Exponentially Faster Massively Parallel Maximal Matching

Soheil Behnezhad, MohammadTaghi Hajiaghayi, and David G. Harris
University of Maryland
soheil@cs.umd.edu, hajiagha@cs.umd.edu, davidgharris29@gmail.com

***Abstract*—**

The study of approximate matching in the *Massively Parallel Computations* (MPC) model has recently seen a burst of breakthroughs. Despite this progress, however, we still have a far more limited understanding of *maximal matching* which is one of the central problems of parallel and distributed computing. All known MPC algorithms for maximal matching either take polylogarithmic time which is considered inefficient, or require a strictly super-linear space of $n^{1+\Omega(1)}$ per machine.

In this work, we close this gap by providing a novel analysis of an extremely simple algorithm. This affirmatively resolves the conjecture of Czumaj *et al.* [STOC'18] that a variant of this algorithm might work. The algorithm edge-samples the graph, randomly partitions the vertices, and finds a random greedy maximal matching within each partition. We show that this algorithm drastically reduces the vertex degrees. This, among some other results, leads to an $O(\log\log\Delta)$ round algorithm for maximal matching with $O(n)$ space (or even *mildly sublinear* in n using standard techniques).

As an immediate corollary, we get a 2 approximate *minimum vertex cover* in essentially the same rounds and space. This is the best possible approximation factor under standard assumptions, culminating a long line of research. It also leads to an improved $O(\log\log\Delta)$ round algorithm for $1+\varepsilon$ approximate matching. All these results can also be implemented in the *congested clique* model within the same number of rounds.

I. Introduction

The success of modern parallel frameworks such as MapReduce [19], Hadoop [36], or Spark [38] has resulted in an active area of research over the past few years for understanding the true computational power of such systems. The *Massively Parallel Computations* (MPC) model, which provides a clean abstraction of these frameworks, has become the standard theoretical model for this purpose (see Section II-B for the model).

In this work, we consider the *maximal matching* problem in the MPC model. It is one of the most fundamental graph problems in parallel and distributed computing with far reaching practical and theoretical implications. The study of maximal matching can be traced back to PRAM algorithms of 1980s [31, 24, 3] and has been studied in various computational models since then.

Soheil Behnezhad and Mohammad Hajiaghayi were supported in part by NSF CAREER award CCF-1053605, NSF AF:Medium grant CCF-1161365, NSF BIGDATA grant IIS-1546108, and NSF SPX grant CCF-1822738. Soheil Behnezhad was also supported in part by a Google PhD Fellowship.

In the MPC model, maximal matching is particularly important; an algorithm for it directly gives rise to algorithms for $1+\varepsilon$ approximate *maximum matching*, $2+\varepsilon$ approximate *maximum weighted matching*, and 2 approximate *minimum vertex cover* with essentially the same number of rounds and space. Each of these problems has been studied on its own [18, 5, 21, 6, 7, 14, 34, 2, 13].

Known bounds: For many graph problems, including maximal matching, $O(\log n)$ round MPC algorithms can be achieved in a straightforward way by simulating PRAM algorithms [31, 24, 3] using $n^{\Omega(1)}$ space. This bound was recently improved by Ghaffari and Uitto [22] to $\widetilde{O}(\sqrt{\log\Delta})$ rounds with the same memory. The main goal, however, is to obtain significantly faster (i.e., subpolylogarithmic round) algorithms by further utilizing MPC's additional powers.

Currently, the only known such algorithm for maximal matching is that of Lattanzi *et al.* [28] which requires $O(1/\delta)$ rounds using a space of $O(n^{1+\delta})$. Their algorithm's round complexity, however, blows up back to $\Theta(\log n)$ as soon as memory becomes $O(n)$. In comparison, due to a breakthrough of Czumaj *et al.* [18], we have algorithms for $1+\varepsilon$ approximate matching that take $O(\log\log n)$ rounds using a space of $O(n)$ [18, 21, 6]. Unfortunately, this progress on approximate matching offers no help for maximal matching or related problems. In fact, these algorithms also require up to $\Omega(\log n)$ rounds to maintain maximality.

Our contribution: In this paper, we give MPC algorithms for maximal matching that are exponentially faster than the state-of-the-art (we describe our precise results in Section I-A). We achieve this by providing a novel analysis of an extremely simple and natural algorithm.

The algorithm edge-samples the graph, randomly partitions the vertices into disjoint subsets, and finds a greedy maximal matching within the induced subgraph of each partition. This partitioning is useful since each induced subgraph can be sent to a different machine. We show that if we commit the edges of each of these greedy matchings to the final output, the vertex degrees in the residual graph are drastically dropped. Czumaj *et al.* [18] had conjectured that a variant of this algorithm might work and left its analysis

2575-8454/19/$31.00 ©2019 IEEE
DOI 10.1109/FOCS.2019.00096

as one of their main open problems:[1]

> "*Finally, we suspect that there is a simpler algorithm for the problem* [...] *by simply greedily matching high-degree vertices on induced subgraphs* [...]. *Unfortunately, we do not know how to analyze this kind of approach.*" [18]

We summarize our results and their implications in Section I-A and give a high-level overview of the analysis in Section I-B.

A. Main Results

Theorem 1 (main result)**.** *Given an n-vertex graph G with m edges and max degree Δ, there exists a randomized* MPC *algorithm for computing a maximal matching that*

(1) takes $O(\log\log\Delta)$ rounds using $O(n)$ space per machine,

(2) or takes $O(\log\frac{1}{\delta})$ rounds using $O(n^{1+\delta})$ space per machine, for any parameter $\delta \in (0,1)$.

The algorithm succeeds w.e.h.p.[a] and requires an optimal total space of $O(m)$.

[a] We say an event occurs *with exponentially high probability* (w.e.h.p.) if it occurs with probability $1 - e^{-n^{\Omega(1)}}$.

Theorem 1 part (1) provides the first subpolylogarithmic round MPC algorithm for maximal matching that does not require a super-linear space in n. In fact, it improves exponentially over the prior algorithms in this regime, which all take polylogarithmic rounds [31, 28, 22]. Furthermore, Theorem 1 part (2) exponentially improves over the δ-dependency of Lattanzi *et al.*'s algorithm [28] which requires $O(1/\delta)$ rounds using $O(n^{1+\delta})$ space.

Theorem 2. *Given an n-vertex graph G with m edges and max degree Δ, there exists an* MPC *algorithm for computing a maximal matching that takes $O(\log\log\Delta + \log\log\log n)$ rounds and uses $n/2^{\Omega(\sqrt{\log n})}$ space per machine. The algorithm succeeds w.e.h.p. and uses a total space of $O(m + n^{1+\gamma})$ for any constant $\gamma > 0$.*

Theorem 2 further improves the space per machine to *mildly sublinear* with the same round complexity (ignoring the lower terms). We comment that the $n/2^{\Omega(\sqrt{\log n})}$ space usage here goes below the $n/\operatorname{poly}\log n$ space that has commonly been considered for problems such as approximate matching [18, 21, 6] and graph connectivity [8].

Other implications: Our algorithm also has a few other implications when used as a black-box.

[1] A more detailed variant of the algorithm was also described in the following TCS+ talk by Artur Czumaj (starts from 1:03:23): https://youtu.be/eq0jwAnJu9c?t=3803.

Corollary 1. *By a well-known reduction, the set of matched vertices in a maximal matching is a 2-approximation of minimum vertex cover. As such, all algorithms of Theorems 1 and 2 can be applied to the 2-approximate minimum vertex cover problem as well.*

The problem of whether an approximate vertex cover can be found faster in MPC with $O(n)$ space was first asked by Czumaj *et al.* [18]. Subsequent works showed that indeed $O(\log\log n)$ algorithms are achievable and the approximation factor has been improved from $O(\log n)$ to $O(1)$ to $2+\varepsilon$ [5, 21, 6]. Corollary 1 reaches a culminating point: If we restrict the machines to run a polynomial-time algorithm, which is a standard assumption (see [25, 4]), no algorithm can achieve a better approximation under the Unique Games Conjecture [26].

Corollary 2. *By known reductions [15, 29], Theorem 1 directly implies an $O(\log\log\Delta)$ round algorithm for maximal matching in the congested clique model. It also leads to $O(\log\log\Delta)$ round congested clique algorithms for 2-approximate vertex cover, $1+\varepsilon$ approximate maximum matching, and $2+\varepsilon$ approximate maximum weighted matching by known reductions.*

Prior to our work, the fastest algorithm for maximal matching in the congested clique model requried polylogarithmic time [31, 22]. Corollary 2 exponentially improves over this bound.

Corollary 3. *For any constant $\varepsilon \in (0,1)$, Theorem 1 can be used to give algorithms for $1+\varepsilon$ approximate matching and $2+\varepsilon$ approximate maximum weighted matching in asymptotically the same number of rounds and space.*

The reduction from maximal matching (and in fact, any $O(1)$ approximate matching) to $1+\varepsilon$ approximate matching is due to an algorithm by McGregor [32] (see [6]) and the reduction to $2+\varepsilon$ approximate weighted matching is due to an algorithm by Lotker et al. [30] (see [18]). We also note that if the space is $O(n\operatorname{polylog} n)$, then our algorithm can be used in a framework of Gamlath *et al.* [20] to get an $O(\log\log\Delta)$ round algorithm for $1+\epsilon$ approximate maximum weighted matching.

Corollary 3 also strengthens the round-complexity of the results in [18, 21, 6] from $O(\log\log n)$ to $O(\log\log\Delta)$ using $O(n)$ space. To our knowledge, the algorithms of [18, 21, 6] do require $\Omega(\log\log n)$ rounds even when $\Delta = \operatorname{poly}\log n$ since they switch to an $O(\log\Delta)$ round algorithm at this threshold. Corollary 3, however, implies an $O(\log\log\log n)$ round algorithm on such graphs.

B. High Level Technical Overview

As discussed above, if the space per machine is $n^{1+\Omega(1)}$, we already know how to find a maximal matching efficiently [28]. The main problem, roughly speaking, is that once the

space becomes $O(n)$, the computational power of *a single machine* alone does not seem to be sufficient to have a significant effect on the whole graph. More concretely, the known algorithms that work based on ideas such as edge-sampling the graph into a single machine and finding a matching there [28, 14, 2], all require $\Omega(\log n)$ rounds of repeating this procedure if the space is $O(n)$.

Vertex partitioning [25, 10, 18, 6, 21, 17], which in the context of matching was first used by [18], helps in utilizing several machines. The general idea is to randomly partition the vertices and find a matching in the induced subgraph of each partition individually in a different machine. It turns out that the choice of the internal matching algorithm over these induced subgraphs, has a significant effect on the global progress made over the whole graph. This is, in fact, the fundamental way that the algorithms within this framework differ [18, 6, 21].

We show that *greedy maximal matching*, which is perhaps the simplest matching algorithm one can think of, has several desirable structural properties that make it a perfect candidate for this purpose. This procedure iterates over the edges for some given ordering π, and at the time of processing each edge, adds it to the matching iff none of its incident edges are part of the matching so far. In other words, it is the lexicographically-first MIS of the line graph of G. We give a brief overview of our algorithm first, then describe the key ideas behind its analysis.

The algorithm: Our main algorithm, which is formalized as Algorithm 1, uses three randomization steps, all of which are necessary for the analysis:

- An ordering π over the edges is chosen uniformly at random.
- Each edge of the graph is sampled independently with some probability p.
- For some k, the vertex set V is partitioned into disjoint subsets $V_1, \ldots, V_k$ where the partition of each vertex is chosen independently at random.

After these steps, for any $i \in [k]$, we put the edge-sampled induced subgraph of V_i into machine i and compute a greedy maximal matching M_i according to ordering π. We note that the choice of k and p in Algorithm 1 ensure that the induced subgraphs fit the memory of a machine.

The analysis outline: Observe that $M = \bigcup_{i \in [k]} M_i$ is a valid matching since the partitions are vertex disjoint. The key to our results, and the technically interesting part of our paper, is to show that if we commit the edges of M to the final maximal matching, then the degree of almost all vertices drops to $\Delta^{1-\Omega(1)}$ in the residual graph. The main challenge here is to bound the vertex degrees across the partitions.

To do this, for any vertex v and any partition $i \in [k]$, we let $Z_{v,i}$ denote the number of neighbors of v in partition i that remain unmatched in greedy matching M_i. Note that $Z_{v,i}$ is a random variable of the three randomizations involved in the algorithm, and that $\sum_{i \in [k]} Z_{v,i}$ is precisely equal to the remaining degree of vertex v. We show the abovementioned degree reduction guarantee through a concentration bound on random variable $Z_{v,i}$.

Let us first outline how a concentration bound on $Z_{v,i}$ can be useful. Suppose, wishfully thinking, that $Z_{v,i} = (1 \pm o(1))\mathbb{E}[Z_{v,i}]$ for every $i \in [k]$ with high probability. By symmetry of the partitions, we have $\mathbb{E}[Z_{v,i}] = \mathbb{E}[Z_{v,1}]$ for every $i \in [k]$. This means that all random variables $Z_{v,1}, \ldots, Z_{v,k}$ take on the same values ignoring the lower terms. Now, if $\mathbb{E}[Z_{v,1}]$ is small enough that $k \cdot \mathbb{E}[Z_{v,1}] < \Delta^{1-\Omega(1)}$, we get the desired bound on residual degree of v. Otherwise, due to the huge number of unmatched neighbors in its own partition, we show that v must have been matched and, thus, cannot survive to the residual graph!

Unfortunately, $Z_{v,i}$ is a rather complicated function and it is not straightforward to prove such sharp concentration bounds on it. Recall that Chernoff-Hoeffding bounds work only on sum of independent random variables. Furthermore, concentration bounds obtained by Azuma's or other "dimension dependent" inequalities seem useless for our purposes: because the partition of every vertex in the graph may potentially affect $Z_{v,i}$, these would give bounds on the order of $Z_{v,i} = \mathbb{E}[Z_{v,i}] \pm \widetilde{O}(\sqrt{n})$. As $\mathbb{E}[Z_{v,i}]$ should be on the order of Δ, this is useless when Δ is small.

Instead of an exponential concentration bound, we aim for a weaker concentration bound by proving an upper bound on the variance of $Z_{v,i}$. To achieve this upper bound, we use a method known as the Efron-Stein inequality (see Proposition 1) which plays a central role in our analysis. On one hand, this weaker concentration bound is still strong enough for our purpose of degree reduction. On the other hand, since we are only bounding the variance, the required conditions are much more relaxed and can be shown to be satisfied by the algorithm.

On a conceptual level, one contribution of our paper is to provide a natural example for how the Efron-Stein inequality, which is often not among the standard tools used in theoretical computer science, can be extremely useful in the analysis of randomized algorithms.

How greedy maximal matching helps: Our proof of the concentration bound relies on a number of unique properties of the random greedy maximal matching algorithm:

1) If we run greedy maximal matching on an edge-sampled subgraph of a graph, the maximum degree in the residual graph drops significantly (see Lemma 4).

2) The set of matched vertices in the greedy maximal matching changes by a constant number of elements if a single vertex or edge is removed (see Lemma 5).

3) If an ordering π is chosen randomly and an edge e of the graph is also chosen randomly, then determining whether e belongs to the greedy maximal matching according to π requires "looking" only at $O(d)$ edges of the graph in expectation, where d is the average degree of the line graph (see Proposition 4).

We summarize these properties in Section VI. Property 3 was originally developed in the context of *sublinear time algorithms* for approximating maximum matching size. To our knowledge, it was first formalized by Nguyen and Onak [33], with the precise bound of Proposition 4 proved by Yoshida *et al.* [37]. We find the application of this methodology in proving concentration bounds extraordinary and possibly of independent interest.

II. Preliminaries

A. Notation

Throughout the paper for any positive integer k, we use $[k]$ to denote the set $\{1, \ldots, k\}$. For any graph $G = (V, E)$ and any $V' \subseteq V$, we let $G[V']$ denote the induced subgraph; that is, $G[V']$ contains edge e in E if and only if both of its endpoints are in V'. For a vertex v, we define the *neighborhood* $N(v)$ to be the set of vertices u with $\{u, v\} \in E$.

An edge subset $M \subseteq E$ is a *matching* if no two edges in M share an endpoint. A matching M of a graph G is a *maximal matching* if it is not possible to add any other edge of G to M and it is a *maximum matching* if every matching in G has size at most $|M|$. When it is clear from the context, we abuse notation to use M for the vertex set of matching M. In particular, we use $G[V \setminus M]$ to denote the graph obtained by removing every vertex of M from G. Furthermore, for any vertex $v \in V$ and matching M, we define the *residual degree* $\deg^{\text{res}}_M(v)$ to be zero if $v \in M$, and otherwise $\deg^{\text{res}}_M(v) := \deg_{G[V \setminus M]}(v)$. Finally, we define the *match-status* of vertex v according to some matching M to be the indicator for the event that $v \in M$.

B. The MPC Model

The Massively Parallel Computations (MPC) model was first introduced by Karloff *et al.* [25] and further refined by [23, 11, 12, 4]. An input of size N is initially distributed among M machines, each with a local space of size S. Computation proceeds in synchronous rounds in which each machine can perform an arbitrary local computation on its data and can send messages to other machines. The messages are delivered at the start of the next round. Furthermore, the total messages sent or received by each machine in each round should not exceed its memory.

We desire algorithms that use a sublinear space per machine (i.e., $S = N^{1-\Omega(1)}$) and only enough total space to store the input (i.e., $S \cdot M = O(N)$). For graph problems, the edges of an input graph $G(V, E)$ with $n := |V|$ and $m := |E|$ are initially distributed arbitrarily among the machines, meaning that $N = \Theta(m)$ words (or $\Theta(m \log n)$ bits). Moreover, we mainly consider the regime of MPC with space per machine of $S = \Theta(n)$ words.

C. Concentration inequalities

We will use two main concentration inequalities: the *Efron-Stein* inequality and the *bounded differences inequality*. These both concern functions $f(x_1, \ldots, x_n)$ which have *Lipschitz properties*, namely, changing each coordinate x_i has a relatively small change to the value of f.

Proposition 1 (Efron-Stein inequality [35])**.** *Fix an arbitrary function $f : \{0,1\}^n \to \mathbb{R}$ and let $X_1, \ldots, X_n$ and $X'_1, \ldots, X'_n$ be $2n$ i.i.d. Bernoulli random variables. For $\vec{X} := (X_1, \ldots, X_n)$ and $\vec{X}^{(i)} := (X_1, \ldots, X_{i-1}, X'_i, X_{i+1}, \ldots, X_n)$, we have*

$$\mathrm{Var}(f(\vec{X})) \leq \frac{1}{2} \cdot \mathbb{E}\Big[\sum_{i=1}^{n} \big(f(\vec{X}) - f(\vec{X}^{(i)})\big)^2\Big].$$

We consider the following form the of the bounded differences inequality (which is a special case of McDiarmid's inequality):

Proposition 2 (Bounded differences inequality)**.** *Let f be a λ-Lipschitz function on k variables, and let $\vec{X} = (X_1, \ldots, X_k)$ be a vector of k independent (not necessarily identically distributed) random variables. Then, $\Pr\left[f(\vec{X}) \geq \mathbb{E}[f(\vec{X})] + t\right] \leq \exp\left(\frac{-2t^2}{k\lambda^2}\right)$.*

A slight reformulation of this, which is more useful for us to use directly, is the following:

Proposition 3. *Let f be a λ-Lipschitz function on k variables, and let $\vec{X} = (X_1, \ldots, X_k)$ be a vector of k independent (not necessarily identically distributed) random variables. Then w.e.h.p.,*

$$f(\vec{X}) \leq \mathbb{E}[f(\vec{X})] + \lambda n^{0.01}\sqrt{k}.$$

D. Sequential Greedy Maximal Matching

As described in Section I-B, a maximal matching can be found by a sequential greedy algorithm:

Definition 1 (Greedy maximal matching)**.** *Given a graph $G = (V, E)$ and an ordering π over the edges in E, the greedy maximal matching algorithm processes the edges in the order of π and adds an edge e to the matching if none of its incident edges have joined the matching so far. We denote the resulting maximal matching by* $\mathsf{GreedyMM}(G, \pi)$.

This greedy maximal matching has a number of nice properties that play a critical role in the analysis of our algorithm. We summarize these properties in Section VI.

We view the permutation π as a function mapping E to $[m]$; we say that e has *higher priority than* e' if $\pi(e) < \pi(e')$. In analyzing the greedy matching algorithm, it is often

convenient to use the following local method of generating the permutation: each edge e is associated with a real $\rho_e \in [0,1]$; we then form π by sorting in order of ρ. Slightly abusing notation, we write $\mathsf{GreedyMM}(G,\rho)$ in this case as shorthand for $\mathsf{GreedyMM}(G,\pi)$ where π is the permutation associated to ρ.

III. Roadmap

As discussed in Section I-B, the key to proving Theorem 1 and Theorem 2 is an algorithm to reduce the graph degree by a polynomial factor. The precise statement of this lemma is as follows:

Lemma 1 (degree reduction)**.** *There is an $O(1)$ round MPC algorithm to produce a matching M, with the following behavior w.e.h.p.: it uses $n/\Delta^{\Omega(1)}$ space per machine and $O(m)$ space in total, and the residual graph $G[V \setminus M]$ has maximum degree $\Delta^{1-\Omega(1)}$.*

Our main result, and the technical core of our analysis in proving Lemma 1 lies in showing that the following Algorithm 1 significantly reduces the degree of nearly all vertices in G.

Algorithm 1.

Input: A graph $G=(V,E)$ with maximum degree Δ.
Output: A matching M in G.

(1) **Permutation:** Choose a permutation π uniformly at random over the edges in E.
(2) **Edge-sampling:** Let $G^L(V,L)$ be an edge-sampled subgraph of G where each edge in E is sampled independently with probability $p := \Delta^{-0.85}$.
(3) **Vertex partitioning:** Partition the vertices of V into $k := \Delta^{0.1}$ groups $V_1,\ldots,V_k$ such that the partition of every vertex in V is chosen independently and uniformly at random.
(4) Each machine $i \in [k]$ receives the graph $G^L[V_i]$ and finds the greedy maximal matching $M_i := \mathsf{GreedyMM}(G^L[V_i],\pi)$.
(5) Return matching $M := \bigcup_{i=1}^{k} M_i$.

Specifically, we will show the following:

Lemma 2. *Algorithm 1 has the following desirable behavior:*

1) *W.e.h.p., it uses $n/\Delta^{\Omega(1)}$ space per machine.*
2) *W.e.h.p., it uses $O(n)+m/\Delta^{\Omega(1)}$ space in total (aside from storing the original input graph.)*
3) *The expected number of vertices $v \in V$ such that $\deg^{res}_M(v) > \Delta^{0.99}$ is at most $O(n/\Delta^{0.03})$.*

We will prove Lemma 2 in Section IV and we will prove Lemma 1 in Section V. Before this, let us show how the degree reduction algorithm of Lemma 1 can be used to prove Theorem 1.

Proof of Theorem 1: The algorithm consists of r iterations that each commits a number of edges to the final maximal matching using the algorithm of Lemma 1. In each iteration, the maximum degree in the remaining graph is reduced from Δ to $\Delta^{1-\alpha}$ given that $\Delta > c$ for some constant c and α. This ensures that by the end of iteration r, maximum degree is at most $\max\{c, \Delta^{(1-\alpha)^r}\}$.

To get the first result, take $r = \Theta(\log\log\Delta)$; at the end of this process, the residual graph has degree $O(1)$. At this point, we put the entire residual graph onto a single machine, and compute its maximal matching. To get the second result, take $r = \Theta(\log(1/\delta))$; at the end of this process, the residual graph has degree n^δ. At this point, we again put the entire residual graph onto a single machine, and compute its maximal matching. ∎

IV. Matching Almost All High-Degree Vertices: Proof of Lemma 2

We now turn to proving Lemma 2. We first need some notation for the analysis of Algorithm 1. For simplicity, we write G_i for the graph $G[V_i]$. Note that G_i is different from G_i^L in that G_i^L includes only a subset of the edges in G_i; those that were sampled in Line 2 of Algorithm 1. We let L_i be the set of edges $\{u,v\} \in L$ with $u,v \in V_i$; that is, L_i is the edge-set of G_i^L. We further define χ to be the partition function of the vertices; that is, each vertex select a value $\chi(v)$ u.a.r from $[k]$, and then we set $V_i = \chi^{-1}(i)$. We also note that throughout the proof, we assume $m \geq n^{0.9}$. This assumption comes w.l.o.g. since otherwise one can put all the edges into one machine with even sublinear memory of $O(n^{0.9})$ and find a maximal matching there.

We begin by analyzing the residual degree of a vertex within its own partition, which are some simple consequences of the method used to generate L.

Claim 1. *The following bounds on the edge set L hold w.e.h.p.:*

1) *Every $i \in [k]$ has $|V_i| = \Theta(n/\Delta^{0.1})$.*
2) *The graph G^L contains $O(m/\Delta^{0.85})$ edges.*
3) *Each graph G_i^L contains $O(n/\Delta^{0.05})$ edges.*

Proof: The first property follows from a straightforward Chernoff bound, noting that $\mathbb{E}[V_i] = n/k = n/\Delta^{0.1} \geq \mathrm{poly}(n)$. For the second property, observe that the expected number of edges in G^L is $m \cdot p = m/\Delta^{0.85}$. As we have discussed above, we can assume that $m \geq n^{0.9}$ and we also know that $\Delta \leq n$; therefore, $m/\Delta^{0.85} \geq n^{0.05}$ and by Chernoff's bound the number of such edges is $O(m/\Delta^{0.85})$ w.e.h.p. For the third property, we consider two cases where $\Delta \geq n^{0.01}$ and $\Delta < n^{0.01}$ separately.

Case 1: $\Delta \geq n^{0.01}$: For each vertex $v \in V_i$, its incident edge $e = \{u,v\}$ will belong to G_i^L if e is sampled in L

and vertex u also belongs to V_i. Both of these events occur at the same time with probability $p \cdot k^{-1} = \Delta^{-0.95}$. This means that the expected number of neighbors of v in G_i^L will be $\Delta \cdot \Delta^{-0.95} = \Delta^{0.05}$. Since we assumed $\Delta \geq n^{0.01}$, a simple Chernoff bound can show that this random variable is concentrated around $O(\Delta^{0.05})$ w.e.h.p. Combined with the first property, the number of edges in each G_i^L will be $O(n/\Delta^{0.1}) \cdot O(\Delta^{0.05}) = O(n/\Delta^{0.05})$ w.e.h.p.

Case 2: $\Delta < n^{0.01}$: Let U denote the number of edges in G_i^L. For the arguments discussed above, we still have $\mathbb{E}[U] \leq O(n/\Delta^{0.05})$. Furthermore, U can be regarded as a function of the vertex partition χ and the edge set L. There are $O(n\Delta)$ such random variables, and each of these can change U by at most Δ. Therefore, by Proposition 3, w.e.h.p., we have

$$U \leq \mathbb{E}[U] + \Delta \cdot n^{0.01} \cdot \sqrt{O(n\Delta)};$$

as $\Delta \leq n^{0.01}$ this in turn implies that $U \leq O(n/\Delta^{0.05})$ w.e.h.p. ∎

These allow us to prove the first two parts of Lemma 2:

Proof of Lemma 2 part 1 and 2: For the space bounds, Claim 1 shows that for each G_i^L, we require $O(n/\Delta^{0.05})$ space for its edges and $O(n/\Delta^{0.1})$ for its vertices. Since Δ is larger than any constant, this is smaller than $n/\Delta^{\Omega(1)}$. To show the bounds on total space usage note that the total edge count of all the graphs G_i^L is clearly at most $|L|$, since each edge lives on at most one machine, and this is at most $m/\operatorname{poly}(\Delta)$. Furthermore, storing partition of each vertex requires only $O(n)$ total space. ∎

As we have discussed before, for any vertex $v \in V$ and any $i \in [k]$, we define the random variable

$$Z_{v,i} := \big|V_i \cap N_{G[V\setminus M]}(v)\big|,$$

to be the degree of vertex v in the i^{th} partition of the residual graph $G[V\setminus M]$. Note here that v does not necessarily belong to V_i. With this definition, if a vertex v is not matched in M, we have $\deg_M^{\text{res}}(v) = Z_{v,1} + \cdots + Z_{v,k}$. We further define the related random variable Z'_v as:

$$Z'_v := \begin{cases} Z_{v,\chi(v)} & \text{if } v \notin M \\ 0 & \text{if } v \in M, \end{cases}$$

which is equivalent to the residual degree of v in its own partition.

Claim 2. *For any vertex v, we have* $\Pr(Z'_v > \Delta^{0.86}) \leq \exp(-\operatorname{poly}(\Delta))$.

Proof: We will show that this bound holds, even after conditioning on the random variables χ and π. Suppose now that $v \in V_i$ and so we need to bound the probability that $Z_{v,i} > \Delta^{0.86}$. Note, here, that $Z'_v = \deg_{M_i}^{\text{res}}(v)$. Also, M_i is formed by performing independent edge sampling on $G[V_i]$ and then taking the greedy maximal matching. Thus by Lemma 4, the probability that $Z'_v > \frac{\ln(1/\beta)}{p}$ is at most β. Setting $\beta = e^{-\Delta^{0.01}}$, we have $Z'_v > \Delta^{0.86}$ with probability at most $\exp(-\operatorname{poly}(\Delta))$. ∎

A. Analysis of the Inter-partition Degrees

The key to analyzing Algorithm 1 is to show that for most vertices v, the values of $Z_{v,i}$ take on similar values across all possible indices i. We had sketched how this leads to the desired bound on vertex degrees in Section I-B; let us provide some more technical details here.

Recall from Section I-B that our concentration inequalities should not have an additive factor depending on n or they become too weak to be useful as Δ gets smaller. To overcome this, we show that with careful analysis, the Efron-Stein inequality (Proposition 1) yields our desired concentration bound; in particular, it gives concentration on the order $Z_{v,i} = \mathbb{E}[Z_{v,i}] \pm \Delta^{1-\Omega(1)}$. However, we emphasize that this concentration bound is *not* with exponentially high probability, or even with high probability: it only holds with a relatively small probability $1 - 1/\operatorname{poly}(\Delta)$. This is the reason that we can only show that the number of high-degree vertices reduces by a $1/\operatorname{poly}(\Delta)$ factor, and not that Algorithm 1 reduces the maximum degree outright.

Due to symmetry, we may consider showing a concentration bound for $Z_{v,1}$. Let us furthermore assume that L and π have been fixed. Therefore, $Z_{v,1}$ becomes only a function of the vertex partitioning χ, or more precisely, a function of the set of vertices that belong to partition V_1. Let us define the vector $\vec{x}$, by setting $x_v = 1$ if $\chi(v) = 1$, and $x_v = 0$ otherwise. We may write $Z_{v,1}(\vec{x})$ to emphasize that $Z_{v,1}$ is merely a function of $\vec{x}$. Observe that $\vec{x}$ is a vector of n i.i.d. Bernoulli-$1/k$ random variables. To use the Efron-Stein inequality for bounding the variance, we have to upper bound the right-hand-side of inequality

$$\operatorname{Var}(Z_{v,1}) \leq \frac{1}{2}\mathbb{E}_{\vec{x}}\Big[\sum_{w\in V}\big(Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)})\big)^2\Big], \quad (1)$$

where $\vec{x}^{(w)}$ is obtained by replacing the value of x_w in $\vec{x}$ with x'_w which is drawn independently from the same distribution. In other words, the w summand of (1) corresponds to the effect of repartitioning vertex w on the value of $Z_{v,1}$. Thus, we need to show that for most of the vertices in V, whether they belong to V_1 or not does not affect $Z_{v,1}$.

To show this, consider a game where we determine $Z_{v,1}(\vec{x})$ by querying entries of $\vec{x}$. The queries can be conducted adaptively, i.e., each query can depend on the answers to previous queries. If we show an upper bound β_v on the number of queries required to determine $Z_{v,1}(\vec{x})$, then no matter what the other $n - \beta_v$ entries of $\vec{x}$ are, $Z_{v,1}(\vec{x})$ remains unchanged and so clearly $Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)}) = 0$ for all such unqueried vertices w. (The subscript v in β_v is used to emphasize that the upper bound can be different for different choices of v.) Therefore, one way to show that

most vertices of V do not affect $Z_{v,1}$ is to design an efficient query process. We also note a particularly useful property of the Efron-Stein inequality in (1) is that even an upper bound on the expected number (taken over choice of $\vec{x}$) of queries suffices.

In addition to showing that most vertices do not affect $Z_{v,1}$, we also need to show that the query process yields an appropriate Lipschitz property on $Z_{v,1}$ as well. That is, even if the query process can guarantee $Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)}) = 0$ for most vertices w, we still have to bound the value of $(Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)}))^2$ on those vertices w where $Z_{v,1}(\vec{x}) \neq Z_{v,1}(\vec{x}^{(w)})$. This also follows from the nice structure of the greedy maximal matching algorithm.

Claim 3 (Lipschitz property)**.** *For any vertex partitioning $\vec{x}$, let $\vec{x}^{(w)}$ be obtained by changing the w index of $\vec{x}$. Then $(Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)}))^2 \leq 4$.*

Proof: Suppose that $x_w = 0$ which means $x_w^{(w)} = 1$. Let V_1 and V_1' denote the vertex partitions due to $\vec{x}$ and $\vec{x}^{(w)}$ respectively, i.e., $V_1 = \{u \mid x_u = 1\}$ and $V_1' = \{u \mid x_u^{(w)} = 1\}$. Observe that V_1 and V_1' differ in only one vertex w which belongs to V_1' but not V_1. Define $M_1 := \mathsf{GreedyMM}(G[V_1], \pi)$ and $M_1' := \mathsf{GreedyMM}(G[V_1'], \pi)$. By Lemma 5 part 1, there are at most two vertices in V whose match-status differs between M_1 and M_1'. Even if these two vertices happen to be neighbors of v, we still have $|Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)})| \leq 2$ and thus get the desired bound. The case with $x_w = 1$ and $x_w^{(w)} = 0$ follows from a similar argument. ∎

The Lipschitz property can be plugged directly into (1) to show $\mathrm{Var}(Z_{v,1}) \leq O(n\Delta^{-0.1})$. In what follows, however, we describe a query process which significantly reduces this upper bound to $\mathrm{poly}(\Delta)$ for nearly all the vertices, i.e., removes the dependence on n.

The query process: We start with a query process to determine whether a given edge belongs to matching $M_1(\vec{x})$ – where here we write $M_1(\vec{x})$ to emphasize that the parameters π, L should be regarded as fixed and so matching M_1 is only a function of the vertex partitioning $\vec{x}$. This process is very similar to a generic edge oracle for the greedy matching (which we briefly discuss in Section VI), except that instead of querying the edges, it queries the entries of the vector $\vec{x}$.

Suppose that we have to determine whether a given edge $e \in L$ belongs to the matching $M_1(\vec{x})$. Instead of revealing the whole vector $\vec{x}$, first note that if one of the end-points of e does not belong to V_1, then e cannot be in the induced subgraph G_1^L and thus we can answer NO immediately. Suppose that e appears in G_1^L. Since the greedy maximal matching algorithm processes the edges in the order of π, it suffices to recursively determine whether any of the incident edges to e belongs to $M_1(\vec{x})$ in the order of their priorities. At any point that we find such incident edge to e, we immediately return NO as e certainly cannot join $M_1(\vec{x})$. Otherwise e has to join $M_1(\vec{x})$, thus we return YES. We summarize the resulting query process as $\mathcal{EO}_\pi(e, \vec{x})$:

$\mathcal{EO}_\pi(e, \vec{x})$: A query-process to determine whether $e \in M_1(\vec{x})$.

Let $e = \{u, v\}$. Query x_u and x_v; **if** $x_u = 0$ or $x_v = 0$, **then return** NO.
Let $e_1, \ldots, e_d$ be the incident edges to e in G^L sorted as $\pi(e_1) < \pi(e_2) < \cdots < \pi(e_d)$.
for $i = 1, \ldots, d$ **do**
  if $\pi(e_i) < \pi(e)$ **then**
    if $\mathcal{EO}_\pi(e_j, \vec{x}) =$ YES **then return** NO
return YES

We also define a *degree oracle* $\mathcal{DO}_\pi(v, \vec{x})$ to determine the value of $Z_{v,1}(\vec{x})$. This checks whether each $w \in N_G(v)$ appears in V_1 and is matched, which in turn requires checking whether every edge incident to w appears in matching of $G^L[V_1]$:

$\mathcal{DO}_\pi(v, \vec{x})$: A query process to determine the value of $Z_{v,1}(\vec{x})$.

$c \leftarrow 0$
for all vertices $u \in N_G(v)$ **do**
  Query x_u.
  if $x_u = 1$ **then**
    Execute $\mathcal{EO}_\pi((u, w), \vec{x})$ for all vertices $w \in N_{G^L}(u)$.
    if $\mathcal{EO}_\pi((u, w), \vec{x}) =$ NO for all such vertices w **then** $c \leftarrow c + 1$ ▷ u is unmatched in M_1
return c

Analysis of the query complexity: We now analyze the *query complexity* of the oracle $\mathcal{DO}_\pi$, i.e., the number of indices in $\vec{x}$ that it queries. For any vertex v, we let $B(v)$ denote the number of vertices that are queried when running $\mathcal{DO}_\pi(v)$. This is precisely the quantity that we need to bound for arguing that $\mathrm{Var}(Z_{v,1})$ is small according to (1). Formally:

Claim 4. *Fix any $\vec{x}, \pi, L$ and and let $\vec{x}^{(w)}$ be a vector obtained by resampling the index x_w. Then*

$$\sum_{w \in V} (Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)}))^2 \leq 4B(v).$$

Proof: By definition, the value of $Z_{v,1}(\vec{x})$ can be uniquely determined by only revealing indices of $\vec{x}$ which are quered by $\mathcal{DO}_\pi(v, \vec{x})$. Therefore, changing other indices w of $\vec{x}$ cannot affect $Z_{v,1}$ and so $Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)}) = 0$. There are $B(v)$ indices queries by v. For any such index w, Claim 3 shows that $(Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)}))^2 \leq 4$. ∎

To bound $B(v)$, let us first define $A(e)$ for an edge $e \in L_1$ to be the number of edges in L_1, on which the edge oracle is called (recursively) in the course of running $\mathcal{EO}_\pi(e, \vec{x})$. Note that when running $\mathcal{EO}$, only edges that are in L_1 can generate new recursive calls; other edges are checked, but immediately discarded.

Claim 5. *We have $\mathbb{E}[\sum_{e\in L_1} A(e)] \le O(n)$ where the expectation is taken over χ, L, and π.*

Proof: Let us first suppose that the random variables L and χ are fixed. Thus also G_i^L is determined. The only randomness remaining is the permutation π. As we are only interested in edges of L_1, the edges outside L_1 have no effect on the behavior of $\mathcal{EO}_\pi$. Thus, $A(e)$ is essentially the query complexity of $\mathsf{GreedyMM}(G_1, \pi)$ under a random permutation. By Proposition 4, we have:

$$\mathbb{E}_\pi\Big[\sum_{e\in L_1} A(e) \mid L, \chi\Big] \le O(|L_1| + |R_1|),$$

where R_1 is the set of intersecting edge pairs in G_1. Integrating now over the random variables L and χ, we get:

$$\mathbb{E}\Big[\sum_{e\in L_1} A(e)\Big] \le O(\mathbb{E}[|L_1| + |R_1|]).$$

Each edge $e \in E$ goes into L_1 with probability $p/k^2 = \Delta^{-1.05}$, and so $\mathbb{E}[|L_1|] = m\Delta^{-1.05}$. Likewise, G contains at most $m\Delta/2$ pairs of intersecting edges and each of these survives to R_1 with probability $p^2/k^3 = \Delta^{-2}$. Therefore, $\mathbb{E}[|R_1|] \le m\Delta^{-1}$. Since $m \le n\Delta$, we therefore get $\mathbb{E}[\sum_{e\in L_1} A(e)] \le O(n)$. ∎

Claim 6. *Suppose that we condition on the event that when running $\mathcal{DO}_\pi(v, \vec{x})$, we make a total of t calls to $\mathcal{EO}_\pi(e, \vec{x})$ with $e \in L_1$. Then the expected number of total entries of $\vec{x}$ queried during $\mathcal{DO}_\pi(v, \vec{x})$ is at most $O(\Delta^{1.15} + t\Delta^{0.15})$.*

Proof: Let us condition on the random variables χ, L_1 and π. This determines the full listing of all edges in L_1 that are queried during the execution of $\mathcal{DO}_\pi(v)$, because only such edges can generate new recursive calls to $\mathcal{EO}_\pi$. Thus, if we show that this bound holds conditioned on χ, L_1, π it will also show that it holds conditioned on the value t. The only remaining randomness at this point is the set $L \setminus L_1$.

Let J denote the set of edges in L_1 queried during $\mathcal{DO}_\pi(v, \vec{x})$, with $|J| = t$. Then $\mathcal{DO}_\pi(v, \vec{x})$ will query x_u for all $u \in N_G(v)$, and it will query w for all $w \in N_{G^L}(u)$ for all such $u \in N_G(v)$. Finally, whenever it encounters edge $e \in J$, it will call $\mathcal{EO}_\pi(f, \vec{x})$ for some edges $f \in L \setminus L_1$ which touch e; each of these will query two vertices, but the query process will not proceed further when they are discovered to lie outside L_1.

The number of vertices $u \in N_G(v)$ queried is clearly at most Δ. Now let us fix some $u \in N_G(v)$ and count the number of vertices $w \in N_{G^L}(u)$ queried. This is precisely $\deg_L(u)$, and for any fixed u, the expected number of such vertices w is at most $\Delta p = \Delta^{0.15}$. Thus, the total expected number of queried vertices in the first two categories is at most $\Delta^{1.15}$.

Finally, let us consider some edge $e = (a, b) \in J$. The number of corresponding queried edges of $L \setminus L_1$ is at most $\deg_{L\setminus L_1}(a) + \deg_{L\setminus L_1}(b)$. Clearly again, for any fixed e we have $\mathbb{E}[\deg_{L\setminus L_1}(a)] \le \Delta p = \Delta^{0.15}$ and similarly for b. Thus, the expected number of queried entries of $\vec{x}$ corresponding to edge e is at most $4\Delta^{0.15}$.

Putting all these together, the expected number of queried entries of $\vec{x}$ is $O(\Delta^{1.15} + t\Delta^{0.15})$. ∎

Lemma 3. *We have $\mathbb{E}[\sum_{v\in V} B(v)] \le O(n\Delta^{1.15})$ where the expectation is taken over χ, L, π.*

Proof: For any vertex $v \in V$, let us first define $B'(v)$ to the number of edges in L_1 that are queried in the course of running $\mathcal{DO}_\pi(v)$. This can be bounded by:

$$B'(v) \le \sum_{u\in N_G(v)\cap V_1} \sum_{w:(u,w)\in L_1} A(u, w).$$

Summing over $v \in V$, we get:

$$\begin{aligned}\sum_v B'(v) &\le \sum_v \sum_{u\in N_G(v)\cap V_1} \sum_{w:(u,w)\in L_1} A(u, w)\\ &\le \sum_{(u,w)\in L_1} A(u, w)\Big(\sum_{v\in N_G(u)} 1 + \sum_{v\in N_G(w)} 1\Big)\\ &\le 2\Delta \sum_{e\in L_1} A(e).\end{aligned}$$

Taking expectations and applying Claim 5, we therefore have

$$\mathbb{E}\Big[\sum_v B'(v)\Big] \le 2\Delta\mathbb{E}\Big[\sum_{e\in L_1} A(e)\Big] \le O(\Delta n).$$

By Claim 6, we have $\mathbb{E}[B(v) \mid B'(v) = t] \le O(\Delta^{1.15} + t\Delta^{0.15})$ for any vertex v. This further implies that $\mathbb{E}[B(v)] \le O(\Delta^{1.15} + \mathbb{E}[B'(v)]\Delta^{0.15})$; thus

$$\begin{aligned}\mathbb{E}\Big[\sum_v B(v)\Big] &\le O\Big(\Delta^{0.15}\mathbb{E}\Big[\sum_v B'(v)\Big] + \Delta^{1.15}n\Big)\\ &\le O(\Delta^{1.15}n),\end{aligned}$$

as desired. ∎

We now say that a vertex v is *bad* if $\mathbb{E}_{\vec{x}}[B(v) \mid \pi, L] > \Delta^{1.4}$ (i.e., $\Omega(\Delta^{0.25})$ times larger than the average value given by Lemma 3) and *good* otherwise. Let us define $\mathcal{B}$ to be the set of bad vertices. Note that, because $\mathcal{B}$ is based on a conditional expectation, it is determined solely by the random variables π, L.

Claim 7. *The expected size of $\mathcal{B}$ satisfies $\mathbb{E}_{\pi,L}[|\mathcal{B}|] \le O(n/\Delta^{0.25})$.*

Proof: Observe that we have $\sum_{v\in V} \mathbb{E}_\chi[B(v) \mid \pi, L] \ge |\mathcal{B}| \cdot \Delta^{1.4}$ with probability one since for each bad vertex

$v \in \mathcal{B}$, by definition the expected value of $B(v)$ is at least $\Delta^{1.4}$. Taking expectations over π and L, we therefore get

$$\begin{aligned}\mathbb{E}_{\pi,L}[|\mathcal{B}|] &\leq \Delta^{-1.4} \cdot \mathbb{E}_{\pi,L}\Big[\sum_{v \in V} \mathbb{E}_{\vec{x}}[B(v)] \mid \pi, L]\Big] \\ &= \Delta^{-1.4} \sum_{v \in V} \mathbb{E}[B(v)].\end{aligned}$$

By Lemma 3, we have $\sum_{v \in V} \mathbb{E}[B(v)] \leq O(\Delta^{1.15} n)$. Putting these two bounds together gives $\mathbb{E}[|\mathcal{B}|] \leq O(n\Delta^{-0.25})$. ∎

Claim 8. *For any π, L, any good vertex v has* $\mathrm{Var}(Z_{v,1} \mid \pi, L) \leq O(\Delta^{1.4})$.

Proof: By Claim 4, for any vertex partitioning $\vec{x}$, we have $\sum_{w \in V}(Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)}))^2 \leq 4B(v)$, where $\vec{x}^{(w)}$ is obtained by changing the w entry of $\vec{x}$. If we fix π, L and take expectations over $\vec{x}$, this gives

$$\mathbb{E}_{\vec{x}}\Big[\sum_{w \in V}(Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)}))^2 \mid \pi, L\Big] \leq 4\mathbb{E}_{\vec{x}}[B(v) \mid \pi, L].$$

On the other hand, by (1), any vertex v has

$$\mathrm{Var}(Z_{v,1} \mid \pi, L) \leq \frac{1}{2}\mathbb{E}_{\vec{x}}\Big[\sum_{w \in V}\big(Z_{v,1}(\vec{x}) - Z_{v,1}(\vec{x}^{(w)})\big)^2 \mid \pi, L\Big].$$

Combining the two inequalities gives $\mathrm{Var}(Z_{v,1} \mid \pi, L) \leq 2\mathbb{E}[B(v) \mid \pi, L]$. Since v is good with respect to π, L, it satisfies $\mathbb{E}_{\vec{x}}[B(v) \mid \pi, L] \leq O(\Delta^{1.4})$ by definition. Thus $\mathrm{Var}(Z_{v,1} \mid \pi, L) \leq O(\Delta^{1.4})$. ∎

We are now ready to prove the main part of Lemma 2.

Proof of Lemma 2, part (3): For each vertex $v \in V$, define the random variable y_v to be the indicator function that $\deg^{\text{res}}_M(v) > \Delta^{0.99}$ after running Algorithm 1. We need to show that $\mathbb{E}[\sum_{v \in V} y_v] \leq O(n/\Delta^{0.03})$.

Depending on π and L, let us partition the vertices in V into two subsets $\mathcal{B}$ and $\mathcal{G}$ of respectively bad and good vertices as defined before. Furthermore, fix $\tau = 2\Delta^{0.86}$ and partition the set $\mathcal{G}$ of good vertices into two subsets $\mathcal{H}$ and $\mathcal{L}$ where for any vertex $v \in \mathcal{H}$, $\mathbb{E}_{\vec{x}}[Z_{v,1} \mid \pi, L] \geq \tau$ and for any $v \in \mathcal{L}$, $\mathbb{E}_{\vec{x}}[Z_{v,1} \mid \pi, L] < \tau$. We have:

$$\sum_{v \in V} y_v = \sum_{v \in \mathcal{B}} y_v + \sum_{v \in \mathcal{L}} y_v + \sum_{v \in \mathcal{H}} y_v.$$

By Claim 7, we know directly that $\mathbb{E}[|\mathcal{B}|] \leq O(n/\Delta^{0.25})$. Since $y_v \leq 1$ for any vertex v, we have $\mathbb{E}[\sum_{v \in \mathcal{B}} y_v] \leq \mathbb{E}[|\mathcal{B}|] \leq O(n/\Delta^{0.25})$.

Now, for any fixed $v \in V$, we compute the probability of the event that $v \in \mathcal{L}$ and $y_v = 1$ (respectively, $v \in \mathcal{H}$ and $y_v = 1$); we show that each such event has probability $O(\Delta^{-0.03})$.

Good vertices of type $\mathcal{L}$: Recall that $\deg^{\text{res}}_M(v) \leq Z_{v,1} + \ldots + Z_{v,k}$ where $k = \Delta^{0.1}$ denotes the number of partitions. Taking expectations we get

$$\begin{aligned}\mathbb{E}[\deg^{\text{res}}_M(v) \mid \pi, L] &\leq \mathbb{E}[Z_{v,1} + \ldots + Z_{v,k} \mid \pi, L] \\ &= k\mathbb{E}[Z_{v,1} \mid \pi, L]\end{aligned}$$

where the latter equality for symmetry of the partitions. If $v \in \mathcal{L}$, then $\mathbb{E}[Z_{v,1} \mid \pi, L] < \tau$, thus, $\mathbb{E}[\deg^{\text{res}}_M(v) \mid \pi, L] \leq k\tau = \Delta^{0.1} \cdot 2\Delta^{0.86} = 2\Delta^{0.96}$. By Markov's inequality, $\Pr[\deg^{\text{res}}_M(v) > \Delta^{0.99} \mid \pi, L] < O(\Delta^{-0.03})$. Therefore, $\Pr[y_v = 1 \wedge v \in \mathcal{L} \mid \pi, L] \leq O(\Delta^{-0.03})$. Integrating over π, L also $\Pr[y_v = 1 \wedge v \in \mathcal{L}] \leq O(\Delta^{-0.03})$ as desired.

Good vertices of type $\mathcal{H}$: We show that good vertices of type $\mathcal{H}$ are highly likely to be matched in their own partition and thus not too many of them will remain in the graph. For such a vertex v, one of the following two events must occur: either $Z'_v \geq \Delta^{0.86}$ or $Z'_v < \Delta^{0.86}$. The first of these events has probability $\exp(-\operatorname{poly}(\Delta)) \ll \Delta^{-0.03}$ by Claim 2. We next need to bound the probability of having $v \in \mathcal{H}$ and also having $Z'_v \leq \Delta^{0.86}$. If this occurs, by definition of Z'_v, we have at least one index $j \in [k]$ with $Z_{v,j} < \Delta^{0.86}$. We bound the occurrence probability of this event.

Since $v \in \mathcal{H}$, by definition it is a good vertex and thus Claim 8 shows that $\mathrm{Var}(Z_{v,i} \mid \pi, L) \leq O(\Delta^{1.4})$. Also, $\mathbb{E}[Z_{v,i} \mid \pi, L] \geq 2\Delta^{0.86}$. Therefore, by Chebyshev's inequality, for any fixed $i \in [k]$,

$$\begin{aligned}&\Pr\big[Z_{v,i} < \Delta^{0.86} \mid \pi, L\big] \\ &\quad\leq \Pr\Big[\big|Z_{v,i} - \mathbb{E}[Z_{v,i} \mid \pi, L]\big| \geq 2\Delta^{0.86} - \Delta^{0.86}\Big] \\ &\quad\leq O\left(\frac{\mathrm{Var}(Z_{v,i} \mid \pi, L)}{(\Delta^{0.86})^2}\right) \leq O\left(\frac{\Delta^{1.4}}{\Delta^{1.72}}\right) \leq O(\Delta^{-0.32}).\end{aligned}$$

By a union bound over the $k = \Delta^{0.1}$ choices of j, we can bound the probability that v is in $\mathcal{H}$ and there exists some $j \in [k]$ with $Z_{v,j} \leq \Delta^{0.86}$ given π and L by $O(\Delta^{-0.22})$. This means that overall, the probability that $y_v = 1$ and $v \in \mathcal{H}$ is $O(\Delta^{-0.22}) \ll O(\Delta^{-0.03})$. ∎

V. Putting Everything Together

We now prove Lemma 1, showing that Algorithm 1 can be used to reduce the overall graph degree. There are two parts to doing this. First, we need to amplify the success probability of Lemma 2, which only showed a degree reduction in expectation, into one holding w.e.h.p. Next, we need to remove the remaining high-degree vertices.

Claim 9. *There is an algorithm to generate a matching M which w.e.h.p. uses $n/\Delta^{\Omega(1)}$ space per machine and $O(m)$ total space, such that there are at most $n/\Delta^{0.02}$ vertices v with $\deg^{res}_M(v) > \Delta^{0.99}$.*

Proof: We may assume that the original graph has as least $n/\Delta^{0.02}$ vertices with $\deg(v) > \Delta^{0.99}$, as otherwise there is nothing to do. This implies that $m \geq n\Delta^{0.97}$.

Now consider running Algorithm 1 to generate a matching M. Let us define Y to be the number of vertices $v \in V$

with $\deg^{\text{res}}_M(v) > \Delta^{0.99}$. Lemma 1 has shown that $\mathbb{E}[Y] \leq O(n/\Delta^{0.03})$, and so we need to show concentration for Y. There are two cases depending on the size of Δ.

Case 1: $\Delta > n^{0.1}$: In this case, Markov's inequality applied to Y shows that $\Pr[Y > n\Delta^{-0.02}] \leq O(\Delta^{-0.01}) \leq 1/2$. Now consider running $t = n^a$ parallel iterations of Algorithm 1 for some constant $a > 0$, generating matchings $M_1, \ldots, M_t$. Since they are independent, there is a probability of at least $1 - 2^{-t}$ that at least one matching M_i has the property that its residual set of high-degree vertices satisfies $Y > n\Delta^{-0.02}$. Thus, w.e.h.p., this algorithm satisfies the condition on the high-degree vertices. Each application of Algorithm 1 separately uses $O(n) + m/\operatorname{poly}(\Delta)$ space. Therefore, the t iterations in total use $O(n^{1+a}) + n^a m/\operatorname{poly}(\Delta)$ space. Since $\Delta > n^{0.1}$ and $m \geq \Delta n^{0.97} > n^{1.07}$, this is $O(m)$ for a a sufficiently small constant.

Case 2: $\Delta < n^{0.1}$: We can regard Y as being determined by $O(n\Delta)$ random variables, namely, the values ρ, χ, L. By Lemma 5, modifying each entry of ρ, χ, or L can only change the match-status of at most $O(1)$ vertices. Each of these, in turn, has only Δ neighbors, which are the only vertices whose degree in $G[V \setminus M]$ is changed. Thus, changing each of the underlying random variables can only change Y by $O(\Delta)$. By Proposition 3, therefore, w.e.h.p. we have

$$Y \leq \mathbb{E}[Y] + O(\Delta) n^{0.01} \sqrt{n\Delta} \leq O(n\Delta^{-0.03}) + O(n^{0.51}\Delta^{1.5}).$$

As $\Delta \leq n^{0.1}$ and Δ is larger than any neded constants, this is at most $n\Delta^{-0.02}$. Therefore, already a single application of Algorithm 1 suceeds w.e.h.p. ∎

Having slightly reduced the number of high-degree vertices, we next use the following Algorithm 2, which significantly decreases the number of high-degree vertices.

Algorithm 2.

(1) Let $\mathcal{Y}$ be the set of vertices in $G[V \setminus M]$ with degree greater than $\tau = \Delta^{0.999}$.

(2) Sample each edge with at least one end-point in $\mathcal{Y}$ with probability $q := \Delta^{-0.99}$ and let L be the set of sampled edges.

(3) Put $G^L = (V, L)$ in machine 1, choose an arbitrary permutation π over its edges and return matching $M' := \mathsf{GreedyMM}(G^L, \pi)$.

Claim 10. *Given a graph G, suppose we apply Claim 9; let M be the resulting matching and $G' = G[V \setminus M]$. Suppose we next run Algorithm 2 on G' and let M' denote the resulting matching. Let $\mathcal{Y}'$ denote the set of vertices v with $\deg^{res}_{M \cup M'}(v) > \tau$. Then, w.e.h.p., $|\mathcal{Y}'| \leq n/\Delta^{1.01}$.*

Proof: Let $\mathcal{Y}$ be the set of vertices with $\deg^{\text{res}}_M(v) > \tau$ and $Y = |\mathcal{Y}|$. By Claim 9, w.e.h.p. $Y \leq n/\Delta^{0.02}$. For the remainder of this proof, we assume that M (and hence Y) is fixed and it satisfies this bound.

We first analyze $\mathbb{E}[Y']$ where we define $Y' = |\mathcal{Y}'|$. Consider some vertex $v \in \mathcal{Y}$. By Lemma 4, with probability at least $1 - \beta$ the vertex v has $\deg^{\text{res}}_{M \cup M'} \leq O(\frac{\log 1/\beta}{q})$. Setting $\beta = e^{-\Delta^{0.001}}$, we get that $\deg^{\text{res}}_{M \cup M'}(v) \leq O(\Delta^{0.991})$ with probability $1 - \exp(-\Delta^{\Omega(1)})$. Since this holds for any vertex $v \in \mathcal{Y}$, we have shown that

$$E[Y'] \leq Y \cdot \exp(-\Delta^{\Omega(1)}) \leq n e^{-\Delta^{\Omega(1)}}.$$

We next need to show concentration for Y'. For this, note that if $\Delta > n^{0.01}$, then the above bound on $\mathbb{E}[Y']$ already implies (by Markov's inequality) that $Y' < 1$ w.e.h.p.

If $\Delta < n^{0.01}$, then we use the bounded differences inequality. Here, Y' can be regarded as a function of $n\Delta$ random variables (the membership of each edge in L). By Lemma 5, each such edge can affect the match-status of $O(1)$ vertices. Each such vertex w, in turn, can only change the membership in $\mathcal{Y}'$ of its neighbors. Hence, each random variable changes Y' by at most $O(\Delta)$. By Proposition 3, we therefore have w.h.p.

$$Y' \leq \mathbb{E}[Y'] + O(\Delta\sqrt{n\Delta} n^{0.01}) \leq n e^{-\Delta^{\Omega(1)}} + O(\Delta^{1.5} n^{0.51}).$$

By our assumption that $\Delta \leq n^{0.01}$, this is easily seen to be smaller than $n/\Delta^{1.01}$. ∎

Proof of Lemma 1: When we apply Claim 9 and then apply Claim 10, this w.e.h.p. gives matchings M, M' respectively such that $G[V \setminus (M \cup M')]$ has at most $n/\Delta^{1.01}$ vertices of degree larger than $\Delta^{0.999}$. Claim 9 already obeys the stated space bounds. For Algorithm 2, observe that $|\mathcal{Y}| \leq n/\Delta^{0.02}$, and so there are at most $n\Delta^{0.98}$ edges incident to $\mathcal{Y}$. This means $\mathbb{E}[|L|] \leq n/\Delta^{0.01}$ and a simple Chernoff bound thus shows that $L \leq n/\Delta^{\Omega(1)}$ w.e.h.p.

Finally, we place all vertices with degree at least $\Delta^{0.999}$ and their incident edges onto a single machine; this clearly takes $O(n/\Delta^{0.01})$ space. Since Δ is larger than any needed constant, this is at most $n/\Delta^{\Omega(1)}$. We thus expand $M \cup M'$ to a maximal matching M'' of $G[V \setminus (M \cup M')]$. At the end of this process, all remaining vertices of G must have degree less than $\Delta^{0.999}$. ∎

A. The Algorithm with Mildly Sublinear Space

We now turn to proving Theorem 2, where we reduce the space per machine to $n/2^{\Omega(\sqrt{\log n})}$ with round complexity at $O(\log\log\Delta + \log\log\log n)$. The follows by combining the fact that our algorithms require $n/\Delta^{\Omega(1)}$ space with a known technique for simulating LOCAL algorithm on low-degree graphs in an exponentially faster time.

Proof of Theorem 2: The degree reduction algorithm of Lemma 1 uses a space per machine of $n/\Delta^{\Omega(1)}$ to get the degree down from Δ to $\Delta^{1-\Omega(1)}$. Therefore, if $\Delta \geq 2^{\Omega(\sqrt{\log n})}$,

the degree reduction automatically requires $n/2^{\Omega(\sqrt{\log n})}$ space. This means that within $O(\log\log\Delta)$ rounds, we can get the maximum degree down to $\Delta' \leq 2^{(\gamma/2)\sqrt{\log n}}$ w.e.h.p., where $\gamma \in (0,1)$ is any small constant. (If $\Delta \leq 2^{(\gamma/2)\sqrt{\log n}}$ originally, then we simply have $\Delta' = \Delta$.)

At this point, we switch to a different algorithm: we simulate the known $t = O(\log\Delta + \text{poly}(\log\log n))$ round LOCAL algorithms for maximal matching [9]. This requires $O(\log t) = O(\log\log\Delta + \log\log\log n)$ rounds; this is possible for the all-to-all communication of the machines (compared to LOCAL) and the fact that the maximum degree is small (so that the neighborhood is not too large that it does not fit the memory). For more details, see for example the *blind coordination lemma* of [16] which shows one can simulate t rounds of all *state-congested* LOCAL algorithms (such as that of [9]) in $O(\frac{t}{\log_\Delta n} + \log t)$ rounds of MPC with $n^{1-\Omega(1)}$ space per machine and $O(n^{1+\gamma/2})$ total space. (This excludes the space needed to store the original graph.)

In our case, since we apply this algorithm to a graph of maximum degree $\Delta' = 2^{(\gamma/2)\sqrt{\log n}}$ and $t = O(\log\Delta' + \text{poly}(\log\log n))$, and we get a runtime of

$$O\Big(\frac{\log\Delta' + \text{poly}\log\log n}{\log_{\Delta'} n} + \log\log\Delta' + \log\log\log n\Big).$$

Since $\Delta' \leq 2^{O(\sqrt{\log n})}$, the first term is $O(1)$; since $\Delta' \leq \Delta$, the second term is at most $\log\log\Delta$.

Note that the LOCAL maximal matching algorithm we are simulating here only succeeds with high probability, i.e. with probability $1 - 1/\text{poly}(n)$. In order to amplify it to exponential success probability, we can run $n^{\gamma/2}$ separate independent executions; w.e.h.p., at least one will succeed. This multiplies the total space by $n^{\gamma/2}$, bringing the total space (aside from the storage of G) up to $n^{1+\gamma}$. ■

VI. Useful Properties of Sequential Greedy Maximal Matching

In this section we prove the properties of the sequential greedy maximal matching that we used throughout the paper.

The first property concerns the behavior of greedy matching when it is run on an edge-sampled subgraph of a graph. This property is very similar to some results in [1, 21, 27] on greedy algorithms for correlation clustering and maximal independent set.

Lemma 4. *Fix a graph $G = (V, E)$, let π be a permutation over E, and let $p \in (0,1]$ be an arbitrary parameter. We define $G_p = (V, E_p)$ to be the random subgraph of G formed wherein each edge in E appears in E_p independently with probability p and define $M := \textsf{GreedyMM}(G_p, \pi)$. For any vertex v and any parameter $\beta \in (0, 1/2)$, with probability at least $1 - \beta$, $\deg^{res}_M(v) \leq \frac{\ln(1/\beta)}{p}$.*

Proof: Consider the following equivalent method of generating M. We iterate over the edges in E in the order of π. Upon visiting an edge e, if one of its incident edges belongs to M, we call it *irrelevant* and discard it. Otherwise, we draw a Bernoulli-p random variable X_e; if $X_e = 1$, we call e *lucky* and add it to M otherwise we call e *unlucky*.

If v is matched in M, then $\deg^{\text{res}}_M(v) = 0$. Otherwise, all of its remaining edges in $G[V \setminus M]$ should have been unlucky. That is, every time we encounter an edge e in this process, it must have been irrelevant or we must have chosen $X_e = 0$. Furthermore, in order to have $\deg^{\text{res}}_M(v) > \tau = \frac{\ln(1/\beta)}{p}$, there must remain at least τ edges which are not irrelevant. During this process, the probability that all such edges are marked unlucky is at most $(1-p)^\tau \leq \exp(-\tau p) = \beta$. ■

The second useful property of the greedy matching is that modifying a single vertex or edge of G does not change the set of matched vertices too much. Note that that the set of *edges* selected for M can change significantly.

Lemma 5. *Fix some graph $G(V, E)$ and let $\rho : E \to [0,1]$ be an associated list of priorities:*

1) *If graph G' is derived by removing a vertex of G, then there are at most 2 vertices whose match-status differs in $\textsf{GreedyMM}(G, \rho)$ and $\textsf{GreedyMM}(G', \rho)$.*
2) *If graph G' is derived by removing an edge of G, then there are at most 2 vertices whose match-status differs in $\textsf{GreedyMM}(G, \rho)$ and $\textsf{GreedyMM}(G', \rho)$.*
3) *If ρ' is derived by changing a single entry of ρ, then there are at most 2 vertices whose match-status differs in $\textsf{GreedyMM}(G, \rho)$ and $\textsf{GreedyMM}(G, \rho')$.*

Proof: We start with the proof of the first part. Suppose that G' is obtained by removing some vertex v from G. Let $M := \textsf{GreedyMM}(G, \rho)$ and $M' := \textsf{GreedyMM}(G', \rho)$. Furthermore, let $D := M \oplus M'$ denote the symmetric difference of M and M', i.e. the set $(M \setminus M') \cup (M' \setminus M)$. Note that the match-status of a vertex v differs in M and M' if and only if its degree in D is one. Therefore, it suffices to show that there are at most two such vertices in D.

We first claim that D has at most one connected component (apart from singleton vertices). For sake of contradiction, suppose that D has multiple such connected components; fix one component C that does not contain v. Let e be the edge in C with the highest priority. The fact that no higher priority edge that is connected to e is part of M or M' (otherwise e would not be the highest priority edge in C) shows that e has to belong to both M and M'. By definition of D, this means that $e \notin D$ which is a contradiction. Next, observe that since D is composed of the edges of two matchings, its maximum degree is at most 2 and thus its unique component is either a path or a cycle. The latter has no vertex of degree one and the former has two; proving part 1 of Lemma 5.

The proof of the other two parts of Lemma 5 follows from a similar argument. If an edge e is removed from G or its entry in ρ is changed, then for the same argument, the symmetric difference $M \oplus M'$ of the two greedy matchings M and M' that are obtained would contain only one

connected component which has to contain e. Since this component is a cycle or a path, the match-statuses of at most two vertices are different in the two matchings. ■

The third property is the most subtle: it can be summarized as stating that the presence of any given edge e appearing in M can be determined from a relatively small number of other edges. To make this more precise, let us consider the following query-based method which we refer to as the "edge oracle" $\mathcal{EO}_\pi(e)$ for determining whether an edge e appears in $\mathsf{GreedyMM}(G, \pi)$:

$\mathcal{EO}_\pi(e)$: A query-process to determine whether $e \in \mathsf{GreedyMM}(G, \pi)$.

Let $e_1, \ldots, e_d$ be the incident edges to e in G sorted such that $\pi(e_1) < \pi(e_2) < \cdots < \pi(e_d)$.
for $i = 1, \ldots, d$ **do**
  if $\pi(e_i) < \pi(e)$ **then**
    if $\mathcal{EO}_\pi(e_j) =$ YES **then return** NO
return YES

It is clear that $e \in \mathsf{GreedyMM}(G, \pi)$ if and only if $\mathcal{EO}_\pi(e) =$ YES. For any edge $e \in E$, let us define $A(e)$ to be the number of (recursive) calls to $\mathcal{EO}_\pi$ that are generated by running $\mathcal{EO}_\pi(e)$. Translating a result of Yoshida *et al.* [37] for maximal independent set into our context, gives:

Proposition 4 ([37])**.** *Let G be a graph with m edges and r pairs of intersecting edges. If π is drawn u.a.r. from permutations on m elements, then $\mathbb{E}_\pi[\sum_{e \in E} A(e)] \leq O(m + r)$.*

Proof: Let H be the line graph of G. Then H has m vertices and r edges, and hence has average degree $2r/m$. Also, $\mathbb{E}_\pi[A(e)]$ is the expected query complexity of the maximal independent set of H under a random sequential greedy independent set. The result of Yoshida *et al.* [37, Theorem 2.1] implies the following bound on the average value of $A(e)$ in terms of the average degree of H:

$$\frac{1}{m}\mathbb{E}_\pi\Big[\sum_{e \in L} A(e)\Big] \leq 1 + \frac{2r}{m}.$$

We obtain the stated result by multiplying through by m. ■

References

[1] Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, pages 2237–2246, 2015.

[2] Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 202–211, 2015.

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

[4] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proceedings of the 46th annual ACM Symposiim on Theory of Computing (STOC)*, pages 574–583, 2014.

[5] Sepehr Assadi. Simple round compression for parallel vertex cover. *CoRR*, abs/1709.04599, 2017.

[6] Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein. Coresets meet edcs: algorithms for matching and vertex cover on massive graphs. In *Proceedings of the 30th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1616–1635, 2019.

[7] Sepehr Assadi and Sanjeev Khanna. Randomized composable coresets for matching and vertex cover. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 3–12, 2017.

[8] Sepehr Assadi, Xiaorui Sun, and Omri Weinstein. Massively parallel algorithms for finding well-connected components in sparse graphs. *CoRR*, abs/1805.02974, 2018.

[9] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The Locality of Distributed Symmetry Breaking. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 321–330, 2012.

[10] MohammadHossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab S. Mirrokni. Affinity clustering: Hierarchical clustering at scale. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS)*, pages 6867–6877, 2017.

[11] Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 273–284, 2013.

[12] Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. *J. ACM*, 64(6):40:1–40:58, 2017.

[13] Soheil Behnezhad, Sebastian Brandt, Mahsa Derakhshan, Manuela Fischer, MohammadTaghi Hajiaghayi, Richard M. Karp, and Jara Uitto. Massively parallel computation of matching and MIS in sparse graphs. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019.*, pages 481–490, 2019.

[14] Soheil Behnezhad, Mahsa Derakhshan, Hossein Esfandiari, Elif Tan, and Hadi Yami. Brief announcement: Graph matching in massive datasets. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 133–136, 2017.

[15] Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Brief announcement: Semi-mapreduce meets congested clique. *CoRR*, abs/1802.10297, 2018.

[16] Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, and Richard M. Karp. Massively parallel symmetry breaking on sparse graphs: MIS and maximal matching. *CoRR*, abs/1807.06701, 2018.

[17] Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Marina Knittel, and Hamed Saleh. Streaming and massively parallel algorithms for edge coloring. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany.*, pages 15:1–15:14, 2019.

[18] Artur Czumaj, Jakub Lacki, Aleksander Madry, Slobodan Mitrovic, Krzysztof Onak, and Piotr Sankowski. Round compression for parallel matching algorithms. In *Proceedings of the 50th annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 471–484, 2018.

[19] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[20] Buddhima Gamlath, Sagar Kale, Slobodan Mitrović, and Ola Svensson. Weighted matchings via unweighted augmentations. *arXiv preprint arXiv:1811.02760*, 2018.

[21] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. Improved massively parallel computation algorithms for MIS, matching, and vertex cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed (PODC)*, pages 129–138, 2018.

[22] Mohsen Ghaffari and Jara Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. *Proceedings of the 30th annual ACM-SIAM Symposium on Discrete Algorithms (SODA), to appear.*, 2019.

[23] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the MapReduce framework. In *Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC)*, pages 374–383, 2011.

[24] Amos Israeli and Alon Itai. A fast and simple randomized parallel algorithm for maximal matching. *Inf. Process. Lett.*, 22(2):77–80, 1986.

[25] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *Proceedings of the 21st annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 938–948, 2010.

[26] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[27] Christian Konrad. Mis in the congested clique model in $O(\log\log\Delta)$ rounds. *CoRR*, abs/1802.07647, 2018.

[28] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in MapReduce. In *Proceedings of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 85–94, 2011.

[29] Christoph Lenzen. Optimal deterministic routing and sorting on the congested clique. In *ACM Symposium on Principles of Distributed Computing, PODC*, pages 42–50, 2013.

[30] Zvi Lotker, Boaz Patt-Shamir, and Adi Rosén. Distributed approximate matching. *SIAM J. Comput.*, 39(2):445–460, 2009.

[31] Michael Luby. A simple parallel algorithm for the maximal independent set problem. In *Proceedings of the 17th annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1985.

[32] Andrew McGregor. Finding graph matchings in data streams. In *8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX) and 9th International Workshop on Randomization and Computation (RANDOM)*, pages 170–181, 2005.

[33] Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *Proceedings of the 49th annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 327–336, 2008.

[34] Krzysztof Onak. Round compression for parallel graph algorithms in strongly sublinear space. *CoRR*, abs/1807.08745, 2018.

[35] J Michael Steele et al. An Efron-Stein inequality for nonsymmetric statistics. *The Annals of Statistics*, 14(2):753–758, 1986.

[36] Tom White. *Hadoop - The Definitive Guide: Storage and Analysis at Internet Scale (2. ed.)*. O'Reilly, 2011.

[37] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In *Proceedings of the 41st annual ACM Symposium on Theory of Computing (STOC)*, pages 225–234, 2009.

[38] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster Computing with Working Sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2010.