM, 1992.

[PS95] Alessandro Panconesi and Aravind Srinivasan. The local nature of Δ-coloring and its algorithmic applications. *Combinatorica*, 15(2):255–280, 1995.

[PS97] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff–Hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.

[SW10] Johannes Schneider and Roger Wattenhofer. An optimal maximal independent set algorithm for bounded-independence graphs. *Distributed Computing*, 22(5):349–361, 2010.

[Viz64] Vadim G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret Analiz*, 3(7):25–30, 1964.