Distributed local approximation algorithms for maximum matching in graphs and hypergraphs

David G. Harris
Department of Computer Science
University of Maryland
College Park, USA
Email: davidgharris29@gmail.com

***Abstract*—We describe approximation algorithms in Linial's classic LOCAL model of distributed computing to find maximum-weight matchings in a hypergraph of rank r. Our main result is a deterministic algorithm to generate a matching which is an $O(r)$-approximation to the maximum weight matching, running in $\tilde{O}(r \log \Delta + \log^2 \Delta + \log^* n)$ rounds. (Here, the $\tilde{O}()$ notations hides $\operatorname{polyloglog} \Delta$ and $\operatorname{polylog} r$ factors). This is based on a number of new derandomization techniques extending methods of Ghaffari, Harris & Kuhn (2017).**

The first main application is to nearly-optimal algorithms for the long-studied problem of maximum-weight graph matching. Specifically, we get a $(1+\epsilon)$ approximation algorithm using $\tilde{O}(\log \Delta/\epsilon^3 + \operatorname{polylog}(1/\epsilon, \log \log n))$ randomized time and $\tilde{O}(\log^2 \Delta/\epsilon^4 + \log^* n/\epsilon)$ deterministic time.

The second application is a faster algorithm for hypergraph maximal matching, a versatile subroutine introduced in Ghaffari et al. (2017) for a variety of local graph algorithms. This gives an algorithm for $(2\Delta-1)$-edge-list coloring in $\tilde{O}(\log^2 \Delta \log n)$ rounds deterministically or $\tilde{O}((\log \log n)^3)$ rounds randomly. Another consequence (with additional optimizations) is an algorithm which generates an edge-orientation with out-degree at most $\lceil (1+\epsilon)\lambda \rceil$ for a graph of arboricity λ; for fixed ϵ this runs in $\tilde{O}(\log^6 n)$ rounds deterministically or $\tilde{O}(\log^3 n)$ rounds randomly.

***Keywords*-matching; hypergraph; derandomization; LOCAL; edge coloring; Nash-Williams decomposition**

I. Introduction

Consider a hypergraph $H = (V, E)$ with rank (maximum edge size) at most r. A matching of H is an edge subset $M \subseteq E$, in which all edges of M are pairwise disjoint. Equivalently, it is an independent set of the line graph of H. The problem of *Hypergraph Maximum Weight Matching (HMWM)* is, given some edge-weighting function $a : E \to [0, \infty)$, to find a matching M whose weight $a(M) = \sum_{e \in M} a(M)$ is maximum. This is often intractable to solve exactly; we define a *c-approximation algorithm for HMWM* to be an algorithm which returns a matching M whose weight $a(M)$ is at least $1/c$ times the maximum matching weight.

We develop distributed algorithms in Linial's classic LOCAL model of computation [22] for approximate HMWM. In this model, time proceeds synchronously and each vertex in the hypergraph can communicate with any other vertex sharing a hyperedge. Computation and message size are unbounded. There are two main reasons for studying hypergraph matching in this context. First, many of the same symmetry-breaking and locality issues are relevant to hypergraph theory as they are to graph theory. In particular, graph maximal matching is one of the "big four" symmetry-breaking problems (which also includes maximal independent set (MIS), vertex coloring, and edge coloring) [28]. It is a natural extension to generalize these problems to the richer setting of hypergraphs.

But there is a second and more important reason for studying hypergraph matching. Recently, Fischer, Ghaffari, & Kuhn [9] showed that distributed hypergraph algorithms are clean subroutines for a number of diverse graph algorithms. Some example applications include maximum-weight matching, edge-coloring and Nash-Williams decomposition [12], [13]. In such algorithms, we need to find a maximal, or high-weight, collection of disjoint "augmenting paths" (in various flavors); by committing to it, we iteratively improve our solution to the underlying graph problem. These paths can be represented by an auxiliary hypergraph, and a disjoint collection of such paths corresponds to a hypergraph matching.

The main contribution of this paper is a deterministic LOCAL algorithm to approximate HMWM.

Theorem I.1 (Simplified)**.** *For a hypergraph of maximum degree Δ and rank r, there is a deterministic $\tilde{O}(r \log \Delta + \log^2 \Delta + \log^* n)$-round algorithm for an $O(r)$-approximation to HMWM.*

Randomization reduces the run-time still further and yields a truly local algorithm:

Theorem I.2. *For any $\delta \in (0, 1/2)$, there is a randomized $\tilde{O}(\log \Delta + r \log \log \frac{1}{\delta} + (\log \log \frac{1}{\delta})^2)$-round algorithm to get an $O(r)$-approximation to HMWM with probability at least $1 - \delta$.*

We discuss a number of applications to graph algorithms. Of these applications, by far the most important is *Graph Maximum Weight Matching (GMWM)*, which is one of the

2575-8454/19/$31.00 ©2019 IEEE
DOI 10.1109/FOCS.2019.00048

most well-studied problems in all of algorithmic graph theory. Since GMWM has been so widely developed, we summarize it next.

A. Graph matching

Many variations in the computational model have been studied for GMWM, including algorithms for specialized graph classes and other unique properties; we cannot summarize this full history here. Overall, there are three main paradigms for approximation algorithms. The first is based on finding a *maximal matching*, which is a 2-approximation to (unweighted) graph maximum matching. Various quantization methods can transform this into an approximation to the weighted case as well [24]. The best algorithms for maximal matching are a deterministic algorithm of [8] in $O(\log^2 \Delta \log n)$ rounds, and a randomized algorithm in $O(\log \Delta + (\log \log n)^3)$ rounds (a combination of a randomized algorithm of [3] with the deterministic algorithm of [8]).

The second paradigm is based on finding and rounding a fractional matching. For general graphs, the fractional matching LP (to be distinguished from the matching polytope) has an integrality gap of $3/2$, and so these algorithms typically achieve an approximation ratio of $3/2$ at best, sometimes with additional loss in the rounding. The most recent example is the deterministic algorithm of Ahmadi, Kuhn, & Oshman [1] to obtain a $(3/2+\epsilon)$-approximate maximum matching in $O(\log W + \log^2 \Delta + \log^* n)$ rounds for fixed ϵ, where W is the ratio between maximum and minimum edge weight. Note that this run-time, unlike those based on maximal matching, has a negligible dependence on n. (For restricted graph classes, this approach can give better approximation ratios. For example, the algorithm of [1] yields a $(1+\epsilon)$ approximation for bipartite graphs.)

In order to get an approximation factor arbitrarily close to one in general graphs, it appears necessary to use the third type of approximation algorithm based on *path augmentation*. These algorithms successively build up the matching by iteratively finding and applying short augmenting path. The task of finding these paths can be formulated as a hypergraph matching problem.

Our focus here will be on this $(1+\epsilon)$-approximate GMWM problem in the LOCAL model. In addition to the run-time, there are some other important properties to keep in mind. The first is how the algorithm depends on the dynamic range W of the edge weights. Some algorithms only work in the case $W = 1$, i.e. maximum cardinality matching. We refer to these as *unweighted* algorithms. Other algorithms may have a run-time scaling logarithmically in W. Note that, after some quantization steps, we can often assume that $W \leq \text{poly}(n)$ without loss of generality.

The second property is the message size. Our focus is on the LOCAL graph model, in which message size is unbounded. A more restrictive model CONGEST is often used, in which message sizes are limited to $O(\log n)$ bits per vertex per round.

A third property is the role of randomness. We have the usual dichotomy in local algorithms between randomized and deterministic. It is traditional in randomized algorithms to seek success probability $1 - 1/\text{poly}(n)$, known as *with high probability* (w.h.p.). Some GMWM algorithms give a weaker guarantee that the algorithm returns a matching M whose *expected weight* is within a $(1+\epsilon)$ factor of the maximum weight. Note that when W is large, we cannot expect any meaningful concentration for the matching weight, since a single edge may contribute disproportionately. We refer to this as a *first-moment* probabilistic guarantee.

We summarize a number of $(1+\epsilon)$-approximation GMWM algorithms in Table I on the following page, listed roughly in order of publication. For readability, we have simplified the run-time bounds, omitting some asymptotic notations as well as dropping some second-order terms.

Our HMWM algorithms yields a $(1+\epsilon)$-approximation algorithm for graph matching:

Theorem I.3. *For $\epsilon > 0$, there is a $\tilde{O}(\epsilon^{-4} \log^2 \Delta + \epsilon^{-1} \log^* n)$-round deterministic algorithm to get a $(1+\epsilon)$-approximate GMWM.*

For any $\delta \in (0, 1/2)$, there is $\tilde{O}(\epsilon^{-3} \log \Delta + \epsilon^{-3} \log \log \frac{1}{\delta} + \epsilon^{-2} (\log \log \frac{1}{\delta})^2)$-round randomized algorithm to get a $(1+\epsilon)$-approximate GMWM with probability at least $1 - \delta$.

For a first-moment bound, we can take $\delta = \epsilon$ getting $\tilde{O}(\epsilon^{-3} \log \Delta)$ run-time. For success w.h.p., we can take $\delta = 1/\text{poly}(n)$ getting $\tilde{O}(\epsilon^{-3} \log \Delta + \epsilon^{-3} \log \log n + \epsilon^{-2} (\log \log n)^2)$ run-time.

Both the randomized and deterministic algorithms here improve qualitatively over prior GMWM approximation algorithms. These are the first $(1+\epsilon)$-approximation algorithms (either randomized or deterministic) that simultaneously have three desirable properties: (1) run-time essentially independent of n; (2) allowing arbitrary edge weights, without any assumptions or run-time dependence on the maximum edge weight W; and (3) a polynomial dependence on $1/\epsilon$.

Note that for any fixed ϵ, the deterministic algorithm has a run-time which matches that of the fastest prior *constant-factor* approximation algorithms [1], [8], which are based on using alternating paths to convert fractional matchings into integral matchings.

For lower bounds, [19] shows that $\Omega(\min(\sqrt{\frac{\log n}{\log \log n}}, \frac{\log \Delta}{\log \log \Delta}))$ rounds are needed for any constant-factor approximation to GMWM, even for randomized algorithms. Thus, these algorithm run-times are close to optimal, and the randomized algorithm has optimal run-time up to factors of $\log \log \Delta$.

Ref	Random?	Message size	run-time	Weighted?
[6]	Det	LOCAL	$(\log n)^{O_\epsilon(1)}$	No
[23]	W.h.p.	LOCAL	$\epsilon^{-3}\log n$	No
[23]	W.h.p.	CONGEST	$2^{1/\epsilon}\log n$	No
[27]	W.h.p.	LOCAL	$\epsilon^{-3}\log n$	Yes
[7]	Det	LOCAL	$\Delta^{1/\epsilon}+\epsilon^{-2}\log^* n$	No
[7]	Det	LOCAL	$(\log Wn)^{1/\epsilon}(\Delta^{1/\epsilon}+\log^* n)$	Yes
[4]	First-moment	CONGEST	$2^{1/\epsilon}\frac{\log\Delta}{\log\log\Delta}$	No
[9]	Det	CONGEST	$\Delta^{1/\epsilon}+\text{poly}(1/\epsilon)\log^* n$	No
[12]	Det	LOCAL	$\epsilon^{-9}\log^5\Delta\log^2 n$	No
[13] + [12]	Det	LOCAL	$\epsilon^{-7}\log^4\Delta\log^3 n$ (for $\Delta\ll n$)	Yes
This paper	Det	LOCAL	$\epsilon^{-4}\log^2\Delta+\epsilon^{-1}\log^* n$	Yes
This paper	W.h.p.	LOCAL	$\epsilon^{-3}\log\Delta+\text{poly}(\log\log n,1/\epsilon)$	Yes
This paper	First-moment	LOCAL	$\epsilon^{-3}\log\Delta$	Yes

Table I
COMPARISON OF GRAPH $(1+\epsilon)$-APPROXIMATE MAXIMUM MATCHING ALGORITHMS.

B. Maximal matching and other applications

The HMWM algorithm can be used to solve a closely related problem, that of hypergraph *maximal* matching (HMM):

Theorem I.4 (Simplified)**.** *There is a $\tilde{O}((\log n)(r^2\log\Delta+r\log^2\Delta))$-round deterministic algorithm and a $\tilde{O}(r\log^2\Delta+r^2(\log\log n)^2+r(\log\log n)^3)$-round randomized algorithm for hypergraph maximal matching.*

This deterministic algorithm is significantly faster than the algorithm of [12] which required $O(r^2\log(n\Delta)\log n\log^4\Delta)$ rounds.

We remark that [2] has shown that maximal matching, even in graphs, cannot be solved $o(\Delta+\frac{\log n}{\log\log n})$ rounds deterministically or $o(\Delta+\frac{\log n}{\log\log n})$ rounds randomly. Thus, as a function of n, the deterministic algorithm here is nearly optimal (up to factors of $\log\log n$), and the randomized algorithm is optimal up to polynomial factors of $\log\log n$.

Using Theorem I.4 as a subroutine, we immediately get improved distributed algorithms for a number of graph problems. Three simple consequences are for edge coloring:

Theorem I.5. *Let G be a graph with maximum degree Δ.*

1) *There is a $\tilde{O}(\log n\log^2\Delta)$-round deterministic algorithm for $(2\Delta-1)$-list-edge-coloring.*
2) *There is a $\tilde{O}((\log\log n)^3)$-round randomized algorithm for $(2\Delta-1)$-list-edge-coloring*
3) *There is a $\tilde{O}(\Delta^4\log^6 n)$-round deterministic algorithm for $\frac{3}{2}\Delta$-edge-coloring.*

By contrast, the best previous algorithms for the first two problems [12] require $O(\log^2 n\log^4\Delta)$ and $O((\log\log n)^6)$ rounds respectively. The best previous algorithms for the third problem [13] requires $\Delta^9\,\text{polylog}(n)$ rounds (the exponent of $\log n$ is not specified, but it is much larger than 6.)

One more involved application of HMM (along with some additional optimizations) is the following approximate Nash-Williams graph decomposition:

Theorem I.6. *For a multi-graph G of arboricity λ there is a $\tilde{O}(\log^6 n/\epsilon^4)$-round deterministic algorithm and a $\tilde{O}(\log^3 n/\epsilon^3)$-round randomized algorithm to find an edge-orientation of G with maximum out-degree $\lceil(1+\epsilon)\lambda\rceil$.*

This hugely improves over the deterministic algorithm of [12] which requires $O(\log^{10} n\log^5\Delta/\epsilon^9)$ rounds and over the randomized algorithm of [14] which requires $O(\log^4 n/\epsilon^3)$ rounds.

C. Overview and outline

Our algorithm is based on a simple randomized rounding procedure which we refer to as *direct rounding*. Suppose we are given a maximum-weight *fractional* matching h for some edge-weighting function a (this can be found using an LP solving procedure of [19]). This can be viewed as a function $h: E\to[0,1]$ satisfying $\sum_{e:v\in e}h(e)\le 1$ for all vertices v.

Consider the process wherein each edge is selected independently with probability $h(e)\log r$; for any vertex with more than $c\log r$ selected edges, for some constant $c>1$, we discard all such edges. Although the resulting edge-set L is not a matching, it is almost as good as one since the maximum degree is just $O(\log r)$. One can easily check that L has expected weight of $\Omega(\frac{\sum_{e\in E}a(e)h(e)}{\log r})$. Since L has such a small degree, it is inexpensive to convert it into a matching of weight $\Omega(\frac{\sum_{e\in L}a(e)}{r\log r})=\Omega(\frac{\sum_{e\in E}a(e)h(e)}{r})$, which is an $O(r)$-approximation to HMWM.

The crux of our algorithm, and the most important technical contribution of this paper, is derandomization of direct rounding. We use three main derandomization techniques. These all build on each other, and may be of interest in other settings.

In Section II, we describe the first technique for derandomizing general LOCAL graph algorithms. This is an adaptation of [12] using the method of conditional expectations via a proper vertex coloring. Roughly speaking, in each stage i, all the vertices of color i select a value for their random bits to ensure that the conditional expectation of some statistic of interest does not decrease. The objective function here acts in a black-box way and can be almost completely arbitrary.

In Section III, we develop our second derandomization technique, which extends this first method to allow a *non-proper* vertex coloring. This allows fewer colors, leading to a faster run-time. This new algorithm is fundamentally white-box: it requires the use of a carefully tailored pessimistic estimator for the conditional expectation. To state it somewhat informally, the estimator must be "multilinear" with respect to the coloring. This allows all the vertices of a given color to simultaneously make their decisions without non-linear interactions.

While this is a significant restriction, we also show that such a pessimistic estimator comes naturally for Chernoff bounds. To explain this at at a very high level, concentration bounds with probabilities of order δ require bounds on the joint behavior of w-tuples of vertices for $w = O(\log \frac{1}{\delta})$. For an appropriately chosen improper vertex coloring, most such w-tuples will have vertices of different colors.

In Section IV, we turn this machinery to derandomize direct rounding. To do so, we slow down the random process: instead of selecting the edges with probability $p = h(e) \log r$ at once, we go through multiple stages in which each edge is selected with probability $1/2$. We then use the conditional expecations method at each stage, ensuring that the weight of the retained edges at the end is at least the expected weight initially; we have seen this is $\Omega(\frac{\sum_e a(e)h(e)}{\log r})$. This is the most technically complex part of the paper. The statistic is a complex, non-linear function, so instead of directly computing its conditional expectation, we carefully construct a family of pessimistic estimators which approximate it but are amenable to the derandomization method in Section III.

This algorithm runs in $\tilde{O}(r \log \Delta + \log^2 \Delta)$ rounds to generate the $O(r)$-approximate maximum weight matching. In particular this is scale-free, without dependence on n. We find it somewhat remarkable that it is possible to deterministically optimize a global statistic $\sum_{e \in L} a(e)$ in an almost completely local way. (Although this part of the algorithm is completely local, it still depends on finding an appropriate proper vertex coloring, which requires $O(\log^* n)$ rounds.)

In Section V, we describe how to perform the initial step of obtaining the fractional matching. In addition to the LP solving algorithm of [19], we employ a few additional quantization steps. For the randomized algorithm, we also randomly sparsify the original graph, reducing the degree from Δ to $\log \frac{1}{\delta}$ where δ is the desired failure probability, and then we execute the deterministic algorithm. Note that the deterministic algorithm is obtained by derandomizing direct rounding, and the randomized algorithm is obtained by "randomizing" the deterministic algorithm. The resulting randomized algorithm, after these two transformations, has a failure probability which is exponentially smaller than direct rounding.

In Section VI, we use approximate HMWM to get a $(1+\epsilon)$-approximation to GMWM. As we have discussed, the GMWM algorithm repeatedly finds and applies a collection of disjoint high-weight augmenting paths. Our hypergraph matching algorithm can be used for this step of finding the paths. After multiple iterations of this process the resulting graph matching is close to optimal. It is critical here that our algorithm finds a *high-weight* hypergraph matching, not merely a maximal matching. We also provide further details on lower bounds for GMWM.

In Section VII, we discuss HMM, including an application to edge-list-coloring. The basic algorithm for HMM is simple: at each stage, we try to find a hypergraph matching of maximum cardinality in the residual graph. Our HMWM approximation algorithm ensures that the maximum matching cardinality in the residual graph decreases by a $1 - 1/r$ factor in each stage. Thus we have a maximal matching after $O(r \log n)$ repetitions. For the randomized algorithm, we also take advantage of the "shattering method" of [3]. Our algorithm here is quite different from a standard shattering-method construction, so we provide a self-contained description in this paper.

In Section VIII, we describe a more elaborate application of HMM to approximate Nash-Williams decomposition. We describe both randomized and deterministic algorithms for this task. Counter-intuitively, the deterministic algorithm is built on our *randomized* HMM algorithm.

D. Comparison with related work

The basic framework of our algorithm, like that of [9], [12], is to start with a fractional matching h, and then round it to an integral one (at some loss to its weight). This can be interpreted combinatorially as the following problem: given a hypergraph $H = (V, E)$, find a matching of weight approximately $\frac{\sum_{e \in E} a(e)}{\Delta r}$ for some edge-weighting function a. Let us summarize the basic approach taken by [9] and [12] and how we are improve the complexity.

The algorithm of [9] gives an $O(r^3)$-approximation to HMWM in $O(r^2(\log \Delta)^{6+\log r} + \log^* n)$ rounds. It is based on a primal-dual method: the fractional matching is gradually rounded, while at the same time a vertex cover (which is the dual problem to maximum matching) is maintained to witness its optimality.

The algorithm of [12], like ours, is based on derandomized rounding. They only aim for a hypergraph maximal matching, not an approximate HMWM. The key algorithmic subroutine for this is *degree-splitting*: namely, selecting an

edge-set $E' \subseteq E$ which has degree at most $\frac{\Delta}{2}(1+\epsilon)$ and which contains approximately half of the edges. This has a trivial 0-round randomized algorithm, by selecting edges independently with probability $1/2$. To derandomize this, [12] divides H into "virtual nodes" of degree $\frac{\log n}{\epsilon^2}$. They then get a proper vertex coloring of the resulting line graph (which has maximum degree $\frac{r \log n}{\epsilon^2}$), and derandomize using the method of conditional expectations.

To reduce the degree further, [12] repeats this edge-splitting process for $s \approx \log_2 \Delta$ steps. This generates nested edge-sets $E = E_0 \supseteq E_1 \supseteq \cdots \supseteq E_s$ wherein each E_i has degree at most $\Delta(\frac{1+\epsilon}{2})^i$ and has $|E_i| \approx 2^{-i}|E|$. The final set E_s has very small maximum degree, and a much simpler algorithm can then be used to select a large matching of it.

The process of generating nested edge sets $E_0, \ldots, E_s$ with decreasing maximum degree is superficially quite similar to our derandomization of direct rounding. Both algorithms generate nested edge sets which simulate the random process of retaining edges independently. But there are two key differences between the methodologies.

First, and less importantly, the algorithm of [12] aims to ensure that *all* the vertices have degree at most $\frac{\Delta}{2}(1+\epsilon)$. Since they use a random process based on a union bound over the vertices, this means they pay a factor of $\log n$ in the run-time. Second, in order to use degree-splitting over s stages with only a constant-factor overall loss, the algorithm of [12] needs $(1+\epsilon)^s = O(1)$, i.e., $\epsilon \approx 1/\log \Delta$. This is a very strict constraint: *every* vertex v must obey tight concentration bounds *at each of the stages* $i = 1, \ldots, s$.

These concentration bounds together give a complexity of $\tilde{O}(r \log n/\epsilon^2) = \tilde{O}(r \log n \log^2 \Delta)$ per stage, and total complexity of $\tilde{O}(r \log n \log^3 \Delta)$ over all stages.

Let us discuss how our algorithm avoids these issues. Like the algorithm of [9], we aim for a run-time independent of n. To this end, we do not insist that all vertices have their degree reduced. We discard the vertices for which certain bad-events occur, e.g. the degree is much larger than expected. These are rare so this does not lose too much in the weight of the matching.

Second, we observe that, if we actually ran the multi-stage process randomly, it would not be unusual for the degree of any given vertex v to deviate significantly from its mean value in a single stage. It would be quite rare to have such a deviation in all s stages. Thus, we really need to keep track of the *total* deviation of $\deg(v)$ from its mean value across *all the stages*, aggregated over all vertices v. This is precisely what we achive through our potential function analysis of direct rounding. By allowing more leeway for each vertex per stage, we can get away with looser concentration bounds, and correspondingly lower complexity.

E. Notation and conventions

For a graph $G = (V, E)$ and $v \in V$, we define $N[v]$ to be the inclusive neighborhood of vertex v, i.e. $\{v\} \cup \{w \mid (w, v) \in E\}$. For a hypergraph $H = (V, E)$ and $v \in V$, we define $N(v)$ to be the set of edges $e \ni v$. We define $\deg(v) = |N(v)|$ and for $L \subseteq E$ we define $\deg_L(v) = |N(v) \cap L|$. Unless stated otherwise, E may be a multi-set.

For a set X and integer k, we define $\binom{X}{k}$ to be the set of all k-element subsets of X, i.e. the collections of sets $Y \subseteq X, |Y| = k$.

For a graph $G = (V, E)$, we define the power graph G^t to be a graph with vertex set V, and with an edge (u, v) if there is a path of length up to t in G. Note that if G has maximum degree Δ, then G^t has maximum degree Δ^t.

We define a *fractional matching* to be a function $h : E \to [0, 1]$ such that $\sum_{e \in N(v)} h(e) \leq 1$ for every $v \in V$. Note that this should be distinguished from a fractional solution to the matching polytope of a graph, which also includes constraints for all odd cuts.

For any edge-weighting function $a : E \to [0, \infty)$, and any edge subset $L \subseteq E$, we define $a(L) = \sum_{e \in L} a(e)$. Similarly, for a fractional matching, we define $a(h) = \sum_{e \in E} h(e)a(e)$. Finally, for any hypergraph $H = (V, E)$ we write $a(H)$ as shorthand for $a(E)$.

For any function $a : E \to [0, \infty)$ we define a^* to be the maximum-weight fractional matching with respect to edge-weight function a.

For any boolean predicate $\mathcal{P}$, we use the Iverson notation so that $[[\mathcal{P}]]$ is the indicator function that $\mathcal{P}$ is true, i.e. $[[\mathcal{P}]] = 1$ if $\mathcal{P}$ is true and $[[\mathcal{P}]] = 0$ otherwise.

We use the $\tilde{O}()$ notation throughout, where we define $\tilde{O}(x)$ to be $x \operatorname{polylog}(x)$.

F. The LOCAL model for hypergraphs

Our algorithms are all based on the LOCAL model for distributed computations in a hypergraph. This is a close relative to Linial's classic LOCAL graph model [21], [29], and in fact the main motivation for studying hypergraph LOCAL algorithms is because they are useful subroutines for LOCAL graph algorithms. Let us first describe how the LOCAL model works for graphs.

There are two variants of the LOCAL model involving randomized or deterministic computations. In the deterministic variant, each vertex is provided with a unique ID which is a bitstring of length $\Theta(\log n)$; here the network size n is a global parameter passed to the algorithm. A vertex has a list of the ID's of its neighbors. Time proceeds synchronously: in any round a vertex can perform arbitrary computations and transmit messages of arbitrary sizes to its neighbors. At the end of this process, each vertex must make some decision as to some local graph problem. For example, if the graph problem is to compute a maximal independent set, then each vertex v sets a flag F_v indicating whether it has joined the MIS.

The closely related randomized model has each vertex maintain a private random string R_v of unlimited length. As before, time proceeds synchronously and all steps except the

generation of R_v can be regarded as deterministic. At the end of the process, we only require that the flags F_v correctly solve the graph problem w.h.p., i.e. with probability at least $1 - n^{-c}$ for any desired constant $c > 0$.

To define the LOCAL model on a hypergraph H, we first form the *incidence graph* $G = \text{Inc}(H)$; this is a bipartite graph in which each edge and vertex of H corresponds to a vertex of G. If u_v and u_e are the vertices in G corresponding to the vertex $v \in V$ and $e \in E$, then G has an edge (u_e, u_v) whenever $v \in e$. The LOCAL model for hypergraph H is simply the LOCAL graph model on G. In other words, in a single timestep on the hypergraph H, each vertex can send arbitrary messages to every edge $e \in N(v)$, and vice versa.

The connection between the hypergraph and graph LOCAL model goes both ways. For a number of LOCAL graph algorithms, we need structures such as maximal disjoint paths. The length-ℓ paths of a graph G can be represented as hyperedges in an auxiliary hypergraph H of rank ℓ. Furthermore, a communication step of H can be simulated in $O(\ell)$ rounds of the communication graph G. Most applications of HMM come from this use a subroutine for graph algorithms.

Many graph and hypergraph algorithms depend on the *maximum degree* Δ, *rank* r, and vertex count n. These parameters cannot actually be computed locally. As is standard in distributed algorithms, we consider Δ, r, n to be globally-known upper-bound parameters which are guaranteed to have the property that $|N(v)| \leq \Delta, |e| \leq r, |V| \leq n$ for every vertex v and edge e. When V is understood, we may say that E has maximum degree Δ and maximum rank r. We also assume throughout that $r \geq 2$, as the cases $r = 0$ and $r = 1$ are trivial. In many cases where hypergraph matching is used as a subroutine, we can deduce such upper bounds from the original, underlying graph problem. (In certain cases, LOCAL algorithms can be used without a priori bounds on Δ, r, n (see, for example [18]), but for simplicity we do not explore this here.)

In the case of weighted matchings, we let W denote the ratio between maximum and minimum edge weight. We note that our algorithm does *not* require that W is globally known, or is bounded in any way; the algorithm behavior is completely oblivious to this parameter.

II. Derandomization via proper vertex coloring

To begin our derandomization analysis, we describe a general method of converting randomized LOCAL algorithms into deterministic ones via proper vertex coloring. This is a slight variant of the method of [12] based on conditional expectations; we describe it here to set the notation and since some of the parameters are slightly different.

Consider a 1-round randomized graph algorithm A run on a graph $G = (V, E)$; at the end of this process, each vertex v has a real-valued flag F_v. In this randomized process, we let R_v denote the random bit-string chosen by vertex v; to simplify the analysis, we take R_v to be an integer selected uniformly from the range $\{0, \ldots, \gamma - 1\}$ for some finite value γ. We define $\vec{R}$ to be the overall collection of values R_v, and we also define $\mathfrak{R} = \{0, \ldots, \gamma - 1\}^n$ to be the space of all possible values for $\vec{R}$. In this setting, each F_v can be considered as a function mapping $\mathfrak{R}$ to $\mathbb{R}$. Since the algorithm A takes just 1 round, each value $F_v(\vec{R})$ is determined by the values R_w for $w \in N[v]$. (The extension to multi-round LOCAL algorithms will be immediate.)

Lemma II.1. *Suppose that G^2 has maximum degree d. Then there is a deterministic graph algorithm in $O(d + \log^* n)$ rounds to determine values $\vec{\rho}$ for the random bits $\vec{R}$, such that when (deterministically) running A with the values $\vec{R} = \vec{\rho}$, it satisfies*

$$\sum_v F_v(\vec{\rho}) \leq \mathbb{E}[\sum_v F_v(\vec{R})]$$

Furthermore, when given a $O(d)$ coloring of G^2 as input, the additive factor of $\log^* n$ can be omitted.

Proof: We set the bits R_v using the method of conditional expectations. See [12] for a full exposition; we provide just a sketch here.

If we are not already given a proper vertex coloring of G^2 with $O(d)$ colors, we use the algorithm of [10] to obtain this in $\tilde{O}(\sqrt{d}) + O(\log^* n)$ rounds. Next, we proceed sequentially through the color classes; at the i^{th} stage, every vertex v of color i selects a value ρ_v to ensure that the expectation of $F_v + \sum_{(u,v) \in E} F_u$, conditioned on $R_v = \rho_v$, does not increase. Note that all the vertices of color i are non-neighbors, so they do not interfere during this process. ∎

This setting, wherein the statistic of interest is $\sum_v F_v$, is more general than the setting of locally-checkable problems considered in [12]. The flag F_v here is not necessarily interpreted as an indication that the overall algorithm has failed with respect to v, and indeed it may not be possible for any individual vertex v to witness that the algorithm has failed.

Also, Lemma II.1 is stated in terms of *minimizing* the sum $\sum_v F_v(\vec{\rho})$; by replacing F with $-F$, we can also maximize the sum, i.e. get $\sum_v F_v(\vec{\rho}) \geq \mathbb{E}[\sum_v F_v(\vec{R})]$. In our applications, we will use whichever form (maximization or minimization) is most convenient; we do not explicitly convert between these two forms by negating the objective functions.

To derandomize hypergraph LOCAL algorithms, we apply Lemma II.1 to the incidence graph $G = \text{Inc}(H)$. It is convenient to rephrase Lemma II.1 in terms of H without explicit reference to G.

Definition II.2. *For a hypergraph H of rank r and maximum degree Δ, and incidence graph G, we define a* good coloring *of H to be a proper vertex coloring of G^2 with $O(r\Delta)$ colors.*

Lemma II.3. *Let A be a randomized 1-round algorithm run on a hypergraph $H = (V, E)$ of rank r and maximum degree Δ. Each $u \in V \cup E$ has private random bitstring R_u and at the termination of A it outputs the real-valued quantities F_u. If we are provided a good coloring of H, then there is a deterministic $O(r\Delta)$-round algorithm to determine values $\vec{\rho}$ such that when (deterministically) running A with the values $\vec{R} = \vec{\rho}$, it satisfies*

$$\sum_{u \in V \cup E} F_u(\vec{\rho}) \leq \mathbb{E}[\sum_{u \in V \cup E} F_u(\vec{R})]$$

Proof: Let G be the incidence graph of H. Note that G^2 has maximum degree $d = \Delta r$. A good coloring of H is by definition a $O(d)$ coloring of G^2. Now apply Lemma II.1 with respect to G. ∎

As a simple application of Lemma II.3, which we need later in our algorithm, we consider a straightforward rounding algorithm for matchings.

Lemma II.4. *Let $H = (V, E)$ be a hypergraph with an edge-weighting $a : E \to [0, \infty)$ and a good coloring of H. Then there is a deterministic $O(r\Delta)$-round algorithm to compute a matching M with $a(M) \geq \Omega(\frac{a(E)}{r\Delta})$.*

Proof: We provide a sketch here; the full proof is in Appendix A. Consider the following 1-round randomized algorithm: we generate edge-set $L \subseteq E$, wherein each edge $e \in E$ goes into L independently with probability $p = \frac{0.1}{r\Delta}$. Next, we form a matching M from L by discarding any pair of intersecting edges. Let us define the following statistic on L:

$$\Phi(L) := \sum_{e \in E} a(e)[[e \in L]] - \sum_{v \in V} \sum_{\substack{e, e' \in N(v) \\ e' \neq e}} a(e)[[e \in L \wedge e' \in L]]$$

One can easily see that $a(M) \geq \Phi(L)$ and that $\mathbb{E}[\Phi(L)] \geq \Omega(\frac{a(E)}{r\Delta})$. Furthermore, the statistic $\Phi(L)$ has the required form for Lemma II.3, where the flag F_u for $u \in V \cup E$ is defined as

$$F_e = [[e \in L]]a(e),$$
$$F_v = \sum_{\substack{e, e' \in N(v) \\ e \neq e'}} -[[e \in L \wedge e' \in L]]a(e)$$

Lemma II.3 thus gives an edge-set L with $\Phi(L) \geq \mathbb{E}[\Phi(L)]$. The resulting matching M satisfies $a(M) \geq \Omega(\frac{a(E)}{r\Delta})$ as desired. ∎

The method of derandomization via proper vertex coloring is not satisfactory on its own, because of its polynomial dependence on Δ. We remove this by adapting the degree-splitting algorithm of [12]. The following definitions are useful to characterize this process:

Definition II.5 (frugal and balanced edge colorings)**.** *For a hypergraph H and edge-coloring $\chi : E \to \{1, \ldots, k\}$, we say that the coloring χ is t-*frugal *if every vertex v has at most t edges of any given color j; more formally, if $|N(v) \cap \chi^{-1}(j)| \leq t$ for all $v \in V, j \in \{1, \ldots, k\}$.*

We say that an edge-coloring $\chi : E \to \{1, \ldots, k\}$ is balanced *if it is t-frugal for $t = O(\Delta/k)$.*

Note that a proper edge coloring would typically not be balanced, as it would have frugality $t = 1$ and use $k = r\Delta$ colors. The trivial randomized coloring algorithm gives a balanced edge coloring for $t = \log n$. The main contribution of [12] is to derandomize this, getting a balanced coloring with $t = \operatorname{polylog} n$. This dependence on n is not suitable for us; we want t to depend only on local parameters r, Δ.

A simple application of the iterated Lovász Local Lemma (LLL) shows that a balanced edge-coloring exists for $t = \log r$. Unfortunately, even the randomized LLL algorithms are too slow for us. We will settle for something slightly weaker: we get a *partial* frugal edge-coloring. This can be obtained by iterating the following simple 1-round randomized algorithm: first randomly 2-color the edges, and then discard all edges incident to a vertex with more than $\frac{\Delta}{2}(1+\epsilon)$ edges of a color. This serves as a "poor man's LLL".

We summarize our edge coloring result as follows. The proof is similar to the work [12], so we defer it to Appendix A.

Lemma II.6. *Suppose we are given a good coloring of hypergraph $H = (V, E)$ with rank r and maximum degree Δ. For any $\delta \in (0, 1)$ and integer parameter $k \geq 2$ satisfying $\Delta \geq Ck^2 \log \frac{r}{\delta}$ for some absolute constant C, there is an $\tilde{O}(r \log \frac{1}{\delta} \log^3 k)$-round deterministic algorithm to find an edge set $E' \subseteq E$ with an edge-coloring $\chi : E' \to \{1, \ldots, k\}$ such that*

1) *$a(E') \geq (1 - \delta)a(E)$*
2) *χ is t-frugal on E' for parameter $t = 4\Delta/k$.*

In particular the coloring χ is balanced.

A remark on maintaining good colorings. A typical strategy for our algorithms will be to first get a $\operatorname{poly}(r, \Delta)$ coloring of the original input hypergraph H, in $O(\log^* n)$ rounds, using the algorithm of [22]. Whenever we modify H (by splitting vertices, replicating edges, etc.) we will update the coloring. All these subsequent updates can be performed in $O(\log^*(r\Delta))$ rounds [22]. Whenever we need an $O(r\Delta')$ coloring of some transformed hypergraph H', we use the algorithm of [10] to obtain it in $o(r\Delta') + O(\log^*(r\Delta))$ rounds. These steps are all negligible compared to the runtime of the overall algorithm.

These coloring updates are routine but quite cumbersome to describe. For simplicity of exposition, we will mostly ignore them for the remainder of the paper.

III. Derandomization without proper vertex coloring

The use of a proper vertex coloring χ in the proof of Lemma II.1 is somewhat limiting. In this section, we describe how to relax this condition. To make this work, we require that the statistic $\Phi = \sum_u F_u$ we are optimizing is highly tailored to χ; this is quite different from Lemma II.1, in which the function F_u is almost arbitrary and is treated in a black-box way. The derandomization result we get this way may seem very abstract. We follow with an example showing how it applies to concentration bounds for sums of random variables.

Let us briefly explain the intuition. If χ is a non-proper vertex coloring and we try to use the conditional expectation method of Lemma II.1, we will incorrectly compute the contributions from adjacent vertices of the same color. The reason for this is that the randomness for one may interact with the randomness for the other one. To avoid these errors, we carefully construct the statistic Φ to be "multilinear" with respect to the coloring χ. This avoids the problematic non-linear interactions. (Linear interactions do not cause problems.)

Let us note that a similar type of derandomization strategy using a non-proper vertex coloring is used in [17]. These methods are somewhat similar, but the work of [17] is based on *ignoring* all interactions between vertices with the same color, followed by a postprocessing step to correct the resulting errors. There is a key difference in our approach to the error term. Our statistic Φ is an approximation to the true statistic of interest (incurring some error), but the algorithm still "correctly" handles the linear interactions and incurs no further error in derandomizing it.

A. The derandomization lemma

Consider some graph $G = (V, E)$ with some global statistic $\Phi : \mathfrak{R} \to \mathbb{R}$ which is a function of the random bits. Each vertex may have some additional, locally-held state; for example, we may be in the middle of a larger multi-round algorithm and so each node will have seen some information about its t-hop neighborhood. The function Φ may depend on this vertex state as well; to avoid burdening the notation, we do not explicitly write the dependence.

We need the statistic Φ to obey a certain multilinearity condition, i.e. it has no mixed derivatives of a certain kind. In order to define, we need to analyze the directional derivative structure of Φ.

In the applications in this paper, we will only need to analyze a relatively simple probabilistic process, wherein every vertex chooses a single bit uniformly at random. In the language of Section II, this is the setting with $\gamma = 2$. In order to simplify the definitions and exposition here, we will focus solely on this case; the results can easily be generalized to more complicated probability distributions.

Thus, we view Φ as a function mapping $\{0,1\}^V$ to $\mathbb{R}$. Formally, for any vertex $v \in V$ we define the *derivative* $D_v\Phi$ as follows:

$$(D_v\Phi)(x_1, \ldots, x_n) = \Phi(x_1, \ldots, x_{v-1}, \overline{x_v}, x_{v+1}, \ldots, x_n) - \Phi(x_1, \ldots, x_{v-1}, x_v, x_{v+1}, \ldots, x_n)$$

where here $\overline{x_v}$ denotes flipping the value of bit x_v. Note that $D_v\Phi$ is itself a function mapping $\mathfrak{R}$ to $\mathbb{R}$, so we can talk about its derivatives as well.

Definition III.1 (uncorrelated potential function)**.** *Function Φ is* pairwise uncorrelated *for vertices v, v' if the function $D_vD_{v'}\Phi$ is identically zero. We say that Φ is* uncorrelated *for vertices $v_1, \ldots, v_s$ if it is pairwise uncorrelated for every pair v_i, v_j and $i \neq j$.*

We are now ready to state our main derandomization lemma.

Lemma III.2. *Suppose that we are provided a vertex coloring $\chi : V \to \{1, \ldots, k\}$ (not necessarily proper) for G, where the potential function $\Phi : \mathfrak{R} \to \mathbb{R}$ has the following properties:*

- (A1) *For all vertices v, w with $(v, w) \notin E$, the function Φ is pairwise uncorrelated for v, w.*
- (A2) *For all vertices v, w with $v \neq w$ and $\chi(v) = \chi(w)$, the function Φ is pairwise uncorrelated for v, w.*
- (A3) *For any vertex v, the value of $D_v\Phi(x_1, \ldots, x_n)$ can be locally computed by v given the values of x_w for $w \in N[v]$.*

Then there is a deterministic $O(k)$-round algorithm to determine values $\vec{\rho}$ for the random bits, such that $\Phi(\vec{\rho}) \leq \mathbb{E}[\Phi(\vec{R})]$.

Proof: We provide a sketch here; see Appendix A for further details.

We proceed through stages $i = 1, \ldots, k$; at each stage i, every vertex v with $\chi(v) = i$ selects some value ρ_v so that $\mathbb{E}[D_v\Phi \mid R_v = \rho_v] \geq 0$, and permanently commits to $R_v = \rho_v$.

We first claim that this information can be computed by v in $O(1)$ rounds. By Property (A1), the conditional expectation $\mathbb{E}[D_v\Phi \mid R_v = x]$ only depends on the values of R_w for $w \in N(v)$. Thus in $O(1)$ rounds v can query the values of ρ_w for $w \in N(v)$. It can then integrate over the possible random values for all R_w in its neighborhood and use Property (A3) to determine $\mathbb{E}[D_v\Phi \mid R_v = x]$ for all values x and u. This allows it to determine ρ_v.

Finally, Property (A2) ensures that the decision made for each vertex v does not interfere with any other vertex v' of the same color. Therefore, the conditional expectation of Φ does not increase during the round i. ∎

Lemma II.1 is a special case of Lemma III.2: for, consider a 1-round LOCAL algorithm which computes for each vertex v a function F_v. If χ is a proper vertex coloring

of G^2, then the potential function $\Phi = \sum_{v \in V} F_v$ satisfies Lemma III.2 with respect to the graph G^2. (Condition (A2) is vacuous, as vertices v, w with $\chi(v) = \chi(w)$ must be non-neighbors and hence by (A1) they are pairwise uncorrelated.)

B. Example: derandomizing edge-splitting

For better motivation, we consider a simplified example, in which Φ corresponds to Chernoff-type bounds on the deviations of sums of random variables. There is a particularly powerful and natural way to apply Lemma III.2 in this setting. (In our application later in Section IV, we will encounter something similar to this, but much more complex.) To provide more intuition, we will not carry out any detailed error estimates.

Let us consider a Δ-regular hypergraph $H = (V, E)$ (i.e. every vertex v has $\deg(v) = \Delta$) and consider the random process where each edge $e \in E$ is added to a set L with probability $1/2$. Our statistic of interest is

$$\Phi(L) = \sum_v a(v) \big[\big[|\deg_L(v) - \tfrac{\Delta}{2}| \geq t\big]\big]$$

for some function $a : V \to \mathbb{R}$ and some desired parameter t. We would like to select some (deterministic) edge set L such that

$$\Phi(L) \leq \mathbb{E}[\Phi] = \sum_v a(v) \Pr\big(|\deg_L(v) - \tfrac{\Delta}{2}| \geq t\big)$$

We are going to apply Lemma III.2 to the graph $G = \mathrm{Inc}(H)^2$. We actually only care about the nodes of G corresponding to edges of H; the nodes of G corresponding to vertices are immaterial and will be ignored. Thus, the vertex-coloring of G that will be used for Lemma III.2 corresponds to an edge-coloring of H. Note that all the results and terminology in Lemma III.2 regarding vertex colors should be translated into edge colors for this application.

Accordingly, let us first choose a balanced k-edge coloring $\chi : E \to \{1, \ldots, k\}$ for k chosen appropriately. The choice of k and the role played by χ will become clear shortly. Next let us define the indicator variables $X_e = [[e \in L]]$, which are independent Bernoulli-$1/2$ random variables. At this point, we use a result of [31] based on a connection between Chernoff bounds and symmetric polynomials, which is based on the following inequality:

$$[[\deg_L(v) \geq \mu(1+\delta)]] \leq \frac{\binom{\deg_L(v)}{w}}{\binom{\mu(1+\delta)}{w}} = \frac{\sum_{W \in \binom{N(v)}{w}} \prod_{e \in W} X_e}{\binom{\mu}{w}}$$

for any integer $w \leq \mu(1+\delta)$, where $\mu = \frac{\Delta}{2}$ and $\delta = t/\mu$. This bound is powerful because we can then calculate

$$\Pr(\deg_L(v) \geq \mu(1+\delta)) \leq \frac{\sum_{W \in \binom{N(v)}{w}} \mathbb{E}[\prod_{e \in W} X_e]}{\binom{\mu(1+\delta)}{w}};$$

further calculations of [31] show that for $w = \lceil t \rceil$ the RHS is at most the well-known Chernoff bound $(\frac{e^\delta}{(1+\delta)^{1+\delta}})^\mu$. Smaller values of w also yield powerful concentration bounds, with probability bounds roughly inverse exponential in w.

The denominator here is a constant for all v, so it can be ignored. Thus, as a proxy for $\Phi(L)$, it would be natural to use a pessimistic estimator

$$\Phi'(L) = \sum_v a(v) \sum_{W \in \binom{N(v)}{w}} \Big(\prod_{e \in W} X_e + \prod_{e \in W} (1 - X_e) \Big)$$

Unfortunately, this function Φ' is not compatible with Lemma III.2. The problem is that some set $W \subseteq N(v)$ may contain multiple edges with the same color; for a pair of edges $e_1, e_2 \in N(v)$, a term such as $\prod_{e \in W} X_e + \prod_{e \in W} (1 - X_e)$ will have a non-vanishing second derivative $D_{e_1} D_{e_2} \prod_{e \in W} X_e$.

To avoid this, we need another statistic Φ''. For any vertex v, define U_v to be the set of all subsets $W \in \binom{N(v)}{w}$ such that all edges f in W have distinct colors. If we restrict the sum to only the subsets $W \in U_v$, then we get a statistic which is compatible with Lemma III.2:

$$\Phi''(L) = \sum_v a(v) \sum_{W \in U_v} \Big(\prod_{e \in W} X_e + \prod_{e \in W} (1 - X_e) \Big)$$

To show that this satisfies (A1), note that if e_1 and e_2 are not connected in G^2, then at most one of them involves vertex v. Thus, the set W contains at most one of the edges e_1, e_2 and so $D_{e_1} D_{e_2} \prod_{e \in W} X_e = 0$.

To show property (A2), suppose that $\chi(e_1) = \chi(e_2)$. Then any $W \in U_v$ contains at most one of e_1, e_2, and so the second derivative $D_{e_1} D_{e_2} (\prod_{e \in W} X_e + \prod_{e \in W} (1 - X_e)$ is indeed zero.

To show that this satisfies (A3), we compute the first derivative D_{e_1} as:

$$D_{e_1} \Phi' = \sum_v a(v) \sum_{W \in U_v : e_1 \in W} D_{e_1} \big(\prod_{e \in W} X_e + \prod_{e \in W} (1 - X_e) \big)$$

For such W, all the other edges $e \in W$ also involve vertex v, and so e is a neighbor to e_1 in G^2. So all such terms can be computed locally by e_1.

This means that Lemma III.2 applies to Φ'', and we get a subset L with $\Phi''(L) \geq \mathbb{E}[\Phi''(L)]$. But this is not what we want; we want $\Phi(L) \geq \mathbb{E}[\Phi(L)]$. This is where our derandomization incurs some small error.

We will not bound the error rigorously here. But, at a high level, let us explain why it should be small. Consider some randomly chosen subset $W \subseteq N(v)$ of size w. Since χ is b-frugal for $b = O(\Delta/k)$, the *expected* number of pairs $f_1, f_2 \in W$ with $\chi(f_1) = \chi(f_2)$ is at most $w^2 b/\Delta = O(w^2/k)$. As long as $k \gg w^2$, then, we expect the edges in W have different colors, and hence $W \in U_{v,e}$.

Thus, *most of the sets* $W \in \binom{N(v)}{w}$ *are already in* $U_{v,e}$. As a result, we will have

$$\sum_{W \in U_v} \prod_{e \in W} X_e \approx \sum_{W \in \binom{N(v)}{w}} \prod_{e \in W} X_e$$

and similarly for the terms $\prod(1 - X_e)$.

For this reason, $\Phi''(L)$ is quite close to $\Phi(L)$, and we thus ensure that

$$\Phi(L) \approx \Phi''(L) \geq \mathbb{E}[\Phi''(L)] \approx \mathbb{E}[\Phi(L)]$$

which is (up to some small error terms) precisely the derandomization we want.

This example illustrates three important caveats in using Lemma III.2. First, we may have some natural statistic to optimize in our randomized algorithm (e.g. the total weight of edges retained in a coloring, indicator functions for whether a bad-event has occurred). This statistic typically will not satisfy condition (A2) directly. Instead, we must carefully construct a pessimistic estimator which does satisfy condition (A2).

Second, to avoid non-linear interactions, we must allow some small error in our potential function. The potential function must be carefully constructed so that these errors remain controlled.

Finally, in order to apply Lemma III.2, we need an appropriate coloring. In our hypergraph matching application this will be a balanced frugal edge coloring χ. We first obtain this coloring using the derandomization Lemma II.6. We then craft the statistic for Lemma III.2 in terms of χ Thus, each application of Lemma III.2 is essentially a derandomization within a derandomization: we first use a relatively crude method to obtain the needed coloring, and then use this to obtain a more refined bound via Lemma III.2.

IV. Derandomization of direct rounding

We now use our derandomization methods to convert a fractional matching into an integral one. We show the following result:

Theorem IV.1. *Let $H = (V, E)$ be a hypergraph of maximum degree Δ and rank r, along with an edge-weighting function $a : E \to [0, \infty)$, and a good coloring of H. There is a $\tilde{O}(\log^2 \Delta + r \log \Delta)$-round deterministic algorithm to find a matching M with*

$$a(M) \geq \Omega\left(\frac{a(E)}{r\Delta}\right)$$

This value of $a(M)$ is precisely what we would obtain by applying Lemma II.4 to H. Unfortunately, Lemma II.4 would take $O(r\Delta)$ rounds, which is much too large.

So we need to reduce the degree of H. Consider the following random process: we select each edge $e \in E$ independently with probability $p = \Theta(\frac{\log r}{\Delta})$. If any vertex v has degree larger than $c \log r$ for some constant c, we discard all the selected edges. We let J denote the selected edges and let $J' \subseteq J$ denote the set of edges which are not discarded.

Clearly, the hypergraph (V, J') has maximum degree $O(\log r)$. It is not hard to see that $\mathbb{E}[\frac{a(J')}{\log r}] \geq \Omega(\frac{a(E)}{\Delta})$. (See Proposition A.5 for further details). We derandomize this to get $\frac{a(J')}{\log r} \geq \Omega(\frac{a(E)}{\Delta})$ (in actuality, not expectation). We can then follow up by applying Lemma II.4 to (V, J'), obtaining a matching M with $a(M) \geq \Omega(\frac{a(J')}{\log r}) \geq \Omega(\frac{a(E)}{\Delta})$ in $\tilde{O}(r)$ rounds.

To derandomize this process, we first slow down the randomness. Instead of selecting edge-set J in a single stage, we go through $\log_2(1/p)$ stages, in which each edge is retained with probability $1/2$. We use the method of conditional expectations to select a series of edge-sets $E_1, \ldots, E_s$. Ideally, we would choose $E_i = L$ such that $\mathbb{E}[a(J') \mid E_i = L] \geq \mathbb{E}[a(J')]$. In this conditional expectation, every edge e remaining in E_i will get selected for J independently with probability $2^i p$.

Unfortunately, the conditional expectation $\mathbb{E}[a(J') \mid E_i = L]$ is a complex, non-linear function of L, so we cannot directly select L to optimize it. Instead, we carefully construct a family of pessimistic estimator $S_0, \ldots, S_s$, which are amenable to our derandomization Lemma III.2 and also satisfies $\mathbb{E}[a(L') \mid E_i = L] \approx S_i(L)$.

For the remainder of this section, we assume that we are given a good coloring of H and edge-weight function a; these will not be mentioned specifically again in our hypotheses.

A. The formal construction

For an edge-set $L \subseteq E$ and integer $i \in \{0, \ldots, s\}$ we define the potential function

$$S_i(L) = (\tfrac{1}{2\alpha})^{s-i} a(L) - b_i \sum_{v \in V} \left(\binom{\Delta 2^{-i}}{w} + \binom{\deg_L(v)}{w} \right) a(N(v) \cap L)$$

where we define the parameters as follows:

$$\begin{aligned} w &= \lceil 2 \log_2(32 r^2 \log_2 \Delta) \rceil \\ x &= w^4 \log^{10}(r\Delta) \\ s &= \lceil \log_2 \tfrac{\Delta}{x} \rceil \\ \alpha &= 2^{1/s} \\ \beta &= 16r(2x/w)^w \\ b_i &= 2^{-(s-i)(w+1)} \alpha^{s-i} / \beta \end{aligned}$$

Our plan is to successively select edge subsets $E = E_0 \supseteq E_1 \supseteq \cdots \supseteq E_s$, such that $S_0(E_0) \leq S_1(E_1) \leq \cdots \leq S_s(E_s)$. Let us try to provide some high-level intuition for the different terms in this expression. Here, the probability of selecting an edge in the direct rounding would be $p = 2^{-s} \approx \frac{x}{\Delta}$. We will form J' by discarding vertices with degree larger than cx for some constant c. At stage i, the quantity $S_i(E_i)$

is supposed to represent the expectation of $a(J')$ after $s-i$ additional stages

The first term represents the expected weight of the edges remaining in J, wjhich is $a(E_i)2^{-(s+i)}$. We include an additional error term $(\frac{1}{\alpha})^{s-i}$ here, because some edges will need to be discarded when we obtain our frugal edge colorings.

The second term is supposed to represent the total weight of the edges *discarded* due to vertices with excessive degree. For a given vertex v and $L = E_i$, this expression is (up to scaling factors) given by:

$$\alpha^{s-i} \times 2^{-(s-i)w} \left(\tbinom{\Delta 2^{-i}}{w} + \tbinom{\deg_{E_i}(v)}{w} \right) \times 2^{-(s-i)} a(N(v) \cap E_i)$$

Let us see where these terms come from:

- The term α^{s-i} is an additional fudge factor, accounting for some small multiplicative errors in our approximations.
- The term $2^{-(s-i)}a(N(v) \cap E_i)$ is the expected value of $a(N(v) \cap J)$
- The quantity $2^{-(s-i)w}\binom{\deg_{E_i}(v)}{w}$ is roughly the expected value of $\binom{\deg_J(v)}{w}$, and thus (up to rescaling) an approximation to the probability that that $\deg_J(v) \gg x$.
- Finally, let us explain the term $\binom{\Delta 2^{-i}}{w}$. We only are worried about vertices whose degree is much higher than the expected value $\Delta 2^{-i}$. If $\deg_{E_i}(v) \ll \Delta 2^{-i}$, then the term $\binom{\deg_{E_i}(v)}{w}$ will negligible compared to $\binom{\Delta 2^{-i}}{w}$ and so we can essentially ignore the vertex v.

We note that a similar technique of developing a pessimistic estimator for a polynomial potential function, and derandomizing it via multiple rounds of conditional expectations was used in the algorithm of [15] for hypergraph MIS. In particular, they developed the technical tool of using an additive term to "smooth" errors in the multiplicative terms of concentration inequalities, which we adopt here.

The key technical result for the algorithm will be the following:

Lemma IV.2. *Suppose $\Delta \geq \Delta_0$ for some universal constant Δ_0. Given an edge-set $L \subseteq E$, there is a deterministic $\tilde{O}(r + \log \Delta)$-round algorithm to find an edge set $L' \subseteq L$ such that $S_{i+1}(L') \geq S_i(L)$ and L' has maximum degree $\Delta 2^{4-i}$.*

This lemma is quite technically involved, and the proof will involve a number of subclaims. We show it next in Section IV-B. Before we do so, let us show some straightforward properties of the potential function, and also show how Theorem IV.1 follows from the lemma. At several places, we will use the elementary inequalities

$$p^w \tbinom{T}{w} \geq \tbinom{pT}{w} \qquad \forall p \in [0,1], T \in \mathbb{Z}_+ \tag{1}$$

and

$$\frac{x}{4\Delta} \leq 2^{-s} \leq \frac{x}{2\Delta} \tag{2}$$

Proposition IV.3. *We have $S_0(E) \geq \Omega(\frac{a(E)x}{\Delta})$.*

Proof: As $\alpha^s = 2$ and using the inequality Eq. (2), at $i = 0$ we have

$$(\tfrac{1}{2\alpha})^{s-i} a(E) \geq 2^{-s-1} a(E) \geq \frac{a(E)x}{4\Delta}$$

Next, let us consider some vertex v, and we want to estimate the contribution of the term $b_i(\binom{\Delta 2^{-i}}{w} + \binom{\deg_E(v)}{w})a(N(v) \cap E)$. First, note that $b_0 = 2^{-s(w+1)}\alpha^s/\beta \leq \frac{1}{\beta} \times (\frac{x}{2\Delta})^{w+1} \times 2$. By definition of Δ, we have $\binom{\deg_E(v)}{w} \leq \binom{\Delta}{w}$. So

$$\begin{aligned} b_i(\tbinom{\Delta 2^{-i}}{w} + \tbinom{\deg_E(v)}{w}) &\leq \tfrac{2}{\beta}(\tfrac{x}{2\Delta})^{w+1}(\tbinom{\Delta}{w} + \tbinom{\Delta}{w}) \\ &= \tfrac{4}{\beta}(\tfrac{x}{2\Delta})^{w+1}\tbinom{\Delta}{w} \\ &\leq \tfrac{4}{\beta}(\tfrac{x}{2\Delta})^{w+1}(\tfrac{e\Delta}{w})^w \\ &\qquad \text{using the inequality } \tbinom{A}{B} \leq (\tfrac{eA}{B})^B \\ &= \frac{4x}{2\beta\Delta} \times (\frac{ex}{2w})^w \leq \frac{2x}{\beta\Delta}(2x/w)^w \end{aligned}$$

Substituting in these values, we get

$$\begin{aligned} & b_i \sum_{v \in V} (\tbinom{\Delta 2^{-i}}{w} + \tbinom{\deg_E(v)}{w}) a(N(v) \cap E) \\ &\leq \frac{2x}{\beta\Delta}(2x/w)^w \sum_{v \in V} a(N(v) \cap E) \\ &\leq \frac{2rx(2x/w)^w}{\beta\Delta} a(E) \qquad \text{since } H \text{ has rank at most } r \\ &= \frac{x}{8\Delta} \qquad \text{substituting the value of parameter } \beta \end{aligned}$$

Thus $S_0(E) \geq \frac{a(E)x}{\Delta}(\frac{1}{4} - \frac{1}{8}) \geq \Omega(\frac{a(E)x}{\Delta})$. ∎

Proposition IV.4. *Given a hypergraph $H = (V, E)$ of maximum degree Δ with a good coloring, there is a $\tilde{O}(r \log \Delta + \log^2 \Delta)$-round deterministic algorithm to find a subset $E' \subseteq E$ such that $H' = (V, E')$ has maximum degree $\Delta' \leq \text{polylog}(\Delta, r)$ and such that $a(E')/\Delta' \geq \Omega(a(E)/\Delta)$*

Proof: We may assume that Δ is larger than any needed constant, as otherwise taking $E' = E, \Delta' = \Delta$ works. Let $E_0 = E$; we will proceed through s applications of Lemma IV.2. At the i^{th} stage, we apply Lemma IV.2 to E_{i-1} to generate subset $E_i \subseteq E_{i-1}$ with $S_i(E_i) \geq S_{i-1}(E_{i-1})$. We return the final set $E' = E_s$. Each such application uses $\tilde{O}(r + \log \Delta)$ rounds. There are $s = O(\log \Delta)$ rounds altogether, giving the stated complexity.

In the initial hypergraph, Proposition IV.3 shows that $S_0(E_0) = S_0(E) \geq \Omega(a(E)x/\Delta)$. Because of the guarantee of Lemma IV.2, we have $S_s(E_s) \geq S_{s-1}(E_{s-1}) \geq \cdots \geq S_0(E_0)$. Finally, let us observe that

$$\begin{aligned} S_s(E_s) &= a(E_s) - b_s \sum_{v \in V} (\tbinom{\Delta 2^{-s}}{w} + \tbinom{\deg_{E_s}(v)}{w}) a(N(v) \cap E_s) \\ &\leq a(E_s). \end{aligned}$$

Putting these inequalities together, we therefore have

$$a(E_s) \geq \Omega(\frac{a(E)x}{\Delta})$$

On the other hand, Lemma IV.2 ensures that each E_i has maximum degree $\Delta 2^{4-i}$, and in particular E_s has maximum degree $\Delta' = \Delta 2^{4-s} = O(x) = \text{polylog}(\Delta, r)$. Furthermore, we have

$$\frac{a(E_s)}{\Delta'} \geq \frac{\Omega(a(E)x/\Delta)}{O(x)} \geq \Omega(\frac{a(E)}{\Delta})$$

■

To prove Theorem IV.1, we reduce the degree of H from Δ to $\text{polyloglog}(\Delta)$ by two applications of Proposition IV.4.

Proof of Theorem IV.1: Given the initial hypergraph $H = (V, E)$ of maximum degree Δ, we apply Proposition IV.4 to obtain hypergraph $H' = (V, E')$ of maximum degree $\Delta' = \text{polylog}(r, \Delta)$ and such that $a(E')/\Delta' \geq \Omega(a(E)/\Delta)$. Next, apply Proposition IV.4 to hypergraph H', obtaining a hypergraph $H'' = (V, E'')$ of maximum degree $\Delta'' = \text{polylog}(r, \Delta') = \text{poly}(\log r, \log\log \Delta)$ and such that $a(E'')/\Delta'' \geq \Omega(a(E')/\Delta') \geq \Omega(a(E)/\Delta)$.

Finally, apply Proposition II.4 to hypergraph H'', obtaining a matching $M \subseteq E''$ such that $a(M) \geq \Omega(\frac{a(E'')}{r\Delta''}) \geq \Omega(\frac{a(E)}{r\Delta})$.

The first application of Proposition IV.4 takes $\tilde{O}(\log^2 \Delta + r \log \Delta)$ rounds. The second application takes $\tilde{O}(\log^2 \Delta' + r \log \Delta')$ rounds. The final application of Proposition II.4 takes $\tilde{O}(r\Delta'')$ rounds. Noting that $\Delta'' = \text{poly}(\log r, \log\log \Delta)$ and $\Delta' = \text{poly}(\log r, \log \Delta)$, the overall complexity is $\tilde{O}(\log^2 \Delta + r \log \Delta)$. ■

B. Proof of Lemma IV.2

Suppose now we are given a set $L \subseteq E$ and some index $i \in \{0, \ldots, s-1\}$. Our goal is to find a subset $L' \subseteq L$ with $S_{i+1}(L') \geq S_i(L)$.

We assume throughout that Δ is larger than some sufficiently large constant Δ_0. At a number of places in our analysis, we use without further comment certain inequalities which hold only for Δ larger than (unspecified) constants.

Consider the random process wherein each edge $e \in L$ goes into L' independently with probability $1/2$. One can check easily that $\mathbb{E}[S_{i+1}(L')] \geq S_i(L)$ in this case. Thus, a natural strategy would be to apply Lemma III.2 to derandomize the statistic $S_{i+1}(L')$. Unfortunately, the potential function S_{i+1} is not directly amenable to Lemma III.2. We will develop an approximating statistic S', which is close enough to S_{i+1}, yet satisfies the properties (A1), (A2), (A3). The derandomization method of Lemma III.2 also depends on an appropriate vertex coloring of G, which in this case corresponds to a frugal edge-coloring of H.

We begin with two key preprocessing steps. First, we discard from L all edges incident to vertices whose degree in L exceeds $\Delta 2^{4-i}$. We let L_0 be the remaining edges, so that L_0 has maximum degree $\Delta 2^{4-i}$. Next, we use Lemma II.6 to obtain an edge-set $L_1 \subseteq L_0$ along with an appropriate balanced edge-coloring of L_1.

Once L_1 and χ are fixed, we next develop the statistic S' to be amenable to applying Lemma III.2. For any vertex $v \in V$ and edge $e \in L_1$ let us define $U_{v,e}(L_1) \subseteq \binom{N(v)\cap L_1}{w}$ to be the set of all w-element subsets $W = \{f_1, \ldots, f_w\} \subseteq N(v) \cap L_1$ with the property that all the values $\chi(e), \chi(f_1), \ldots, \chi(f_w)$ are distinct. We likewise define $U_{v,e}(L') \subseteq \binom{N(v)\cap L'}{w}$ to be the set of such subsets W where $f_1, \ldots, f_w$ are also in L'. With this notation, we define the statistic S' (which is a function of the set L') as:

$$S' = (\tfrac{1}{2\alpha})^{s-(i+1)} a(L') - \alpha b_{i+1} \sum_{\substack{v\in V \\ e\in N(v)\cap L'}} \Big(|U_{v,e}(L')| + \tbinom{\Delta 2^{-(i+1)}}{w}\Big) a(e)$$

Our final step is to apply the derandomization Lemma III.2 with respect to statistic S' on the graph $G = (\text{Inc(H)})^2$. This gives an edge-set $L' \subseteq L_1$ with $S' \geq \mathbb{E}[S']$; here the expectation is taken over the random process wherein edges of L_1 go into L' independently with probability $\frac{1}{2}$.

In order to show that L' has the desired properties, we will show the following chain of inequalities:

$$S_{i+1}(L') \geq S' \geq \mathbb{E}[S'] \geq S_i(L_0) \geq S_i(L)$$

Let us begin by analyzing the first step wherein the set L_0 is formed from L.

Proposition IV.5. *We have $S_i(L_0) \geq S_i(L)$.*

Proof: Letting $U = \{v \in V \mid \deg_L(v) \geq \Delta 2^{4-i}\}$, we have $L_0 = L - \bigcup_{v\in U} N(v) \cap L$. There are three main terms in the difference $S_i(L_0) - S_i(L)$:

$$\begin{aligned} S_i(L_0) - S_i(L) &= (\tfrac{1}{2\alpha})^{s-i}(a(L_0) - a(L)) \\ &+ b_i \sum_{v\in V-U} \big(\tbinom{\Delta 2^{-i}}{w} + \tbinom{\deg_L(v)}{w}\big) a(N(v)\cap L) \\ &\quad - \big(\tbinom{\Delta 2^{-i}}{w} + \tbinom{\deg_{L_0}(v)}{w}\big) a(N(v)\cap L_0) \\ &+ b_i \sum_{v\in U} \big(\tbinom{\Delta 2^{-i}}{w} + \tbinom{\deg_L(v)}{w}\big) a(N(v)\cap L) \\ &\quad - \big(\tbinom{\Delta 2^{-i}}{w} + \tbinom{\deg_{L_0}(v)}{w}\big) a(N(v)\cap L_0) \end{aligned}$$

Let us estimate these in turn. First, we have

$$\begin{aligned} a(L_0) - a(L) &= -a(L - L_0) = -a(\bigcup_{v\in U} N(v)\cap L) \\ &\geq -\sum_{v\in U} a(N(v)\cap L) \end{aligned}$$

Next, for $v \in V - U$, we have $\deg_L(v) \geq \deg_{L_0}(v)$ and $a(N(v)\cap L) \geq a(N(v)\cap L_0)$, so

$$\big(\tbinom{\Delta 2^{-i}}{w} + \tbinom{\deg_L(v)}{w}\big) a(N(v)\cap L) - \big(\tbinom{\Delta 2^{-i}}{w} + \tbinom{\deg_{L_0}(v)}{w}\big) a(N(v)\cap L_0) \geq 0$$

Finally, for $v \in U$, we have $N(v) \cap L_0 = \emptyset$ and $\deg_L(v) \geq \Delta 2^{4-i}$, so that

$$\begin{aligned}\big(\tbinom{\Delta 2^{-i}}{w} + \tbinom{\deg_L(v)}{w}\big)a(N(v)\cap L) \\ - \big(\tbinom{\Delta 2^{-i}}{w} + \tbinom{\deg_{L_0}(v)}{w}\big)a(N(v)\cap L_0) \\ \geq \tbinom{\Delta 2^{4-i}}{w} a(N(v)\cap L)\end{aligned}$$

Putting these three terms together, we have shown that

$$\begin{aligned}S_i(L_0) - S_i(L) \geq (\tfrac{1}{2\alpha})^{s-i}(-\sum_{v\in U} a(N(v)\cap L)) \\ + b_i \sum_{v\in V-U} 0 \\ + b_i \sum_{v\in U} \tbinom{\Delta 2^{4-i}}{w} a(N(v)\cap L) \\ = \sum_{v\in U}\Big(-(\tfrac{1}{2\alpha})^{s-i} + b_i\tbinom{\Delta 2^{4-i}}{w}\Big)a(N(v)\cap L)\end{aligned}$$

In order to show that the sum is non-negative, we will show that

$$b_i\tbinom{\Delta^{4-i}}{w} \geq (\tfrac{1}{2\alpha})^{s-i} \tag{3}$$

Substituting the value of b_i, we need to show that

$$2^{-(s-i)} \times 2^{-(s-i)w}\alpha^{s-i}\tbinom{\Delta^{4-i}}{w} \geq \beta(\tfrac{1}{2\alpha})^{s-i} \tag{4}$$

Using the bound Eq. (1), we have

$$2^{-(s-i)w}\tbinom{\Delta 2^{4-i}}{w} \geq \tbinom{\Delta 2^{4-i}\times 2^{-(s-i)}}{w} = \tbinom{2^{4-s}\Delta}{w} \geq \tbinom{4x}{w}$$

Since $\alpha \geq 1$, in order to show the bound Eq. (4) it suffices to show that $\binom{4x}{w} \geq \beta$. Using the inequality $\binom{A}{B} \geq (\frac{A}{B})^B$ we have $\binom{4x}{w} \geq (4x/w)^w$. So, substituting in the value of β, it suffices to show that

$$(4x/w)^w \geq 16r(2x/w)^w$$

or, equivalently, $2^w \geq 16r$; this clearly holds due to our choice of w. ∎

We next analyze the second preprocessing step, which is a straightforward application of Lemma II.6.

Proposition IV.6. *The set $L_1 \subseteq L_0$ can be generated in $\tilde{O}(r\,\mathrm{polyloglog}(\Delta))$ rounds such that $a(L_1) \geq a(L_0)/\alpha$ and edge-coloring $\chi : L_1 \to \{1,\ldots,k\}$ is t-frugal, where we define the parameters*

$$k = \lceil 2048w^2 \log\Delta\rceil, \qquad t = 2^{6-i}\Delta/k.$$

Proof: Edge-set L_0 has maximum degree $\Delta' = \Delta 2^{4-i}$. We will apply Lemma II.6 with parameters k and $\delta = 1 - 1/\alpha$ and the maximum degree bound Δ' to obtain an edge set $L_1 \subseteq L_0$ with $a(L_1) \geq (1-\delta)a(L_0)$ and an edge-coloring $\chi : L_1 \to \{1,\ldots,k\}$ which has frugality $4\Delta'/k = t$.

Let us check that hypotheses of Lemma II.6 are satisfied. With our parameter Δ', we need to check that $\Delta 2^{4-i} \geq Ck^2\log(r/\delta)$. By Eq. (2) we have $\Delta 2^{4-i} \geq \Delta 2^{4-s} \geq 4x$. Also, $\frac{1}{\delta} = \frac{\alpha}{\alpha-1} \leq \frac{1}{2^{1/s}-1} = O(s) = O(\log\Delta)$. Thus, it suffices to show that $x \geq C'k^2\log(r\Delta)$ for some constant C'. Since $k = O(w^2\log\Delta)$ and $x = w^4\log^{10}(r\Delta)$, this indeed holds for Δ sufficiently large.

Lemma II.6 runs in $\tilde{O}(r\log\frac{1}{\delta}\log^3 k)$ rounds. As $1/\delta = O(\log\Delta)$ and $k = O(w^2\log\Delta) = \tilde{O}(\log\Delta\log r)$, this is overall $\tilde{O}(r\,\mathrm{polyloglog}(\Delta))$. ∎

After the two preprocessing steps, we come to the heart of the construction: applying Lemma III.2. We break this down into a number of smaller claims.

Proposition IV.7. *We have $S_{i+1}(L') \geq S'$.*

Proof: We compute the difference:

$$\begin{aligned}S_{i+1}(L') - S' = \\ - b_{i+1}\sum_{v\in V}\big(\tbinom{\deg_{L'}(v)}{w} + \tbinom{\Delta 2^{-(i+1)}}{w}\big)a(N(v)\cap L') \\ + \alpha b_{i+1}\sum_{v\in V}\sum_{e\in N(v)\cap L'}(|U_{v,e}(L')| + \tbinom{\Delta 2^{-(i+1)}}{w})a(e) \\ = b_{i+1}\sum_{v\in V}\sum_{e\in N(v)\cap L'} a(e)\Big(\alpha|U_{v,e}(L')| \\ + (\alpha-1)\tbinom{\Delta 2^{-(i+1)}}{w} - \tbinom{\deg_{L'}(v)}{w}\Big)\end{aligned}$$

To show this is non-negative, we claim that for any vertex v and edge $e \in N(v)\cap L'$ we have

$$\alpha|U_{v,e}(L')| + (\alpha-1)\tbinom{\Delta 2^{-(i+1)}}{w} \geq \tbinom{\deg_{L'}(v)}{w} \tag{5}$$

There are two cases to show Eq. (5).

Case I: $\deg_{L'}(v) \leq \Delta 2^{-(i+2)}$**.** Then it suffices to show that

$$(\alpha-1)\tbinom{\Delta 2^{-(i+1)}}{w} \geq \tbinom{\Delta 2^{-(i+2)}}{w}$$

Since $\frac{\binom{\Delta 2^{-(i+1)}}{w}}{\binom{\Delta 2^{-(i+2)}}{w}} \geq (\frac{\Delta 2^{-(i+1)}}{\Delta 2^{-(i+2)}})^w = 2^w$ and $(\alpha-1) = 2^{1/s}-1 \geq \frac{1}{2s}$, it suffices to show that $2^w \geq 2s$. This holds because $2s \leq 2\log_2\Delta$ and $w \geq 2\log_2\log_2\Delta$.

Case II: $\deg_{L'}(v) > \Delta 2^{-(i+2)}$**.** Let us define $y = \deg_{L'}(v)$. Then, in order to show Eq. (5), it suffices to show that

$$\alpha|U_{v,e}(L')| \geq \tbinom{y}{w}$$

Note here that we have

$$\begin{aligned}\frac{wt}{y} \leq \frac{4w(\Delta 2^{6-i})/k}{\Delta 2^{-(i+2)}} = \frac{256w}{k} \\ \leq \frac{256w}{2048w^2\log\Delta} = \frac{1}{8w\log\Delta}\end{aligned}$$

In particular, $y \geq wt$. We now claim that we have the bound:

$$|U_{v,e}(L')| \geq (y - tw)^w/w! \tag{6}$$

To show Eq. (6), note that we can construct a set $W \in U_{v,e}(L')$ as follows. First, select some edge f_1 in $N(v)\cap L'$ with a different color than e. Since there are at most t edges

with the same color as e, there are at least $y-t$ such choices. Next, select edge $f_2 \in N(v) \cap L'$ with a different color than e or f_1. Again, there are at least $y - 2t$ such choices. Continue this process to choose edges $f_3, \ldots, f_w$. Since $y \geq tw$, we will always have at least $y - tw \geq 0$ choices for the edge f_j in this process. Now observe that any set of edges $W = \{f_1, \ldots, f_w\}$ is counted $w!$ times in this enumeration process.

As $\binom{y}{w} \leq y^w/w!$, the bound Eq. (6) and our bound on wt/y show that:

$$\frac{|U_{v,e}(L')|}{\binom{y}{w}} \geq \frac{(y-tw)^w/w!}{y^w/w!} = (1 - \frac{wt}{y})^w \geq (1 - \frac{1}{8w\log\Delta})^w$$

For Δ sufficiently large, we have $1 - \frac{1}{8w\log\Delta} \geq e^{-\frac{1}{4w\log\Delta}}$, so that

$$\frac{\alpha|U_{v,e}(L')|}{\binom{y}{w}} \geq \alpha(e^{-\frac{1}{4w\log\Delta}})^w = 2^{1/s} \times e^{-\frac{1}{4\log\Delta}} = e^{\frac{\log 2}{s} - \frac{1}{4\log\Delta}}$$

As $s \leq \log_2 \Delta$, this shows that $\frac{\alpha|U_{v,e}(L')|}{\binom{y}{w}} \geq 1$, as desired. ∎

Proposition IV.8. *Consider the random process wherein each edge $e \in L_1$ goes into L' independently with probability $\frac{1}{2}$. Then the expected value $\mathbb{E}[S']$ satisfies $\mathbb{E}[S'] \geq S_i(L_0)$.*

Proof: For a given edge e and vertex v, let $u_{v,e} = \mathbb{E}\big[|U_{v,e}(L')| \mid e \in L'\big]$. We compute:

$$\mathbb{E}[S'] = (\tfrac{1}{2\alpha})^{s-(i+1)}\,\mathbb{E}[a(L')] - \alpha b_{i+1} \sum_{\substack{v\in V\\ e\in N(v)\cap L_1}} \Pr(e \in L')a(e)\big(u_{v,e} + \tbinom{\Delta 2^{-(i+1)}}{w}\big)$$

Clearly $\mathbb{E}[a(L')] = \frac{a(L_1)}{2} \geq \frac{a(L_0)}{2\alpha}$, so $(\frac{1}{2\alpha})^{s-(i+1)}\,\mathbb{E}[a(L')] \geq (\frac{1}{2\alpha})^{s-i} a(L_0)$.

Next consider some vertex v and edge $e \in N(v) \cap L_1$. The edge e goes into L' with probability $1/2$, and also each set $W \in U_{v,e}(L_1)$ survives to $U_{v,e}(L')$ with probability exactly 2^{-w} conditional on e going into L', since $e \notin W$. So

$$u_{v,e} = \mathbb{E}\big[|U_{v,e}(L')| \mid e \in L'\big] = |U_{v,e}(L_1)|2^{-w} \quad (7)$$

Here $U_{v,e}(L_1) \subseteq \binom{N(v)\cap L_0}{w}$, so $|U_{v,e}(L_1)| \leq \binom{\deg_{L_0}(v)}{w}$. Also, by inequality Eq. (1), we have $\binom{\Delta 2^{-(i+1)}}{w} \leq 2^{-w}\binom{\Delta 2^{-i}}{w}$. With this inequality and Eq. (7), we have

$$u_{v,e} + \tbinom{\Delta 2^{-(i+1)}}{w} \leq 2^{-w}\Big(\tbinom{\deg_{L_0}(v)}{w} + \tbinom{\Delta 2^{-i}}{w}\Big)$$

Putting these contributions together, we see:

$$\begin{aligned}\mathbb{E}[S'] &\geq (\tfrac{1}{2\alpha})^{s-i}a(L_0) - \alpha b_{i+1} \sum_{\substack{v\in V\\ e\in N(v)\cap L_1}} 2^{-w-1}\Big(\tbinom{\deg_{L_0}(v)}{w} + \tbinom{\Delta 2^{-i}}{w}\Big)a(e)\\ &= (\tfrac{1}{2\alpha})^{s-i}a(L_0) - \alpha 2^{-w-1} b_{i+1} \sum_{v\in V}\Big(\tbinom{\deg_{L_0}(v)}{w} + \tbinom{\Delta 2^{-i}}{w}\Big)a(N(v)\cap L_1)\end{aligned}$$

Substituting the values of the parameters, we can calculate $\alpha 2^{-w-1} b_{i+1} = b_i$. Also note that $a(N(v)\cap L_1) \leq a(N(v)\cap L_0)$, so we have the lower bound:

$$\begin{aligned}\mathbb{E}[S'] &\geq (\tfrac{1}{2\alpha})^{s-i}a(L_0) - b_i \sum_{v\in V}\Big(\tbinom{\deg_{L_0}(v)}{w} + \tbinom{\Delta 2^{-i}}{w}\Big)a(N(v)\cap L_0)\\ &= S_i(L_0)\end{aligned}$$

∎

Lemma IV.9. *The derandomization algorithm of Lemma III.2 can generate a set $L' \subseteq L_1$ with $S' \geq \mathbb{E}[S']$ in $O(k)$ rounds.*

Proof: We will apply Lemma III.2 to the potential function S' with respect to the graph G and the coloring χ, where $G = \mathrm{Inc}(H, L_1)^2$. Recall that the random process we are derandomizing is that each edge of L_1 goes into L' independently with probability $1/2$. More concretely, let us say that each edge $e \in L'$ chooses a 1-bit random quantity X_e, and goes into L' if $X_e = 1$. This is precisely the randomization scenario considered in Lemma III.2.

We need to show that the potential function S' satisfies criteria (A1), (A2), (A3). Let us first write the value S' as a polynomial involving the X_e variables. Note that we have

$$|U_{v,e}(L')| = \sum_{W\in U_{v,e}(L_1)} \prod_{f\in W} X_f$$

We write $S'(L')$ in terms of monomial functions as:

$$S' = (\tfrac{1}{2\alpha})^{s-(i+1)} \sum_{e\in L_1} a(e)X_e - \alpha b_{i+1} \sum_{\substack{v\in V\\ e\in N(v)\cap L_1}} a(e)X_e\big(\tbinom{\Delta 2^{-(i+1)}}{w} + \sum_{W\in U_{v,e}(L_1)} \prod_{f\in W} X_f\big)$$

We want to compute the derivative $D_g S'(L')$ for some edge $g \in L_1$. The linearity of the differentiation operator

shows that:

$$D_g S' = (\tfrac{1}{2\alpha})^{s-(i+1)} \sum_{e \in L_1} a(e) D_g X_e$$
$$- \alpha b_{i+1} \sum_{\substack{v \in V \\ e \in N(v) \cap L_1}} a(e) \tbinom{\Delta 2^{-(i+1)}}{w} D_g X_e$$
$$- \alpha b_{i+1} \sum_{\substack{v \in V \\ e \in N(v) \cap L}} \sum_{W \in U_{v,e}(L_1)} a(e) D_g (X_e \prod_{f \in W} X_f)$$

Note that $D_g X_e = 0$ for $g \neq e$ while $D_g X_g = 1$, so this simplifies as:

$$D_g S' = (\tfrac{1}{2\alpha})^{s-(i+1)} a(g)$$
$$- \alpha b_{i+1} \sum_{v \in V : g \in N(v)} a(g) \big(\tbinom{\Delta 2^{-(i+1)}}{w} + \sum_{W \in U_{v,g}(L_1)} \prod_{f \in W} X_f \big)$$
$$- \alpha b_{i+1} \sum_{v \in V} \sum_{\substack{e \in N(v) \cap L \\ e \neq g}} a(e) \sum_{\substack{W \in U_{v,e}(L_1) \\ g \in W}} \prod_{f \in W - \{g\}} X_f$$

To show (A3), note that for edge g, this quantity only depends on the values X_f for edges f such that $f \in U_{v,g}(L_1)$ or $f \in U_{v,e}(L_1), g \in U_{v,e}(L_1)$. In either case, the edges f, g both contain vertex v. So the nodes corresponding to f, g are adjacent in the graph G^2 and hence g can locally compute the value $D_g S'$.

Now consider some edge $h \neq g$. Again by linearity and $D_h X_g = 0$, we compute the second derivative as:

$$D_h D_g S' = -\alpha b_{i+1} \sum_{v \in V : g \in N(v)} a(g) \sum_{W \in U_{v,g}(L_1)} D_h \prod_{f \in W} X_f$$
$$- \alpha b_{i+1} \sum_{v \in V} \sum_{\substack{e \in N(v) \cap L \\ e \neq g}} a(e) \sum_{\substack{W \in U_{v,e}(L_1) \\ g \in W}} D_h \prod_{f \in W - \{g\}} X_f$$

To show (A1), suppose that h is not connected to g in the graph G, i.e. $h \cap g = \emptyset$. Note that in the product, $\prod_{f \in W} X_f$, all the edges f are in W, in particular, $f \cap g \neq \emptyset$. Thus, $f \neq h$. Therefore, all the derivatives $D_h \prod_f X_f$ are equal to zero. This implies that $D_h D_g S' = 0$.

To show (A2), suppose that $\chi(h) = \chi(g)$. By definition of $U_{v,g}$, all the edges $f \in W \in U_{v,g}$ must have $\chi(f) \neq \chi(g)$. So $f \neq h$ for all such edges f. Similarly, by definition of $U_{v,e}$, all edges $f \in W - \{g\}$ have $\chi(f) \neq \chi(g)$, and again this implies that $f \neq h$. Consequently, we have $D_h \prod_{f \in W} X_f = 0$ for $W \in U_{v,g}$ and $D_h \prod_{f \in W - \{g\}} X_f = 0$ for $W \in U_{v,e}$. So $D_h D_g S' = 0$. ∎

At this point, we have all the pieces to prove Lemma IV.2:

Proof of Lemma IV.2: By Proposition IV.5 we have $S_i(L_0) \geq S_i(L)$. By Proposition IV.8 we have $\mathbb{E}[S'] \geq S_i(L_0)$, for the random process wherein each edge of L_1 goes into L_0 independently with probability $1/2$. By Proposition IV.9 we have $S' \geq \mathbb{E}[S']$. By Proposition IV.7 we have $S_{i+1}(L') \geq S'$. Thus $S_{i+1}(L') \geq S_i(L)$.

Next, let us examine the run-time of the process. The edge-set L_0 can be easily generated in $O(1)$ rounds. and the edge-set L_1 is generated in $\tilde{O}(r \operatorname{polyloglog}(\Delta))$ rounds. Generating edge-set L' using Lemma III.2 takes $O(k) = O(w^2 \log \Delta) = \tilde{O}(\log \Delta \operatorname{polylog}(r))$ rounds. Overall, the complexity is $\tilde{O}(r + \log \Delta)$.

Finally, observe that L_0 has maximum degree $\Delta 2^{4-i}$, and $L' \subseteq L_1 \subseteq L_0$. So L' also has maximum degree $\Delta 2^{4-i}$. ∎

V. Finding fractional hypergraph matchings

The hypergraph matching algorithms of Theorems I.1 and I.2 are based on finding a high-weight fractional matching, which we describe in this section. It is critical here to keep track of how close the fractional matching is to being integral. We use the following definition:

Definition V.1 (q-proper fractional matching)**.** *A fractional matching $h : E \to [0, 1]$ is* q-proper *if all the entries of h are rational numbers with denominator q.*

Thus, an integral matching is a 1-proper fractional matching, and an arbitrary fractional matching can be regarded as ∞-proper. There is a simple correspondence between q-proper fractional matchings and bounded-degree hypergraphs, as we explain in the next definition:

Definition V.2 (replicate hypergraph)**.** *Given a hypergraph $H = (V, E)$ and a q-proper fractional matching h, we define the* replicate hypergraph $H^{[h]}$ *to be the hypergraph with vertex set V, and whose edges form a multi-set E', wherein each edge $e \in E$ has $h(e)q$ copies in E'.*

Note that if h is q-proper fractional matching, then $a(h) = a(H^{[h]})/q$. This correspondence goes the other way as well: for a hypergraph H of degree Δ, the fractional matching which assigns $h(e) = 1/\Delta$ for every edge is a Δ-proper fractional matching.

The following result shows how to obtain the desired fractional matching. The proof is based on an algorithm of [19] to solve packing or covering LP systems, along with some techniques for quantizing edge weights inspired by Lotker et al. [24]. These are relatively routine details so we defer the proof to Appendix A.

Lemma V.3. *Let H be a hypergraph with maximum degree Δ, along with an edge-weighting function $a : E \to [0, \infty)$, for which a^* is the maximum weight fractional matching.*

1) *There is a deterministic $O(\log^2(\Delta r))$-round algorithm to generate a fractional matching h which is $O(\Delta)$-proper and which satisfies $a(h) \geq \Omega(a^*)$.*
2) *There is a randomized $O(\log r \log(\Delta r))$-round algorithm to generate a fractional matching h which is $O(\log r)$-proper and which satisfies $\mathbb{E}[a(h)] \geq \Omega(a^*)$.*

This gives the main deterministic approximation algorithm for hypergraph matching:

Theorem I.1. *Let $H = (V, E)$ be a hypergraph which is provided with a good coloring of H. For an arbitrary edge-weight function $a : E \to [0, \infty)$, there is a deterministic $\tilde{O}(r \log \Delta + \log^2 \Delta)$-round algorithm to generate a matching M with $a(M) \geq \Omega(a^*/r)$.*

Proof: Use Lemma V.3 to obtain a fractional matching h which is $O(\Delta)$-proper and with $a(h) \geq \Omega(a^*)$ in $O(\log^2(\Delta r))$ rounds. Hypergraph $H^{[h]}$ has maximum degree $\Delta' = O(\Delta)$. By Theorem IV.1, we can generate a matching M with $a(M) \geq \Omega(a(H^{[h]})/\Delta') = \Omega(a^*)$ in $\tilde{O}(\log^2 \Delta' + r \log \Delta') = \tilde{O}(\log^2 \Delta + r \log \Delta)$ rounds. ■

We next turn to the randomized algorithm. To emphasize that it is truly local, we show that our algorithm achieves success probability of $1 - \delta$ for an arbitrary parameter $\delta \in (0, 1/2)$, which may depend on n or any other quantities. The strategy is to first randomly sparsify the graph so that $\Delta \approx \text{polylog}\, \frac{1}{\delta}$, and then use our deterministic algorithm on the resulting sparsified graph. Note that this strategy is very different from a straightforward simulation of direct rounding.

Theorem I.2. *For a hypergraph $H = (V, E)$ with an arbitrary edge-weight function $a : E \to [0, \infty)$ and any parameter $\delta \in (0, 1/2)$, there is a randomized $\tilde{O}(\log \Delta + r \log \log \frac{1}{\delta} + (\log \log \frac{1}{\delta})^2)$-round algorithm to generate a matching M such that $a(M) \geq \Omega(a^*/r)$ with probability at least $1 - \delta$.*

Proof: We execute $t = \lceil c \log \frac{1}{\delta} \rceil$ independent parallel applications of the randomized part of Lemma V.3, for some constant $c > 0$ to be determined. This runs in $O(\log r \log(\Delta r))$ rounds and generates fractional matchings $h_1, \ldots, h_t$ which are each $O(\log r)$-proper and which satisfy $\mathbb{E}[a(h_i)] \geq \Omega(a^*)$.

Now set $H'_i = H^{[h_i]} = (V, E'_i)$, and consider the hypergraph $H' = (V, E')$ where we define the multi-set $E' = E'_1 \cup \cdots \cup E'_t$. We claim that, with probability at least $1 - \delta$, hypergraph H' has a fractional matching of value $\Omega(a^*)$. For, we have $\mathbb{E}[a(h_i)] \geq \Omega(a^*)$, and $a(h_i) \leq a^*$ with probability one by definition of a^*. Applying Markov's inequality to the non-negative random variable $a^* - a(h_i)$ shows that $a(h_i) \geq \Omega(a^*)$ with probability $\Omega(1)$. Since $a(h_1), \ldots, a(h_t)$ are independent random variables, there is a probability of at least $1 - 2^{-\Omega(t)}$ that $a(h_i) \geq \Omega(a^*)$ for at least one value of i; in this case also H' has a fractional matching of value $\Omega(a^*)$. By choosing c sufficiently large, we can ensure that this is at least $1 - \delta/2$.

Now randomly choose a coloring of E' with $c = 4(rt)^{10}/\delta$ colors, and discard all pairs of edges with the same color. Let $E'' \subseteq E'$ denote the set of retained edges. Clearly, we have a $\text{poly}(t, r, 1/\delta)$ coloring of the hypergraph $H'' = (V, E'')$. Simple calculations show that with probability at least $1 - \delta$ the hypergraph H'' has a fractional matching of value $\Omega(a^*)$. Furthermore, the hypergraph H'' has maximum degree $\Delta'' = O(t \log r) = O(\log \frac{1}{\delta} \log r)$.

Let us suppose that this event has occurred, and so the maximum-weight fractional matching of H'' has value $a^{**} = \Omega(a^*)$. In $O(\log^* \frac{1}{\delta})$ rounds the coloring of H'' can be converted into $\text{poly}(t, r)$-coloring of H''. Finally, we apply Theorem I.1 to get a matching M of H'' in $\tilde{O}(r \log \Delta'' + \log^2 \Delta'')$ rounds with $a(M) \geq \Omega(a^{**}) = \Omega(a^*)$. With our value of Δ'' this takes $\tilde{O}(r \log \log \frac{1}{\delta} + (\log \log \frac{1}{\delta})^2)$ rounds and the overall algorithm has run-time $\tilde{O}(\log \Delta + r \log \log \frac{1}{\delta} + (\log \log \frac{1}{\delta})^2)$. ■

VI. Maximum-weight graph matching

We now consider the problem of approximate GMWM. We consider throughout a graph $G = (V, E)$ of maximum degree Δ equipped with an edge-weighting function $a : E \to [0, \infty)$. We let T denote the (unknown) maximum weight of any matching; our goal is to find a matching M with $a(M) \geq (1 - \epsilon)T$, for some desired parameter $\epsilon > 0$. We refer to such M as an *ϵ-near matching* of G. The overall plan is to iteratively augment the matching, bringing it closer to the maximum at each stage. This basic idea has been used for parallel algorithms in [16], adapted to the setting in [27], [13].

For any set of edges $L \subseteq E$, we define the *gain* of L (with respect to matching M) as

$$g(L) = a(L - M) - a(L \cap M)$$

For a matching M of G, an *ℓ-augmentation* is a path P of length at most 2ℓ which alternately passes through matched and unmatched edges, and has the additional property that if it ends at an edge $(u, v) \in E - M$ or starts at an edge $(v, u) \in E - M$, then vertex v must be unmatched in M. This property allows us to augment the matching with respect to P, obtaining a new matching M' with $a(M') = a(M) + g(P)$.

Similarly, if we have a collection $\mathcal{P} = \{P_1, \ldots, P_t\}$ of vertex-disjoint augmenting paths, then we can augment them all, getting a new matching M' with $a(M') = a(M) + g(\mathcal{P})$ where we define $g(\mathcal{P}) = g(P_1) + \cdots + g(P_t)$. We quote the following key result from [30]:

Proposition VI.1 ([30])**.** *Let M be an arbitrary matching of G. For an integer $\ell \geq 1$, there is a collection $\mathcal{P}$ of vertex-disjoint ℓ-augmentations with $g(\mathcal{P}) \geq \frac{1}{2}(T(1 - 1/\ell) - a(M))$.*

For any integer $\ell \geq 1$, define the *path hypergraph* $\overline{H}_\ell$ to have vertex set V and to have a hyperedge $\{v_1, \ldots, v_s\}$ for every path or cycle $(v_1, v_2, \ldots, v_s)$ of length $s \leq 2\ell$ in G. Here $\overline{H}_\ell$ is a multi-hypergraph of rank-2ℓ and maximum degree at most $\Delta^{2\ell}$, and a matching of it corresponds to a collection of length-2ℓ vertex-disjoint paths in G.

Our algorithm begins with finding augmenting paths in the graph, which we do as follows:

Proposition VI.2. *Let $\ell \geq 1$ be an integer.*

1) *If we are given a good coloring of $\overline{H}_\ell$, there is a deterministic $\tilde{O}(\ell^3 \log^2 \Delta)$-round algorithm to find a matching M' with $a(M') \geq a(M) + \Omega(\frac{T(1-1/\ell)-a(M)}{\ell})$*
2) *There is a randomized $\tilde{O}(\ell^2 \log \Delta + \ell^2 \log\log \frac{1}{\delta} + \ell(\log\log \frac{1}{\delta})^2)$-round algorithm to find a matching M' such that $a(M') \geq a(M) + \Omega(\frac{T(1-1/\ell)-a(M)}{\ell})$ with probability at least $1-\delta$.*

Proof: Let H be the hypergraph on vertex set V, with hyperedges corresponding to every ℓ-augmentation with respect to matching M. The weight of a hyperedge is the gain of the corresponding path. A collection of vertex-disjoint ℓ-augmentations corresponds to a matching of the hypergraph H. Therefore, by Proposition VI.1, the hypergraph H has a matching N^* with $g(N^*) \geq \Omega(T(1-1/\ell) - a(M))$.

Note that $\Delta' = \Delta^{2\ell}$ is an upper bound on the maximum degree of H as well as $\overline{H}_\ell$.

For the deterministic algorithm, we execute Theorem I.1 to get a matching N of H with

$$g(N) \geq \Omega(g(N^*)/\ell) \geq \Omega(\frac{T(1-\frac{1}{\ell}) - a(M)}{\ell})$$

It takes $\tilde{O}(\ell \log \Delta' + \log^2 \Delta') = \tilde{O}(\ell^2 \log^2 \Delta)$ steps on H to execute Theorem I.1. Since a communication step of H can be simulated in $O(\ell)$ rounds on G, we get an overall run-time of $\tilde{O}(\ell^3 \log^2 \Delta)$.

The randomized version is similar, except we use Theorem I.2 instead of Theorem I.1. ∎

Theorem I.3. *Let $\epsilon \in (0,1)$. There is a deterministic $\tilde{O}(\epsilon^{-4} \log^2 \Delta + \epsilon^{-1} \log^* n)$-round algorithm to find an ϵ-near graph matching. For any $\delta \in (0,1)$, there is a randomized $\tilde{O}(\epsilon^{-3} \log \Delta + \epsilon^{-3} \log\log \frac{1}{\delta} + \epsilon^{-2}(\log\log \frac{1}{\delta})^2)$-round algorithm to find an ϵ-near graph matching with probability at least $1-\delta$.*

Proof: Let us first describe the deterministic algorithm. Set $\ell = \lceil 2/\epsilon \rceil$ and let $T' = T(1-\frac{\epsilon}{2})$. Our first step is to get a good coloring of $\overline{H}_\ell$ in $O(\epsilon^{-1} \log^* n)$ rounds. We then start with matching $M = \emptyset$ and we repeatedly apply Proposition VI.2 with parameter ℓ to augment M. Letting M_i be the matching after i iterations of this process, we have:

$$\begin{aligned} a(M_{i+1}) &\geq a(M_i) + \Omega(\frac{T(1-1/\ell) - a(M_i)}{\ell}) \\ &\geq a(M_i) + \Omega(\epsilon(T' - a(M_i))) \end{aligned}$$

Let us define $B_i = T' - a(M_i)$. We can rewrite the above recurrence relation for $a(M_i)$ as $B_{i+1} \leq B_i(1-\Omega(\epsilon))$. Since $B_0 = T' \leq T$, this implies that $B_i \leq T(1-\Omega(\epsilon))^i$, and therefore for $t = O(\frac{\log \frac{1}{\epsilon}}{\epsilon})$ we get $B_t \leq T\epsilon/2$. Thus the matching M_t satisfies

$$a(M_t) = T' - B_t \geq T(1-\tfrac{\epsilon}{2}) - \tfrac{\epsilon}{2}T = T(1-\epsilon)$$

Each iteration of Proposition VI.2 takes $\tilde{O}(\ell^3 \log^2 \Delta)$ rounds, and there are $t = \tilde{O}(1/\epsilon)$ rounds overall, so we get an overall run-time of $\tilde{O}(\epsilon^{-4} \log^2 \Delta + \epsilon^{-1} \log^* n)$.

For the randomized algorithm, use part (2) of Proposition VI.2 with parameter $\delta' = \frac{\delta}{t}$. The analysis is completely analogous to the deterministic algorithm. ∎

A. Lower bounds

Let us compare our algorithm with known lower bounds. First, [19] showed a lower bound of $\Omega(\min(\sqrt{\frac{\log n}{\log\log n}}, \frac{\log \Delta}{\log\log \Delta}))$ rounds for any constant-factor approximation to GMWM. Since our algorithm is parameterized mostly in term of Δ, the lower bound in n is not directly comparable to our upper bounds. As far as the bound in terms of Δ, we note that [19] provided a nearly-matching randomized constant approximation running in $O(\log \Delta)$ rounds. This was improved by [4] to $O(\frac{\log \Delta}{\epsilon^3 \log\log \Delta})$ rounds for an ϵ-near matching with constant probability. Thus, the randomized round complexity of maximum matching is precisely $\Theta(\frac{\log \Delta}{\log\log \Delta})$. Our randomized algorithm matches this up to $\log\log \Delta$ factors, and our deterministic algorithm matches this up to a factor of $\log \Delta$.

Additionally, [5] showed an $\Omega(1/\epsilon)$ lower bound on the run-time for deterministic or randomized algorithms to get ϵ-near matchings.

To finish the picture of the lower bounds for GMWM approximation, we show that the $\log^* n$ term is needed, by a simple reduction to 3-coloring a ring graph. We defer the proof to Appendix A.

Theorem VI.3. *Any deterministic c-approximation algorithm for GMWM uses $\Omega(\frac{\log^* n}{c})$ rounds.*

VII. Hypergraph maximal matching and applications

For any hypergraph H, let us define $\tau(H)$ to be the size of the largest matching in H; note that $\tau(H) \leq n$ always. If H is understood, we just write τ. We first note that the algorithms for approximate maximum matching immediately gives algorithms for maximal matching.

Theorem I.4. *There is an deterministic algorithm to find a maximal matching of H, running in $\tilde{O}((r \log \tau(H))(r \log \Delta + \log^2 \Delta) + \log^* n) \leq \tilde{O}(r^2 \log \Delta \log n + r \log^2 \Delta \log n)$ rounds. There is a randomized algorithm to find a maximal matching of H with probability at least $1-\delta$ running in $\tilde{O}((r \log \tau(H))(\log \Delta + r \log\log \frac{1}{\delta} + (\log\log \frac{1}{\delta})^2))$ rounds.*

Proof: We describe only the deterministic algorithm; the randomized algorithm is completely analogous. We first compute a good coloring of H in $O(\log^* n)$ rounds and use this for the remainder of the algorithm.

Consider using edge-weighting function $a(e) = 1$, and repeatedly applying Theorem I.1 to find a matching of the residual graph. Let R_i denote the residual graph after i iterations and let L_i be the size of the maximum matching of R_i. Initially, $R_0 = H$ and $L_0 = \tau(H)$. After stage i, Theorem I.1 gives a matching M of size $|M| \geq \Omega(L_i/r)$.

Any matching of R_{i+1} could be combined with M to give a matching of R_i, and thus $L_{i+1} \leq L_i - M \leq L_i(1 - \Omega(1/r))$. This implies that $L_i \leq n(1 - \Omega(1/r))^i \leq ne^{-\Omega(i/r)}$. Thus $L_i = 0$ for $i \geq \Omega(r \log n)$. This implies that R_i must be empty, so that the matching thus obtained is maximal. Each iteration of Theorem I.1 runs in $\tilde{O}(r \log \Delta + \log^2 \Delta)$ time. ∎

For small values of Δ there is an alternative algorithm, based on combining the MIS algorithm of [11] with the randomized hypergraph matching algorithm via the "shattering" technique developed in [3]. Most of the analysis of has already been done; we need one additional technical result. Since the proof depends on some non-standard concentration bounds, we defer it to Appendix A.

Proposition VII.1. *If $\tau(H) \geq (r \log n)^{10}$, then there is an $O(\log r \log \Delta)$-round randomized algorithm to find a matching M such that $|M| \geq \Omega(\tau(H)/r)$ w.h.p.*

This leads to our main randomized HMM algorithm;

Theorem VII.2. *There is an $\tilde{O}(r \log^2 \Delta + r^2(\log \log n)^2 + r(\log \log n)^3)$-round randomized algorithm to get a maximal matching of a hypergraph H w.h.p.*

Proof: To summarize, the algorithm has three phases. The first phase is the MIS algorithm of [11] on the line graph; the second is applying Proposition VII.1 for a few iterations; the final phase is applying Theorem I.4.

We first apply the first randomized stage of the MIS algorithm of [11] to the line graph G of H. As G has maximum degree $\Delta' = r\Delta$ and has $m \leq n\Delta$ nodes, this takes $O(\log \Delta') = O(\log(r\Delta))$ rounds. This generates a partial matching M' such that, w.h.p, the residual graph with respect to M' has every connected component containing at most $\text{poly}(r, \Delta) \log n$ nodes.

All of the connected components at this stage will be handled independently, so let us consider some arbitrary component H' of the residual hypergraph; we need to find a maximal matching of H'. Initially, $\alpha_{H'} \leq \text{poly}(r, \Delta) \log n$ (since that is the maximum number of nodes in H'). As long as $\alpha_{H'} \geq (r \log n)^{10}$, each application of Proposition VII.1 generates w.h.p a matching of size $\Omega(\alpha_{H'}/r)$. Thus, $\alpha_{H'}$ shrinks by a $(1-\Omega(1/r))$ factor. After $O(r \log(r\Delta))$ rounds, we reduce $\alpha_{H'}$ to size $(r \log n)^{10}$. Since each round takes time $O(\log r \log \Delta)$, this overall takes time $\tilde{O}(r \log^2 \Delta)$.

At this point, the residual component H' has $\alpha_{H'} \leq (r \log n)^{10}$. We finish by applying Theorem I.4 to H' with parameter $\delta = 1/n^{10}$, in $\tilde{O}(r \log \log n \log \Delta + r^2(\log \log n)^2 + r(\log \log n)^3)$ rounds. ∎

At this point, we have a number of incomparable HMM algorithms, which we summarize as follows:

Theorem VII.3. *Consider a hypergraph H with rank r, maximum degree Δ, maximum matching size τ, n vertices, and m edges, where all these parameters are globally known. There are LOCAL distributed algorithms with the following complexities:*

(a) $O(\log(r\Delta)) + 2^{O(\sqrt{\log \log(mn)})}$ *rounds and failure probability* $1/\text{poly}(mn)$.
(b) $O(\log m)$ *rounds and failure probability* $1/\text{poly}(m)$.
(c) $\tilde{O}((r \log \tau)(r \log \Delta + \log^2 \Delta) + \log^* n) \leq \tilde{O}((r \log n)(r \log \Delta + \log^2 \Delta))$ *rounds and failure probability zero.*
(d) $\tilde{O}((r \log \tau)(\log \Delta + r \log \log \frac{1}{\delta} + (\log \log \frac{1}{\delta})^2))$ *rounds and failure probability δ.*
(e) $\tilde{O}(r \log^2 \Delta + r^2(\log \log n)^2 + r(\log \log n)^3)$ *rounds and failure probability* $1/\text{poly}(n)$.

Proof: Parts (a) and (b) are the MIS algorithm [11] applied to the line graph of H. Parts (c) and (d) are restatements of Theorem I.4. Part (e) is a restatement of Theorem VII.2. ∎

HMM is used as a subroutine for a number of edge coloring problems. Plugging in the appropriate algorithms here, we obtain Theorem I.5.

Theorem I.5. *Let G be a graph with maximum degree Δ.*

1) *There is a $\tilde{O}(\log n \log^2 \Delta)$-round deterministic algorithm for $(2\Delta - 1)$-list-edge-coloring.*
2) *There is a $\tilde{O}((\log \log n)^3)$-round randomized algorithm for $(2\Delta - 1)$-list-edge-coloring*
3) *There is a $\tilde{O}(\Delta^4 \log^6 n)$-round deterministic algorithm for $\frac{3}{2}\Delta$-edge-coloring.*

Proof: For (1), Fischer, Ghaffari, and Kuhn [9] reduces $(2\Delta - 1)$-list-edge-coloring of a graph $G = (V, E)$ to maximal matching on a hypergraph of rank $r = 3$, with $O(|V| + |E|)$ vertices and maximum degree $O(\Delta^2)$. With these parameters, Theorem VII.3(c) takes $\tilde{O}(\log^2 \Delta \log n)$ time.

For (2), there are a number of cases depending on the size of Δ. See [9] for further details; the critical case is when $\Delta \leq \text{polylog}\, n$, in which case Theorem VII.3(e) takes $\tilde{O}((\log \log n)^3)$ rounds.

For (3), Ghaffari et al. [13] describe a deterministic edge-coloring algorithm which uses HMM as a black box. The run-time of this algorithm is shown to be $O(\Delta \log^3 n)$ plus $\Delta^2 \log n$ times the complexity of solving HMM on hypergraphs with n vertices, rank $r = \Delta \log n$, and maximum degree $\Delta' = \Delta^{O(r)}$.

To find the HMM, we Theorem VII.3(d) with parameter $\delta = 2^{-n^c}$ for a large constant c. This runs in time $\tilde{O}(r \log n(\log \Delta' + r \log \log \frac{1}{\delta} + (\log \log \frac{1}{\delta})^2) \leq \tilde{O}(\Delta^2 \log^4 n)$. Overall, this gives an algorithm to find the

desired edge-coloring, with run-time $\tilde{O}(\Delta^4 \log^6 n)$ and failure probability 2^{-n^c} for any desired constant c.

To derandomize this, note that there are at most $2^{\text{poly}(n)}$ possibilities for the graph G (including the ID's of all vertices). Since the randomized algorithm has failure probability 2^{-n^c}, for c sufficiently large there must exist a random seed which succeeds on *all* such graphs G. Fixing this seed (which can be determined as a function of n and Δ), we are guaranteed that the randomized algorithm for HMM does in fact succeed with probability one. ∎

VIII. Approximate Nash-William decomposition

Let us now consider a (slight variant) of the classical Nash-Williams decomposition of a multi-graph $G = (V, E)$ [26]. If G has arboricity a, then there exists an orientation of the edges such that every vertex has out-degree at most a. So, given some globally-known parameter λ which is an upper bound on the arboricity a, we will describe an algorithm to compute an edge-orientation where every vertex has out-degree at most $D = \lceil (1+\epsilon)\lambda \rceil$. We refer to this task as *approximate edge-orientation*. Note that $\lambda \leq \Delta$, and it is possible that $\lambda \ll \Delta$. By quantizing the adjacency matrix, we can assume that $\lambda \leq \text{poly}(n, 1/\epsilon)$.

In [14], Ghaffari & Su showed how to obtain such an orientation via a series of augmenting paths; they obtain a randomized algorithm running in $O(\log^4 n/\epsilon^3)$ rounds for simple graphs. This can be viewed as a HMM problem (wherein each augmenting path corresponds to a hyperedge). The deterministic HMM algorithm of [9] converts this into a deterministic algorithm, which was subsequently improved by [12] to $O(\log^{10} n \log^5 \Delta/\epsilon^9)$ rounds.

Let us first summarize the algorithm of [14] and describe how to apply our HMM algorithm. The basic outline of [14] is to maintain an orientation of the edges of G, and then proceed through a series of path augmentations for stages $i = 1, \ldots, \ell = \Theta(\frac{\log n}{\epsilon})$. We let G_i denote the resulting directed graph after stage i. Initially, the orientation G_0 can be arbitrary.

To find the augmenting path at stage i, we form an auxiliary graph G_i' from G_i by adding a source node s and a sink node t. For each vertex $v \in G_i$ of out-degree $d > D$, we add $d - D$ edges from s to v. For each vertex $v \in G_i$ of out-degree $d < D$, we add $D - d$ edges from v to t. We then select a maximal set P_i of edge-disjoint length-i directed paths in G_i' going from s to t; for each such path $p \in P_i$, we reverse the orientation of all its edges.

By a blocking path argument, each iteration of this process increases the length of the shortest augmenting paths. Furthermore, the number of potential paths of length i increases exponentially in i and so the maximum path length is at most $O(\log n/\epsilon)$. The following result of [14] formalizes this, and explains why this process works.

Theorem VIII.1 ([14])**.** *The graph G_i' has no $s-t$ paths of length strictly less than i. The graph G_ℓ, for $\ell = O(\log n/\epsilon)$, has all its vertices with out-degree at most D.*

In order to find the path-set P_i, we form an associated hypergraph H_i, whose edge set consists of all length-i paths in G_i' going from s to t, and whose vertex set corresponds to all edges of G_i'. A maximal matching of H_i is a maximal set of length-i edge-disjoint paths in G_i'.

Proposition VIII.2. *Hypergraph H_i has $\text{poly}(n\lambda)$ vertices and at most $n^3(2\lambda)^i$ edges.*

Proof: G has $m \leq n\lambda$ edges. H_i has a vertex for each of these, plus for each of the special edges leaving s and coming to t. Each vertex of degree d has at most d special edges, so this contributes another factor of m as well.

For the edge bound, let U denote the set of vertices $v \in G_i$ with out-degree larger than D. We claim that any $s-t$ path p in G_i' contains at most one vertex $v \in U$. For, suppose that it contains two such vertices v_1, v_2, where v_1 comes before v_2. We could short-circuit this path, getting a path directly from s to v_2 to t, which has length strictly less than i in G_i', contradicting Theorem VIII.1.

Thus, in order to enumerate a directed path $p = (s, v_1, v_2, \ldots, v_{i-2}, t)$ in G_i', we begin by looping over v_1, v_2. We also must loop over which copy of the edge from s to v_1 to take; this has at most na choices since the degree of v_1 can be at most $m \leq n\lambda$. For each $v_3, \ldots, v_{i-2}$, we know that v_i is an out-neighbor of v_{i-1}, which has out-degree at most $D \leq 2\lambda$. Consequently, there are at most 2λ choices for each $v_3, \ldots, v_{i-2}$. Overall, we have $n^3\lambda(2\lambda)^{i-4}$ choices for the path p. ∎

Theorem I.6 (The deterministic part)**.** *There is a deterministic $\tilde{O}(\frac{\log^6 n}{\epsilon^4})$-round algorithm to compute an approximate edge-orientation.*

Proof: First consider using the algorithm of Theorem VII.3(d) to find the maximal matching of each hypergraph H_i, with failure probability $\delta = 2^{-(n/\epsilon)^c}$ for some constant to be determined. Note that H_i has at most $b = n^3(2\lambda)^i$ edges, so it has maximum degree $\Delta \leq b$. Therefore, the HMM algorithm takes $\tilde{O}(\log \alpha_{H_i}(i \log b + i^2 \log\log \frac{1}{\delta} + i(\log\log \frac{1}{\delta})^2))$ rounds. Noting that $\tau \leq n, i \leq \ell = O(\frac{\log n}{\epsilon}), \log\log \frac{1}{\delta} = O(\log \frac{n}{\epsilon})$ and $\log b \leq O(\frac{\log n \log \lambda}{\epsilon})$, this is $\tilde{O}(\frac{\log^4 n}{\epsilon^2})$.

To derandomize this, note that as $\lambda \leq \text{poly}(n, 1/\epsilon)$, there are at most $2^{\text{poly}(n,1/\epsilon)}$ possibilities for the graph G (including the ID's of all vertices). Since the randomized algorithm has failure probability $2^{-(n/\epsilon)^c}$, for c sufficiently large there must exist a random seed which succeeds on *all* such graphs G. Fixing this seed (which can be determined as a function of n, ϵ, and other globally-known parameters) gives a deterministic algorithm.

Since it requires $i \leq O(\frac{\log n}{\epsilon})$ rounds on G to simulate

a round on H, we find the HMM of hypergraph H_i in $\tilde{O}(\frac{\log^5 n}{\epsilon^3})$ rounds. There are $\ell = O(\frac{\log n}{\epsilon})$ rounds in total. ■

We can get further advantage for the randomized algorithm by using sparsification.

Theorem I.6 (The randomized part)**.** *There is a randomized $\tilde{O}(\frac{\log^3 n}{\epsilon^3})$-round algorithm to compute an approximate edge-orientation w.h.p.*

Proof: We first show how to get a randomized algorithm running in $\tilde{O}(\frac{\log^3 n \log \lambda}{\epsilon^3})$ rounds. We first get a maximal matching of H by applying Theorem VII.3(b). Since H has at most $b = n^3(2\lambda)^\ell$ edges, this takes $O(\log b) = O(\ell \log \lambda)$ steps w.h.p. Simulating H takes $O(\ell)$ rounds with respect to G and the algorithm has $O(\ell)$ iterations, so we get an overall complexity of $\tilde{O}(\frac{\log^3 n \log \lambda}{\epsilon^3})$ rounds.

We next remove the $\log \lambda$ factor. If $\lambda \leq O(\frac{\log n}{\epsilon^2})$, then the $\log \lambda$ term is already hidden in the $\tilde{O}$ notation. Otherwise, randomly partition the edges as $E = E_1 \cup \cdots \cup E_k$, for $k = \lceil \lambda / y \rceil$ classes, where $y = \frac{c \log n}{\epsilon^2}$ for a sufficiently large constant c.

We claim that w.h.p, each graph (V, E_i) has arboricity at most $y(1+\epsilon)$. For, consider some edge-orientation A of G with out-degree at most λ. In the edge-orientation $A \cap E_i$, each vertex v has at most y outgoing edges in expectation. By Chernoff's bound, due to the size of y w.h.p. the number of outgoing edges does not exceed $y(1+\epsilon)$ for any vertex.

We now run the previous randomized algorithm in parallel on each (V, E_i) with parameter $\epsilon/10$, getting an edge-orientation of maximum out-degree $\lceil (\lambda/k)(1+\epsilon/10)^2 \rceil$. If we combine all these edge-orientations, then any vertex has out-degree at most $k\lceil (\lambda/k)(1+\epsilon/10)^2 \rceil \leq \lambda(1+\epsilon/10)^2 + k \leq \lambda(1+\epsilon/10)^2 + \lambda\epsilon^2/c$. For ϵ sufficiently small and c sufficiently large, this is at most $\lambda(1+\epsilon)$. ■

IX. Acknowledgments

Thanks to Mohsen Ghaffari and Fabian Kuhn for helpful discussions and reviewing some early drafts. Thanks to anonymous conference reviewers for helpful comments.

References

[1] Ahmadi, M., Kuhn, F., Oshman, R.: Distributed approximate maximum matching in the CONGEST model. Proceedings of the 32nd International Symposium on Distributed Computing (DISC) (2018).

[2] Balliu, A., Brandt, S., Hirvonen, J., Olivetti, D., Rabie, M., Suomela, J.: Lower bounds for maximal matchings and maximal independent sets. arxiv:1901.02441 (2019)

[3] Barenboim, L., Elkin, M., Pettie, S., Schneider, J.: The locality of distributed symmetry breaking. Journal of the ACM 63-3, Article #20 (2016)

[4] Bar-Yehuda, R., Censor-Hillel, K., Ghaffari, M., Schwartzman, G.: Distributed approximation of maximum independent set and maximum matching. Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), pp. 165-174 (2017)

[5] Ben-Basat, R., Kawarabayashi, K., Schwartzman, G.: Parameterized distributed algorithms. arxiv:1807.04900 (2018)

[6] Czygrinow, A., Hańćkowiak, M.: Distributed algorithm for better approximation of the maximum matching. International Computing and Combinatorics Conference (COCOON), pp. 242-251 (2003)

[7] Even, G., Medina, M. Ron, D.: Distributed maximum matching in bounded degree graphs. Proceedings of the 2015 International Conference on Distributed Computing and Network (ICDCN), Article #18 (2015)

[8] Fischer, M.: Improved deterministic distributed matching via rounding. Proceedings of the 31st International Symposium on Distributed Computing (DISC), pp. 17:1–17:15 (2017)

[9] Fischer, M., Ghaffari, M., Kuhn, F.: Deterministic distributed edge-coloring via hypergraph maximal matching. Proceedings of the 58th annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 180-191 (2017)

[10] Fraigniaud, P., Heinrich, M., Kosowki, A.: Local conflict coloring. Proceedings of the 57th annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 625-634 (2016)

[11] Ghaffari, M.: An improved distributed algorithm for maximal independent set. Proceedings of the 27th annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 270-277 (2016)

[12] Ghaffari, M., Harris, D., Kuhn, F.: On derandomizing local distributed algorithms. Proceedings of the 59th annual IEEE Symposium on Foundations of Computer Science, pp. 662-673 (2018)

[13] Ghaffari, M., Kuhn, F., Maus, Y., Uitto, J.: Deterministic distributed edge-coloring with fewer colors. Proceedings of the 50th annual ACM SIGACT Symposium on Theory of Computing (STOC), pp. 418-430 (2018)

[14] Ghaffari, M., Su, H.: Distributed degree splitting, edge coloring, and orientations. Proceedings of the 28th annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 2505-2523 (2017)

[15] Harris, D..: Derandomized concentration bounds for polynomials, and hypergraph maximal independent set. Proceedings of the 29th annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 2161-2180 (2018)

[16] Hougardy, S., Vinkemeier, D.: Approximating weighted matchings in parallel. Information Processing Letters 99-3, pp. 119-123 (2006)

[17] Kawarabayashi, K., Schwartzman, G.: Adapting local sequential algorithms to the distributed setting. Proceedings of the 32nd annual Symposium on Distributed Computing (DISC), Article #35 (2018)

[18] Korman, A., Sereni, J., Viennot, L.: Toward more localized local algorithms: removing assumptions concerning global knowledge. Distributed Computing 26-5,6, pp. 289-308 (2013)

[19] Kuhn, F., Moscibroda, T., Wattenhofer, R.: The price of being near-sighted. Proceedings of the 17th annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 980-989 (2006)

[20] Kuhn, F., Wattenhofer, R.: On the complexity of distributed graph coloring. Proceedings of the 25th annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 7-15 (2006)

[21] Linial, N.: Distributive graph algorithms – global solutions from local data. Proceedings of the 28th annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 331-335 (1987)

[22] Linial, N.: Locality in distributed graph algorithms. SIAM Journal on Computing 21-1, pp. 193-201 (1992)

[23] Lotker, Z., Patt-Shamir, B., Pettie, S.: Improved distributed approximate matching. Journal of the ACM 62-5, Article #38 (2015)

[24] Lotker, Z., Patt-Shamir, B., Rosén, A.: Distributed approximate matching. SIAM Journal on Computing 39-2, pp. 445-460 (2009)

[25] Luby, M.: A simple parallel algorithm for the maximal independent set problem. SIAM Journal on Computing 15-4, pp. 1036-1053 (1986)

[26] Nash-Williams, C.: Decomposition of graphs into closed and endless chains. Proceedings of the London Mathematical Society 3-1, pp. 221-238 (1960)

[27] Nieberg, T.: Local, distributed weighted matching on general and wireless topologies. Proceedings of the 5th International Workshop on Foundations of Mobile Computing (DIALM-POMC), pp. 87-92 (2008)

[28] Panconesi, A., Rizzi, R.: Some simple distributed algorithms for sparse networks. Distributed Computing 14-2, pp. 97-100 (2001)

[29] Peleg, D.: Distributed computing: a locality-sensitive approach. SIAM (2000)

[30] Pettie, S., Sanders, P.: A simpler linear $2/3-\epsilon$ approximation for maximum weight matching. Information Processing Letters 91, pp. 271-276 (2004)

[31] Schmidt, J., Siegel, A., Srinivasan, A.: Chernoff-Hoeffding bounds for applications with limited independence. SIAM Journal on Discrete Mathematics 8-2, pp. 223-250 (1995)

[32] Schudy, W., Sviridenko, M.: Concentration and moment inequalities for polynomials of independent random variables. Proceedings of the 23rd annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 437-446 (2012)

APPENDIX

Consider the following 1-round randomized distributed algorithm. First, we form an edge-set $L \subseteq E$, wherein each edge $e \in E$ goes into L independently with probability $p = \frac{0.1}{r\Delta}$. Next, we form a matching M from L by discarding any pair of intersecting edges. One can easily check that the resulting matching M has expected weight of

$$\begin{aligned}\mathbb{E}[a(M)] &\geq \sum_{e\in E} a(e)\Pr(e \in L) \\ &\quad - \sum_{\substack{e,e'\in N(v)\\ e'\neq e}} a(e)\Pr(e \in L \wedge e' \in L) \\ &\geq \Omega(\frac{a(E)}{r\Delta})\end{aligned}$$

We derandomize this via a pessimistic estimator of $\mathbb{E}[a(M)]$. Let us define the following flags computed by each edge e and vertex v of H:

$$\begin{aligned}F_e &= [[e \in L]]a(e) \\ F_v &= \sum_{\substack{e,e'\in N(v)\\ e\neq e'}} -[[e \in L \wedge e' \in L]]a(e)\end{aligned}$$

We compute $\sum_{u\in V\cup E}\mathbb{E}[F_u]$ as:

$$\begin{aligned}\sum_{u\in V\cup E}\mathbb{E}[F_u] &= \sum_{e\in E} a(e)\Pr(e \in L) \\ &\quad - \sum_{v\in V}\sum_{\substack{e,e'\in N(v)\\ e'\neq e}} a(e)\Pr(e \in L \wedge e' \in L)\end{aligned}$$

and we can estimate

$$\begin{aligned}&\sum_{v\in V}\sum_{\substack{e,e'\in N(v)\\ e\neq e'}} a(e)\Pr(e \in L \wedge e' \in L) \\ &\quad\leq \sum_{v\in V}\sum_{e,e'\in N(v)} a(e)p^2 \leq \sum_{v\in V}\sum_{e\in N(v)} a(e)p^2\deg(v) \\ &\quad\leq \sum_{v\in V}\sum_{e\in N(v)} a(e)\Delta\frac{0.1^2}{r^2\Delta^2} \\ &\quad\leq \frac{0.01}{r^2\Delta}\sum_{v\in V}\sum_{e\in E} a(e) \leq \frac{0.01a(E)}{r\Delta}\end{aligned}$$

where the last inequality holds by double-counting, noting that the rank of H is at most r.

This implies that

$$\sum_{u\in V\cup E}\mathbb{E}[F_u] = \frac{0.1a(E)}{r\Delta} - \frac{0.01a(E)}{r\Delta} \geq \Omega(\frac{a(E)}{r\Delta})$$

Lemma II.3 gives a $\tilde{O}(r\Delta)$-round deterministic algorithm to find random bits such that $\sum_u F_u \geq \sum_u \mathbb{E}[F_u]$. Given these random bits, let L denote the corresponding set of marked edges. When we form the matching M from L by

discarding edges, we get

$$\begin{aligned} a(M) &\geq \sum_{e \in E} [[e \in L]] a(e) \\ &\quad - \sum_{\substack{e,e' \in N(v) \\ e \neq e'}} [[e \in L \wedge e' \in L]](a(e)) \\ &= \sum_u F_u \end{aligned}$$

Therefore, the overall matching has $a(M) \geq \Omega(\frac{a(E)}{r\Delta})$ as desired.

We begin by using Lemma II.3 for degree-splitting.

Lemma A.1. *Let parameters $\epsilon, \delta \in [0,1]$ be chosen so that $\Delta \geq 100 \log(r/\delta)/\epsilon^2$. Given a good coloring of H, there is a deterministic algorithm in $\tilde{O}(r \log \frac{1}{\delta}/\epsilon^2)$ rounds to generate disjoint edge subsets $L_1, L_2 \subseteq E$ with the following two properties:*

1) $a(L_1 \cup L_2) \geq (1-\delta)a(E)$
2) *Every vertex $v \in V$ has $\deg_{L_j}(v) \leq (1+\epsilon)\Delta/2$ for $j = 1, 2$.*

Proof: We can ignore any node $v \in V$ with $\deg(v) \leq \Delta/2$, since it is under no obligation.

Define $\alpha = 40 \log(r/\delta)/\epsilon^2$. We construct a new hypergraph $H' = (E, U)$ by dividing every node $v \in V$ into $\ell = \lfloor \deg_H(v)/\alpha \rfloor$ virtual nodes $u_1, \ldots, u_\ell$ in H' and we assign each of the hyperedges of v to exactly one of the virtual nodes $u_1, \ldots, u_\ell$ in such a way that each virtual node u_i has $\deg(u_i) \in [\alpha, 2\alpha)$ and $\sum_{i=1}^{\ell} \deg(u_i) = \deg(v)$. This subdivision process is possible due to our assumption that every node in V (that we are not ignoring) has degree at least $\Delta/2 \geq \alpha$.

Our construction will have three parts. First, we define a function $F : U \to [0, \infty)$, which can be computed via a 1-round randomized algorithm on H'. Next, we derandomize this to select random bits such that $\sum_{u \in U} F_u \leq \sum_{u \in U} \mathbb{E}[F_u]$. Finally, having fixed the random bits, we construct L_1, L_2.

To begin, we randomly partition the edges E into two sets L_1', L_2', wherein each edge e goes into L_1' or L_2' independently with probability $1/2$. For each virtual node $u \in U$ and $j = 1, 2$, we define $Z_{u,j} = \deg_{L_j'}(u)$, and we set

$$F_u = a(N(u)) \big[\big[\bigvee_{j=1,2} Z_{u,j} > \deg(u)(1+\epsilon)/2 \ \big] \big]$$

Now let us compute $\sum_u \mathbb{E}[F_u]$. For a node $u \in U$, the value $Z_{u,j}$ is a binomial random variable with mean $\deg(u)/2 \geq \alpha/2$. By the Chernoff bound,

$$\Pr(Z_{u,j} > \deg(u)(1+\epsilon)/2) \leq e^{-\epsilon^2 \deg(u)/6} \leq e^{-\epsilon^2 \alpha/16}$$

Thus, overall we have

$$\begin{aligned} \sum_{u \in U} \mathbb{E}[F_u] &\leq \sum_{u \in U} \sum_j a(N(u)) \Pr(Z_{u,j} > \deg(u)(1+\epsilon)/2) \\ &\leq \sum_{u \in U} 2a(N(u)) e^{-\epsilon^2 \alpha/16} \end{aligned}$$

Since H' has rank r, by double-counting we have $\sum_{u \in U} a(N(u)) \leq r a(E)$, and thus overall

$$\sum_u \mathbb{E}[F_u] \leq a(E)(2re^{-\epsilon^2 \alpha/16}) \leq \delta a(E)$$

where the last inequality follows from our choice of α.

By Lemma II.3, there is a deterministic $O(r\alpha)$-round algorithm to find random bits such that $\sum_u F_u \leq \sum_u \mathbb{E}[F_u] \leq \delta a(E)$. Now, suppose we have fixed such random bits; in particular, the sets L_1', L_2' are determined. We form L_1, L_2 by starting with the sets L_1', L_2' and then discarding any edges incident to a vertex $u \in U$ with $Z_{u,j} > \deg(u)(1+\epsilon)/2$.

By summing the virtual nodes u corresponding to a node $v \in V$, we see that

$$\begin{aligned} \deg_{L_j}(v) &\leq \sum_{\substack{u \in U \\ \text{corresponding to } v}} (1+\epsilon)\deg(u)/2 \\ &\leq (1+\epsilon)\deg(v)/2 \leq (1+\epsilon)\Delta/2 \end{aligned}$$

Furthermore, this ensures that

$$\begin{aligned} &a(L_1 \cup L_2) \\ &\quad \geq a(E) - \sum_{u \in U} a(N(u)) [[\bigvee_{j=1,2} Z_{u,j} > (1+\epsilon)\deg(u)/2]] \\ &\quad = a(E) - \sum_{u \in U} F_u \end{aligned}$$

which is by construction at least $(1-\delta)a(E)$. ∎

By iterating this degree-splitting, we prove Lemma II.6.

Proof of Lemma II.6: We will iteratively partition the edge sets for stages $i = 0, 1, \ldots, s-1$ where $s = \lfloor \log_2 k \rfloor$, so that at the i^{th} stage we have sets $T_{i,1}, \ldots, T_{i,2^i}$ such that $|N(v) \cap T_{i,j}| \leq (1+\epsilon)^i \Delta/2^i$, and so that $a(T_{i,1} \cup \cdots \cup T_{i,2^i}) \geq (1-\delta')^i a(E)$, for parameters $\epsilon = \frac{1}{4s}, \delta' = \frac{\delta}{4s}$.

We define the hypergraph $H_{i,j} = (V, T_{i,j})$ and we form the sets $T_{i+1,2j}, T_{i+1,2j+1}$ by applying Lemma A.1 to $H_{i,j}$. This is done in parallel for all values of j.

We claim that $\Delta_i = (1+\epsilon)^i \Delta/2^i$ is an upper bound on the degree of each $H_{i,j}$. We show this by induction. The base case is trivial. For the induction step, let us first show that the condition of Lemma A.1 is satisfied, namely

$$\Delta_i \geq 100 \log(r/\delta')/\epsilon^2$$

Since $\Delta_i = (1+\epsilon)^i \Delta/2^i \geq \Delta/2^s$ and $s \leq \log_2 k$, it suffices to show that

$$\frac{\Delta}{2^{(\log_2 k)}} \geq 100 \log(4r(\log_2 k)/\delta)/\epsilon^2 \tag{8}$$

Our hypothesis gives inequality (8) for C sufficiently large. Therefore, Lemma A.1 ensures that every vertex v has $\deg_{T_{i+1,j}}(v) \le (1+\epsilon)\Delta_i/2 = \Delta_{i+1}$, thus completing the induction. Furthermore, since Lemma A.1 applies at each step, we have $a(T_{i+1,2j} \cup T_{i+1,2j+1}) \ge a(T_{i,j})(1-\delta')$ for every i, j.

Let $E_i = T_{i,1} \cup \cdots \cup T_{i,2^i}$. Summing over j gives $a(E_{i+1}) \ge a(E_i)(1-\delta')$, which implies that

$$a(E_s) \ge a(E)(1-\delta')^s \ge a(E)(1-\delta)$$

Now set $E' = E_s$. For $e \in E'$, we define $\chi(e)$ to be the unique value j with $e \in T_{t,j}$. This ensures that every vertex $v \in V$ has $|N(v) \cap T_j| \le \Delta_s \le (1+\epsilon)^s \Delta/2^s \le 2\Delta/2^s \le 4\Delta/k$ as desired.

This procedure has s stages of applying Lemma A.1, each of which costs $\tilde{O}(r\log(1/\delta')/\epsilon^2) = \tilde{O}(r\log\frac{1}{\delta}s^2)$. Thus, overall the cost is $\tilde{O}(rs^3\log\frac{1}{\delta}) = \tilde{O}(r\log^3 k\log\frac{1}{\delta})$. ∎

We begin with a few elementary observations about directional derivatives; the proofs will be found in the full paper.

Proposition A.2. *Suppose that Φ is uncorrelated for vertices $v, w_1, \ldots, w_s$. Then for any values $x, y_1, \ldots, y_s$ we have*

$$\begin{aligned}&\mathbb{E}[D_v\Phi \mid R_v = x, R_{w_1} = y_1, \ldots, R_{w_s} = y_s]\\ &\quad = \mathbb{E}[D_v\Phi \mid R_v = x]\end{aligned}$$

Proposition A.3. *For any vertex v and value $x \in \{0,1\}$, we have $\mathbb{E}[D_v\Phi \mid R_v = x] \ge 0$ iff $\mathbb{E}[\Phi \mid R_v = x] \le \mathbb{E}[\Phi \mid R_v = \overline{x}]$.*

Full proof of Lemma III.2: We proceed through stages $i = 1, \ldots, k$; at each stage i, every vertex v with $\chi(v) = i$ selects some value ρ_v, and commits to $R_v = \rho_v$. Therefore, after each round i, all the values R_v for $\chi(v) = i$ can be regarded as fixed values.

In analyzing round i, we suppose that we have fixed the values ρ_w for every vertex w with $\chi(w) < i$. All our expectation calculations will be conditioned on such vertices having $R_w = \rho_w$. Bearing this in mind, each vertex v with $\chi(v) = i$ selects ρ_v so that $\mathbb{E}[D_v\Phi \mid R_v = \rho_v] \ge 0$.

We first claim that this information can be computed by v in $O(1)$ rounds. By Property (A1), the conditional expectation $\mathbb{E}[D_v\Phi \mid R_v = x]$ only depends on the values of R_w for $w \in N(v)$. Thus in $O(1)$ rounds v can query the values of the ρ_w for $w \in N(v)$, and use (A3) to determine $\mathbb{E}[D_v\Phi \mid R_v = x]$ for all values x. By Proposition A.3, there is at least one such value ρ_v which ensures $\mathbb{E}[D_v\Phi \mid R_v = \rho_v] \ge 0$. (Namely, the value ρ_v which minimizes $\mathbb{E}[\Phi \mid R_v = \rho_v]$.

Next, we claim that the conditional expectation of Φ does not increase during the round i. Suppose that the color-i vertices are $v_1, \ldots, v_s$ and they select values $\rho_{v_1}, \ldots, \rho_{v_s}$ respectively. By Property (A2), Φ is uncorrelated for $v_1, \ldots, v_s$. We claim now that for all $j = 1, \ldots, s$ we have

$$\begin{aligned}&\mathbb{E}[\Phi \mid R_{v_1} = \rho_{v_1}, \ldots, R_{v_j} = \rho_{v_j}]\\ &\qquad \le \mathbb{E}[\Phi \mid R_{v_1} = \rho_{v_1}, \ldots, R_{v_{j-1}} = \rho_{v_{j-1}}] \quad (9)\end{aligned}$$

To show this, we compute:

$$\begin{aligned}&\mathbb{E}[\Phi \mid R_{v_1} = \rho_{v_1}, \ldots, R_{v_{s-1}} = \rho_{v_{s-1}}, R_{v_j} = \overline{\rho_{v_j}}]\\ &\quad - \mathbb{E}[\Phi \mid R_{v_1} = \rho_{v_1}, \ldots, R_{v_{j-1}} = \rho_{v_{j-1}}, R_v = \rho_{v_j}]\\ &\quad = \mathbb{E}[D_{v_j}\Phi \mid R_{v_1} = \rho_{v_1}, \ldots, R_{v_j} = \rho_{v_j}]\\ &\quad = \mathbb{E}[D_{v_j}\Phi \mid R_{v_j} = \rho_{v_j}] \qquad \text{by Proposition A.2}\\ &\quad \ge 0 \qquad \text{by definition of } \rho_{v_j}\end{aligned}$$

Thus, we have shown that

$$\begin{aligned}&\mathbb{E}[\Phi \mid R_{v_1} = \rho_{v_1}, \ldots, R_{v_j} = \rho_{v_j}]\\ &\quad \le \mathbb{E}[\Phi \mid R_{v_1} = \rho_{v_1}, \ldots, R_{v_{j-1}} = \rho_{v_{j-1}}, R_{v_j} = \overline{\rho_{v_j}}]\end{aligned}$$

To obtain Eq. (9), we now sum over y:

$$\begin{aligned}&\mathbb{E}[\Phi \mid R_{v_1} = \rho_{v_1}, \ldots, R_{v_{j-1}} = \rho_{v_{j-1}}]\\ &\quad = \tfrac{1}{2}\,\mathbb{E}[\Phi \mid R_{v_1} = \rho_{v_1}, \ldots, R_{v_{j-1}} = \rho_{v_{j-1}}, R_{v_j} = \rho_{v_j}]\\ &\qquad + \tfrac{1}{2}\,\mathbb{E}[\Phi \mid R_{v_1} = \rho_{v_1}, \ldots, R_{v_{j-1}} = \rho_{v_{j-1}}, R_{v_j} = \overline{\rho_{v_j}}]\\ &\quad \ge \tfrac{1}{2}\,\mathbb{E}[\Phi \mid R_{v_1} = \rho_{v_1}, \ldots, R_{v_{j-1}} = \rho_{v_{j-1}}, R_{v_j} = \rho_{v_j}]\end{aligned}$$

∎

We begin by considering an edge-weighting function $a : E \to [1, W]$ which has *bounded* range; by quantizing edge weights later we remove the dependence on W.

Lemma A.4. *For a hypergraph $H = (V, E)$ and edge-weighting function $a : E \to [1, W]$, there is a deterministic $O(\epsilon^{-4}\log(Wr)\log(W\Delta))$-round algorithm to find a fractional matching h with $a(h) \ge (1-\epsilon)a^*$.*

Proof: The problem of finding a maximum-weight fractional matching h can be interpreted as a type of packing LP, namely

$$\begin{aligned}&\text{maximize } \sum_e a(e)h(e)\\ &\qquad\qquad \text{subject to } \forall v \sum_{e \in N(v)} h(e) \le 1 \quad (10)\end{aligned}$$

Kuhn, Moscibroda & Wattenhofer [19] provides a deterministic algorithm for solving such problems. Their analysis applies to generic packing LP's; however, it requires these to be parameterized in the following form:

$$\text{maximize } \sum_i x(i) \qquad \text{subject to } x \in \mathbb{R}^n, Ax \le c$$

All the entries of the constraint matrix A must either have $A_{ij} = 0$ or $A_{ij} \ge 1$. Furthermore, if $A_{ij} > 0$ and $A_{i'j} > 0$ for two rows i, i', the communications graph must have an edge from i to i'. Note that with this parameterization, the corresponding dual LP has a very similar nice structure.

For LP's in this form, there are two key parameters Γ_p and Γ_d (here, p and d stand for primal and dual) which determine the run-time of the [19] algorithm. These are defined as

$$\Gamma_p = \big(\max_{j'} c_{j'}\big)\big(\max_j \frac{\sum_{i=1}^n A_{ij}}{c_j}\big), \qquad \Gamma_d = \max_i \sum_{j=1}^m A_{ij}$$

With this parametrization, [19] runs in time $O(\epsilon^{-4}\log\Gamma_p \log\Gamma_d)$ to get a $(1+\epsilon)$-approximation.

To transform the fractional matching LP into this form, we define variables $x(e) = a(e)h(e)$ for each edge e, and our constraints become

$$\max \sum_e x(e) \qquad \text{subject to } \forall v \sum_{e \ni v} (W/a(e))x(e) \le W \tag{11}$$

Here the constraint matrix A is given by

$$A_{ev} = \begin{cases} W/a(e) & \text{if } e \in N(v) \\ 0 & \text{if } e \notin N(v) \end{cases}$$

which has its entries either zero or in the range $[1, W]$.

Given a solution vector x to Eq. (11), we will then set $h(e) = x(e)/a(e)$; this will clearly satisfy the fractional matching LP given by Eq. (10). The LP of Eq. (11) has the form required by [19], with the constraint vector c having all its entries equal to W. Therefore, we have

$$\Gamma_p = W \max_{v \in V} \sum_{e \in N(v)} (W/a(e)) \le W\Delta,$$
$$\Gamma_d = \max_{e \in E} \sum_{v \in e} (W/a(e)) \le Wr$$

With these parameters, the algorithm of [19] runs in $O(\epsilon^{-4}\log(W\Delta)\log(Wr))$ rounds to get a $(1+\epsilon)$ approximation to maximum fractional matching. ∎

The next results describe deterministic and randomized methods to *partially* discretize a fractional matching, achieving fractional matchings which are $O(\Delta)$ and $O(\log r)$-proper respectively, while losing only a constant factor in the weight.

Proposition A.5. *Let H be a hypergraph of degree Δ with an edge-weighting function $a : E \to [1, W]$, along with a fractional matching h' satisfying $a(h') \ge a^*/2$.*

1) *There is a deterministic $O(1)$-round algorithm to generate an 10Δ-proper fractional matching h with $a(h) \ge \Omega(a^*)$.*
2) *There is a randomized $O(1)$-round algorithm to generate a $\lceil 20\log r\rceil$-proper fractional matching h with $\mathbb{E}[a(h)] \ge \Omega(a^*)$.*

Proof:
(1) Form h by rounding down h' to the nearest multiple of $\delta = \frac{1}{10\Delta}$. So h is clearly 10Δ-proper. The loss incurred is at most $\delta a(E)$, i.e. $a(h) \ge a(h') - \delta a(E)$. By hypothesis, $a(h') \ge a^*/2$. Also, we must have $a^* \ge a(E)/\Delta$, as there is a trivial fractional matching (setting $a(e) = 1/\Delta$ for every edge) with this value. Thus, $a(h) \ge a^*/2 - 0.1a(E)/\Delta \ge \Omega(a^*)$.

(2) Consider the following random process: first, we generate edge multi-set L' consisting of X_e copies of each edge e where X_e is a Poisson random variable with rate $p(e) = 10h'(e)\log r$. Next, if any vertex v has more than $20\log r$ neighbors in L', it discards all such neighbors; any surviving edges (which are not discarded) go into the set L.

Clearly $\mathbb{E}[a(L')] = 10a(h')\log r$. Let us compute the probability that an edge e gets removed from L', due to some vertex $v \in e$ having a degree exceeding $20\log r$. For every vertex $v \in e$, the value $Z = \deg_{L'-e}(v)$ is a Poisson random variable with mean at most $10\log r$ (since h' is a fractional matching). By our assumption that $r \ge 2$, we have $20\log r - 1 \ge 1.86(10\log r)$ and so Chernoff's bound gives $\Pr(Z \ge 20\log r - 1) \le e^{-10\log r \times 0.86^2/3} \le r^{-2.46}$. A union bound over the vertices in e shows that e is removed from L' with probability at most $r^{-1.46}$.

Since this holds for every edge $e \in L'$, the expected weight of removed edges is at most $\sum_{e\in E} p(e) \times a(e) \times r^{-1.46} = 10a(h')r^{-1.46}\log r \le 3.7a(h')\log r$. Summing over e we thus get $\mathbb{E}[a(L)] \ge 10a(h')\log r - 3.7a(h')\log r \ge \Omega(a^*\log r)$. Since the resulting edge-set L' has maximum degree $20\log r$, it follows that setting $h(e) = \frac{[[e\in L]]}{\lceil 20\log r\rceil}$ gives a fractional matching h which is $\lceil 20\log r\rceil$-proper and which has $a(h) \ge \Omega(\frac{a(L)}{\log r})$. So $\mathbb{E}[a(h)] \ge \Omega(\frac{\mathbb{E}[a(L)]}{\log r}) = \Omega(a^*)$. ∎

The following key quantization result now allows us to remove the parameter W.

Lemma A.6. *Let $H = (V, E)$ be a hypergraph with edge-weighting function $a : E \to [0, \infty)$, and let $q \ge 2$ be an arbitrary integer parameter. For all integers i, define E_i to be the set of edges e with $(rq)^i \le a(e) < (rq)^{i+1}$. Suppose we are given q-proper fractional matchings h'_i for each hypergraph $H_i = (V, E_i)$. Then there is an $O(1)$-round deterministic algorithm to generate an $O(q)$-proper fractional matching h of H with $a(h) \ge \Omega(\sum_i a(h'_i))$.*

Proof Sketch: Form h as follows: for any edge $e \in E_i$, if there is some $f \in E_j$ with $h'_j(f) > 0$ and $f \cap e \ne \emptyset$, for $j \ge i+3$, then set $h(e) = 0$; otherwise, set $h(e) = h'_i(e)/3$. Formally, we set:

$$h(e) = \frac{h'_i(e)}{3}[[\bigwedge_{j=i+3}^{\infty} \bigwedge_{\substack{f\in E_j \\ f\cap e\ne\emptyset}} h'_j(f) = 0]]$$

The proof that this satisfies the requirements of the lemma is standard but somewhat technical, so we defer it to the full paper. ∎

Finally, we are ready to prove Lemma V.3.

Proof of Lemma V.3: Let us first show the deterministic algorithm. We ignore any edges of weight zero, and let

E_i denote the set of edges with weights in the range $[(rq)^i, (rq)^{i+1})$ for parameter $q = 10\Delta$. Let $H_i = (V, E_i)$ for each value i. Let a_i^* denote the maximum weight fractional matching of H_i; note that $\sum_i a_i^* \geq a^*$.

Our first step is to find a fractional matching h_i' of each H_i using Lemma A.4. Each hypergraph H_i has edge weights in the range $(rq)^i$ to $(rq)^{i+1}$. When applying Theorem IV.1 to H_i, we may divide all the edge weights by $(rq)^i$; the rescaled edge weights are then in the range $[1, W]$ for $W = rq$. So Lemma IV.1 uses $O(\log^2(\Delta r))$ rounds, and generates a matching h_i' with $a(h_i') \geq a_i^*/2$. Next, we use Proposition A.5 to get fractional matchings h_i with $a(h_i) \geq \Omega(a_i^*)$ and such that each h_i is q-proper. Finally, we use Lemma A.6 to combine these fractional matchings h_i into a single fractional matching h which is $O(q)$-proper and which satisfies $a(h) \geq \sum_i a(h_i) \geq \sum_i \Omega(a_i^*) \geq \Omega(a^*)$.

The randomized algorithm is the same, except we use the randomized part of Proposition A.5 to ensure that $\mathbb{E}[a(h_i)] \geq \Omega(a_i^*)$ and that each h_i remains q-proper for $q = \lceil 20 \log r \rceil$. Thus, $\mathbb{E}[a(h)] \geq \sum_i \mathbb{E}[a(h_i')] \geq \Omega(a^*)$. Also, with this value of q, the edge weights when applying Theorem IV.1 have range $W = rq = O(r \log r)$. So Theorem IV.1 uses only $O(\log r \log(\Delta r))$ rounds. ■

We show here that the dependence on network size n is necessary for any deterministic GMWM algorithm. Our construction will use the ring graph (with $\Delta = 2$) and will use the unit edge-weighting function (i.e. $a(e) = 1$ for all edges $e \in E$).

Proposition A.7. *Suppose that A is a deterministic t-round algorithm which guarantees a c-approximation to graph maximum matching. Running algorithm A on a ring graph gives a matching M with the property that every contiguous sequence of $8ct$ edges has at least one edge in M.*

Proof: Suppose that, for some choice of vertex labels, running A on the n-vertex ring graph G has a contiguous sequence at vertices $v_1, \ldots, v_\ell$ without any edges of M. Now form the ring graph G' consisting only of the vertices $v_1, \ldots, v_\ell$ (the vertex v_ℓ becomes joined to the vertex v_1).

The vertices $v_t, \ldots, v_{\ell-t}$ will not see any difference in their t-neighborhood compared to G, and so when we run A on G', the resulting matching M' will not have any edges between $v_t, \ldots, v_{\ell-t}$. Therefore, the $|M'| \leq 2\lceil t/2 \rceil \leq t+1$. On the other hand, a maximum matching of G has size $\lfloor \ell/2 \rfloor \geq \ell/2 - 1$. Since A guarantees a c-approximation, we must have have $t+1 \geq \frac{(\ell/2)-1}{c}$, i.e. $\ell \leq 2(1+c+ct) < 8ct$. ■

Proof of Theorem VI.3: By Proposition A.7, when we run a t-round algorithm A on a ring graph G, we generate an matching M such that every vertex in G sees an edge of M within distance $8ct$. We can form an $8ct+1$-ruling set U for G, by placing into U the lower-ID vertex of each edge of M. This allows to generate a 3-coloring of G in $O(ct)$ rounds: we sort the vertices by their increasing distance from the closest element of U, and at each stage $i = 0, \ldots, 8ct+1$, the vertices at distance i greedily choose a color. On the other hand, Linial [22] showed that 3-coloring a ring graph requires $\Omega(\log^* n)$ rounds. Thus $t \geq \Omega(\frac{\log^* n}{c})$. ■

Apply Lemma A.4 with the edge-weight function $a(e) = 1 \; \forall e$ to get a fractional matching h with $h(E) \geq \Omega(\tau)$. Next, use the following simple randomized rounding procedure: we form a set M' by placing each edge e into M independently with probability $p_e = h(e)/(10r)$. If any vertex has more than 1 neighbor in M', it discards them all and M is the resulting matching.

Let $s = h(E)$ and define the indicator variable $X_e = [[e \in M']]$, so that $|M| \geq \sum_{e \in E} X_e - \sum_{e \in E} \sum_{e' \cap e \neq \emptyset} X_e X_{e'}$. As $\mathbb{E}[\sum_{e \in E} X_e] = \frac{s}{10r}$, a standard Chernoff bound calculation shows that $\sum_{e \in E} X_e \geq \frac{s}{20r}$ with probability at least $1 - e^{-\Omega(s/r)}$; by our assumption on τ, this is at least $1 - 1/\operatorname{poly}(n)$. Thus, let us define

$$S = \sum_{e \in E} \sum_{e' \cap e \neq \emptyset} X_e X_{e'}$$

and we will show $|S| \leq \frac{s}{50r}$ w.h.p.; this will show that $|M| \geq \frac{s}{20r} - \frac{s}{50r} \geq \Omega(\tau/r)$ w.h.p.

Observe that S is a quadratic polynomial applied to the independent variables X_e, each of which is Bernoulli-p_e. We use a general concentration bound of [32] for such polynomials; this bound requires significant notation to state properly, but to summarize it depends on parameters μ_i, where μ_i is the maximum expected value of any order-i partial derivative of S. In our case, the second derivatives of S have value at most 1, while the expected zeroth derivative (aka the expectation of S) is given by

$$\begin{aligned} \mu_0 &= \sum_{e \in E} \sum_{e \cap e' \neq \emptyset} p_e p_{e'} \leq \sum_{e \in E} p_e \sum_{v \in e} \sum_{e' \in N(v)} \frac{h(e')}{10r} \\ &\leq \sum_{e \in E} \frac{p_e}{10r} = \frac{s}{100r} \end{aligned}$$

Finally, for any fixed edge e, the expected first derivative in the direction of X_e is bounded as

$$\mu_1 = \max_e \sum_{e' : e \cap e' \neq \emptyset} p_{e'} \leq \max_e \sum_{v \in e} \sum_{e' \in N(v)} \frac{h(e')}{10r} \leq \frac{1}{10}$$

So here $\mu_1, \mu_2 \leq O(1)$ and $\mu_0 \gg \log^2 n$. A simple application of [32] shows that $S \leq 2\,\mathbb{E}[S]$ w.h.p.; please see [32] for further explication and definitions. Therefore, w.h.p. $S \leq \frac{s}{50r}$ as desired.