Round Compression for Parallel Matching Algorithms

Artur Czumaj* University of Warwick
Jakub Łącki† Google Research, New York
Aleksander Mądry‡ MIT
Slobodan Mitrović§ EPFL
Krzysztof Onak¶ IBM Research
Piotr Sankowski‖ University of Warsaw

July 2017

Abstract

The last decade has witnessed the success of a number of *massive parallel computation* (MPC) frameworks, such as MapReduce, Hadoop, Dryad, or Spark. These frameworks allow for much more local memory and computation, compared to the classical distributed or PRAM models. In this context, we investigate the complexity of one of the most fundamental graph problems: *Maximum Matching*. We show that if the memory per machine is $\Omega(n)$ (or even only $n/(\log n)^{O(\log\log n)}$), then for any fixed constant $\epsilon > 0$, one can compute a $(2+\epsilon)$-approximation to Maximum Matching in $O\left((\log\log n)^2\right)$ MPC rounds. This constitutes an exponential improvement over previous work—both the one designed specifically for MPC and the one obtained via connections to PRAM or distributed algorithms—which required $\Theta(\log n)$ rounds to achieve similar approximation guarantees.

The starting point of our result is a (distributed) algorithm that computes an $O(1)$-approximation in $O(\log n)$ parallel phases. Its straightforward MPC implementation requires $\Theta(\log n)$ rounds. Our exponential speedup stems from compressing several phases of a modified version of this algorithm into a constant number of MPC rounds. We obtain this via a careful analysis of controlled randomness, which enables the emulation of multiple phases on separate machines without any coordination between them.

We leave it as an intriguing open question whether a similar speedup can be obtained for other PRAM and distributed graph algorithms.

*a.czumaj@warwick.ac.uk
†jlacki@google.com
‡madry@mit.edu
§slobodan.mitrovic@epfl.ch
¶konak@us.ibm.com
‖sank@mimuw.edu.pl

1 Introduction

Over the last decade, massive parallelism became a major paradigm in computing, and we have witnessed the deployment of a number of very successful massively parallel computation frameworks, such as MapReduce [DG04, DG08], Hadoop [Whi12], Dryad [IBY+07], or Spark [ZCF+10]. This paradigm and the corresponding models of computation are rather different from classical parallel algorithms models considered widely in literature, such as the PRAM model. In particular, in this paper, we study the *Massive Parallel Computation (MPC)* model (also known as Massively Parallel Communication model) that was abstracted out of capabilities of existing systems, starting with the work of Karloff, Suri, and Vassilvitskii [KSV10, GSZ11, BKS13, ANOY14, BKS14]. The main difference between this model and the PRAM model is that it allows for much more local storage (polynomial in the input size) and much more (in principle, unbounded) local computation. This enables it to capture a more "coarse–grained," and thus, potentially, more meaningful aspect of parallelism. It is often possible to simulate one clock step of PRAM in a constant number of rounds on MPC [KSV10, GSZ11]. This implies that algorithms for the PRAM model usually give rise to MPC algorithms without incurring any asymptotic blow up in the number of parallel rounds. As a result, a vast body of work on PRAM algorithms naturally translates to the new model.

It is thus natural to wonder: Are the MPC parallel round bounds "inherited" from the PRAM model tight? In particular, which problems can be solved in significantly *smaller* number of MPC rounds than what the lower bounds established for the PRAM model suggest?

It is not hard to come up with an example of a problem for which indeed the MPC parallel round number is much smaller than its PRAM round complexity. For instance, computing the parity of n Boolean values takes only $O(1)$ parallel rounds in the MPC model when space per machine is $n^{\Omega(1)}$, while on PRAM it provably requires $\Omega(\log n/\log\log n)$ time [BH87] (as long as the total number of processors is polynomial). However, the answer is typically less obvious for other problems. This is particularly the case for graph problems, whose study in a variant of the MPC model was initiated already by Karloff et al. [KSV10].

In this paper, we focus on one such problem, which is also one of the most central graph problems both in sequential and parallel computations: maximum matching. Maximum matchings have been the cornerstone of algorithmic research since 1950s and their study inspired many important ideas, including the complexity class P [Edm65]. In the PRAM model we can compute $1-\epsilon$ approximate matching in $O(\log n)$ rounds [LPP15] using randomization. Deterministically, a $2+\epsilon$ approximation can be computed in $O\left(\log^2 n\right)$ rounds [FG17]. We note that these results hold in a distributed message passing setting, where processors are located at graph nodes and can communicate only with neighbors. In such a distributed setting $\Omega\left(\sqrt{\log n/\log\log n}\right)$ time lower bound is known for computing any constant approximation to maximum matching [KMW06].

So far, in the MPC setting, the only prior results are due to Lattanzi, Moseley, Suri, and Vassilvitskii [LMSV11] and Ahn and Guha [AG15]. Lattanzi et al. [LMSV11] put forth algorithms for several graph problems, such as connected components, minimum spanning tree, and maximum matching problem, that were based on a so-called *filtering technique*. In particular, using this technique, they have obtained an algorithm that can compute a 2-approximation to maximum matching in $O(1/\delta)$ MPC rounds, provided S, the space per machine, is significantly larger than the total number of vertices n, that is $S=\Omega\left(n^{1+\delta}\right)$, for any constant $\delta\in(0,1)$. Later on, Ahn and Guha [AG15] provided an improved algorithm that computes a $(1+\epsilon)$-approximation in $O(1/(\delta\epsilon))$ rounds, provided $S=\Omega\left(n^{1+\delta}\right)$, for any constant $\delta>0$. Both these results, however, crucially require that space per machine is significantly superlinear in n, the number of vertices. In fact, if the space S is linear in n, which is a very natural setting for massively parallel graph

algorithms, the performance of both these algorithms degrades to $O(\log n)$ parallel round complexity, which matches what was known for the PRAM model, i.e., when the space S of each machine is only *constant*.

We also note that the known PRAM maximal independent set algorithms [Lub86, ABI86] can be used to find a maximal matching (i.e., 2-approximation to maximum matching) in $O(\log n)$ MPC rounds as long as space per machine is at least $n^{\Omega(1)}$ (i.e., $S \geq n^c$ for some constant $c > 0$). We omit further details here, except mentioning that a more or less direct simulation of those algorithms is possible via an $O(1)$-round sorting subroutine [GSZ11].

The above results give rise to the following fundamental question: Can the maximum matching be (approximately) solved in $o(\log n)$ parallel rounds in $O(n)$ space per machine? The main result of this paper is an affirmative answer to that question. We show that, for any $S = \Omega(n)$, one can obtain an $O(1)$-approximation to maximum matching using $O\left((\log\log n)^2\right)$ parallel MPC rounds. So, not only do we break the existing $\Omega(\log n)$ barrier, but also provide an exponential improvement over the previous work. Our algorithm can also provide a $(2+\epsilon)$, instead of $O(1)$-approximation, at the expense of the number of parallel rounds increasing by a factor of $O(\log(1/\epsilon))$. Finally, our approach can also provide algorithms that have $o(\log n)$ parallel round complexity also in the regime of S being (mildly) sublinear. For instance, we obtain $O\left((\log\log n)^2\right)$ MPC rounds even if space per machine is $S = n/(\log n)^{O(\log\log n)}$. The exact comparison of our bounds with previous results is given in Table 1.2.

1.1 The model

In this work, we adopt a version of the model introduced by Karloff, Suri, and Vassilvitskii [KSV10] and refined in later works [GSZ11, BKS13, ANOY14]. We call it *massive parallel computation* (MPC), which is a mutation of the name proposed by Beame et al. [BKS13].

In the MPC model, we have m machines at our disposal and each of them has S words of space. Initially, each machine receives its share of the input. In our case, the input is a collection E of edges and each machine receives approximately $|E|/m$ of them.

The computation proceeds in *rounds*. During the round, each of the machines processes its local data without communicating with other machines. At the end of each round, machines exchange messages. Each message is sent only to a single machine specified by the machine that is sending the message. All messages sent and received by each machine in each round have to fit into the machine's local memory. Hence, their total length is bounded by S.[1] This in particular implies that the total communication of the MPC model is bounded by $m \cdot S$ in each round. The messages are processed by recipients in the next round.

At the end of the computation, machines collectively output the solution. The data output by each machine has to fit in its local memory. Hence again, each machine can output at most S words.

The range of values for S and m. If the input is of size N, one usually wants S sublinear in the N, and the total space across all the machines to be at least N—so the input fits onto the machines—and ideally not much larger. Formally, one usually considers $S \in \Theta\left(N^{1-\epsilon}\right)$, for some $\epsilon > 0$.

In this paper, the focus is on graph algorithms. If n is the number of vertices in the graph, the input size can be as large as $\Theta\left(n^2\right)$. Our parallel algorithm requires $\Theta(n)$ space per machine (or even slightly less), which is polynomially less than the size of the input for dense graphs.

[1] This for instance allows a machine to send a single word to $S/100$ machines or $S/100$ words to one machine, but not $S/100$ words to $S/100$ machines if $S = \omega(1)$, even if the messages are identical.

Communication vs. computation complexity. The main focus of this work is the number of (communication) rounds required to finish computation. Also, even though we do not make an effort to explicitly bound it, it is apparent from the design of our algorithms that every machines performs $O(S \operatorname{polylog} S)$ computation steps locally. This in particular implies that the overall work across all the machines is $O(rN \operatorname{polylog} S)$, where r is the number of rounds and N is the input size (i.e., the number of edges).

The total communication during the computation is $O(rN)$ words. This is at most $O\left(rn^2\right)$ words and it is known that computing a $(1+\epsilon)$-approximate matching in the message passing model with $\Theta(n)$ edges per player may require $\Omega\left(n^2/(1+\epsilon)^2\right)$ bits of communication [HRVZ15]. Since our value of r is $O\left((\log\log n)^2\right)$ when $\Theta(n)$ edges are assigned to each player, we lose a factor of $\tilde{\Theta}(\log n)$ compared to this lower bound if words (and vertex identifiers) have $\Theta(\log n)$ bits.

1.2 Our results

In our work, we focus on computing an $O(1)$-approximate maximum matching in the MPC model. We collect our results and compare to the previous work in Table 1.2. The table presents two interesting regimes

Source	Approx.	Space	Rounds	Remarks
[LMSV11]	2	$n^{1+\Omega(1)}$	$O(1)$	Maximal matching
		$O(n)$	$O(\log n)$	
[AG15]	$1+\epsilon$	$O\left(n^{1+1/p}\right)$	$O(p/\epsilon)$	$p > 1$
	2	$n^{\Omega(1)}$	$O(\log n)$	Maximal matching Simulate [Lub86, ABI86, II86]
here	$O(1)$	$O(n)$	$O\left((\log\log n)^2\right)$	$\epsilon \in (0, 1/2)$ $2 \le f(n) = O\left(n^{1/2}\right)$
	$2+\epsilon$		$O\left((\log\log n)^2 \cdot \log(1/\epsilon)\right)$	
	$O(1)$	$O(n)/f(n)$	$O\left((\log\log n)^2 + \log f(n)\right)$	
	$2+\epsilon$		$O\left((\log\log n)^2 + \log f(n)\right) \cdot \log(1/\epsilon)$	

Table 1: Comparison of our results for computing approximate maximum size matchings to the previous results for the MPC model.

for our algorithms. On the one hand, when the space per machine is $S = O(n)$, we obtain an algorithm that requires $O((\log\log n)^2)$ rounds. This is the first known algorithm that, with linear space per machine, breaks the $O(\log n)$ round barrier. On the other hand, in the mildly sublinear regime of space per machine, i.e., when $S = O(n/f(n))$, for some function $f(n)$ that is $n^{o(1)}$, we obtain an algorithm that still requires $o(\log n)$ rounds. This, again is the first such result in this regime. In particular, we prove the following result.

Theorem 1.1. *There exists an MPC algorithm that constructs an $O(1)$-approximation to maximum matching with $\Omega(1)$ probability in $O\left((\log\log n)^2 + \max\left(\log \frac{n}{S}, 0\right)\right)$ rounds, where $S = n^{\Omega(1)}$ is the amount of space on each machine.*

As a corollary, we obtain the following result that provides nearly 2-approximate maximum matching.

Corollary 1.2. *For any $\epsilon \in (0, \frac{1}{2})$, there exists an MPC algorithm that constructs a $(2+\epsilon)$-approximation to maximum matching with $99/100$ probability in $O\left((\log\log n)^2 + \max\left(\log \frac{n}{S}, 0\right)\right) \cdot \log(1/\epsilon)$ rounds, where $S = n^{\Omega(1)}$ is the amount of space on each machine.*

1.3 Related work

We note that there were efforts at modeling MapReduce computation [FMS$^+$10] before the work of Karloff et al. Also a recent work [RVW16] investigates the complexity of the MPC model.

In the *filtering* technique, introduced by Lattanzi et al. [LMSV11], the input graph is iteratively sparsified until it can be stored on a single machine. For the matching problem, the sparsification is achieved by first obtaining a small sample of edges, then finding a maximal matching in the sample, and finally removing all the matched vertices. Once a sufficiently small graph is obtained, a maximal matching is computed on a single machine. In the $S = \Theta(n)$ regime, the authors show that their approach reduces the number of edges by a constant factor in each iteration. Despite this guarantee, until the very last step, each iteration may make little progress towards obtaining even an approximate maximal matching, resulting in a $O(\log n)$ round complexity of the algorithm. Similarly, the results of Ahn and Guha [AG15] require $n^{1+\Omega(1)}$ space per machine to compute a $O(1)$-approximate maximum weight matching in a constant number of rounds and do not imply a similar bound for the case of linear space.

We note that the algorithm of Lattanzi et al. [LMSV11] cannot be turned easily into a fast approximation algorithm when space per machine is sublinear. Even with $\Theta(n)$ space, their method is able to remove only a constant fraction of edges from the graph in each iteration, so $\Omega(\log n)$ rounds are needed until only a matching is left. When $S = \Theta(n)$, their algorithm works as follows: sample uniformly at random $\Theta(n)$ edges of the graph, find maximal matching on the sampled set, remove the matched vertices, and repeat. We do not provide a formal proof here, but on the following graph this algorithm requires $\tilde{\Omega}(\log n)$ rounds, even to discover a constant factor approximation. Consider a graph consisting of t separate regular graphs of degree 2^i, for $0 \leq i \leq t-1$, each on 2^t vertices. This graph has $t2^t$ nodes and the algorithm requires $\tilde{\Omega}(t)$ rounds even to find a constant approximate matching. The algorithm chooses edges uniformly at random, and few edges are selected each round from all but the densest remaining subgraphs. Thus, it takes multiple rounds until a matching of significant size is constructed for sparser subgraphs. This example emphasizes the weakness of direct edge sampling and motivates our vertex sampling scheme that we introduce in this paper.

Similarly, Ahn and Gupta [AG15] build on the filtering approach of Lattanzi et al. and design a primal-dual method for computing a $(1-\epsilon)$-approximate weighted maximum matching. They show that each iteration of their distributed algorithm either makes large progress in the dual, or they can construct a large approximate matching. Regardless of their new insights, their approach is inherently edge-sampling based and does not break the $O(\log n)$ round complexity barrier when $S = O(n)$.

Despite the fact that MPC model is rather new, computing matching is an important problem in this model, as the above mentioned two papers demonstrate. This is further witnessed by the fact that the distributed and parallel complexity of maximal matching has been studied for many years already. The best deterministic PRAM maximal matching algorithm, due to Israeli and Shiloach [IS86], runs in $O\left(\log^3 n\right)$ rounds. Israeli and Itai [II86] gave a randomized algorithm for this problem that runs in $O(\log n)$ rounds. Their algorithm works as well in CONGEST, a distributed message-passing model with a processor assigned to each vertex and a limit on the amount of information sent along each edge per round. A more recent paper by Lotker, Patt-Shamir, and Pettie [LPP15] gives a $(1-\epsilon)$-approximation to maximum matching in $O(\log n)$ rounds also in the CONGEST model, for any constant $\epsilon > 0$. On the deterministic front, in the LOCAL model, which is a relaxation of CONGEST that allows for an arbitrary amount of data sent along each edge, a line of research initiated by Hańćkowiak, Karoński, and Panconesi [HKP01, HKP99] led to an $O\left(\log^3 n\right)$-round algorithm by Fischer and Ghaffari [FG17].

On the negative side, Kuhn, Moscibroda, and Wattenhofer [KMW06] showed that any distributed algo-

rithm, randomized or deterministic, when communication is only between neighbors requires $\Omega\left(\sqrt{\log n / \log\log n}\right)$ rounds to compute a constant approximation to maximum matching. This lower bound applies to all distributed algorithms that have been mentioned above. Our algorithm circumvents this lower bound by loosening the only possible assumption there is to be loosened: single-hop communication. In a sense, we assign subgraphs to multiple machines and allow multi-hop communication between nodes in each subgraph.

Finally, the ideas behind the peeling algorithm that is a starting point for this paper can be traced back to the papers of Israeli, Itai, and Shiloach [II86, IS86], which can be interpreted as matching high degree vertices first in order to reduce the maximum degree. A sample distributed algorithm given in a work of Parnas nad Ron [PR07] uses this idea to compute an $O(\log n)$ approximation for vertex cover. Their algorithm was extended by Onak and Rubinfeld [OR10] in order to provide an $O(1)$-approximation for vertex cover and maximum matching in a dynamic version of the problems. This was achieved by randomly matching high-degree vertices to their neighbors in consecutive phases while reducing the maximum degree in the remaining graph. This approach was further developed in the dynamic graph setting by a number of papers [BHI15, BHN16, BHN17, BCH16]. Ideas similar to those in the paper of Parnas and Ron [PR07] were also used to compute polylogarithmic approximation in the streaming model by Kapralov, Khanna, and Sudan [KKS14]. Our version of the peeling algorithm was directly inspired by the work of Onak and Rubinfeld [OR10] and features important modifications in order to make our analysis go through.

1.4 Overview of our techniques

The main technical contribution underlying our result is a new method of *round compression*. We transform a global multi-phase algorithm for maximum matching into an MPC algorithm with a very small number of rounds. The underlying idea is quite simple: we take multiple phases of a global multi-phase algorithm and run them on every single machine independently. This approach, however, has obvious challenges since the machines cannot communicate in a single round of the MPC algorithm. In particular, the algorithm may commit to some sub-optimally chosen matching on one machine that can later lead to a strong deficiency globally. As a result, the output of the MPC algorithm could strongly deviate from the guarantees of the original algorithm's output. To cope with these challenges, we introduce a carefully crafted emulation of the global multi-phase algorithm that uses controlled randomness to simulate multiple phases of the global algorithm on separate machines without any coordination between the machines, thus allowing for a small number of parallel MPC rounds.

In what follows, we discuss this approach in more detail.

Global algorithm. Our starting point is a sequential peeling algorithm `GlobalAlg` (see page 9), which is a modified version of an algorithm used by Onak and Rubinfeld [OR10]. The algorithm had to be significantly adjusted in order to make our later analysis of a parallel version possible.

The execution of `GlobalAlg` is divided into $\Theta(\log n)$ *phases*. In each phase, the algorithm first computes a set H of high-degree vertices. Then it selects a set F of vertices, which we call *friends*. Next the algorithm selects a matching $\widetilde{M}$ between H and F, using a simple randomized strategy. F is carefully constructed so that both F and $\widetilde{M}$ are likely to be of order $\Theta(|H|)$. Finally, the algorithm removes all vertices in $H \cup F$, hence reducing the maximum vertex degree in the graph by a constant factor, and proceeds to the next phase. The central property of `GlobalAlg` is that it returns an $O(1)$-multiplicative approximation to Maximum Matching with constant probability (Corollary 2.4). A full description of `GlobalAlg` is given in Section 2.

Vertex based sampling. All the previous works on the distributed maximal matching problem known to the authors ([LMSV11], [AG15], [II86], simulation of [Lub86, ABI86]) process the input graph by sampling a subset of edges and performing the computation on the sampled subset only. These kinds of approaches, however, tend to select many "unnecessary" edges incident to high degree vertices while picking only very few edges from a hidden large matching. This tendency appears to be a fundamental barrier in reducing the number of computation rounds needed to obtain a maximal matching. Intuitively, if there is a large "hidden" matching, then it may takes many rounds to "peel off" edges that do not contribute to the large matching size. The starting point of our new approach is alleviating this sampling issue by resorting to a more careful vertex based sampling. Specifically, at each round, we randomly partition the vertex set into sets $V_1, \ldots, V_m$ and consider induced graphs on those subsets independently. This sampling scheme allows for faster edge elimination and maximum vertex degree reduction. We show that throughout our local algorithm execution local vertex degrees on each induced subgraph correspond to global vertex degrees.

Parallel emulation of the global algorithm. The following two ways could be used to execute `GlobalAlg` in the MPC model: (1) place the whole graph on one machine, and trivially execute all the phases of `GlobalAlg` in a single round; or (2) simulate one phase of `GlobalAlg` in one MPC round while using $O(n)$ space per machine, by distributing vertices randomly onto machines (see Section 5.1 for details). However, each of these approaches has severe drawbacks. The first approach requires $\Theta(|E|)$ space per machine, which is likely to be prohibitive for large graphs. On the other hand, while the second approach uses $O(n)$ space, it requires $\Theta(\log n)$ rounds of MPC computation. We achieve the best of both worlds by showing how to emulate the behavior of *multiple phases* of `GlobalAlg` in a *single MPC round* with each machine using $O(n)$ space, thus obtaining an MPC algorithm requiring $o(\log n)$ rounds. More specifically, we show that it is possible to emulate the behavior of `GlobalAlg` in $O\left((\log\log n)^2\right)$ rounds with each machine using $O(n)$ space.

Before we provide more details about our parallel emulation of `GlobalAlg`, let us mention some of the main obstacles such an emulation encounters. First, at the beginning of every phase `GlobalAlg` has access to the full graph. Therefore, it can easily compute the set of heavy vertices H. Machines in our MPC algorithm use $O(n)$ space and thus have access only to a small subgraph of the input graph (when $|E| \gg n$). Second, the degrees of vertices can significantly change from phase to phase. After each phase, it is not clear how to select high degree vertices in the next phase without inspecting the entire graph again. Hence, one of the main challenges in designing a multi-phase emulation of `GlobalAlg` is to ensure that machines at the beginning of every phase can estimate *global* degrees of vertices well enough to identify the set of heavy vertices. This property is achieved using a few modifications to the algorithm.

Preserving randomness. Our algorithm partitions the vertex set into m disjoint subsets V_i by assigning each vertex independently and uniformly at random. Then the graph induced by each subset V_i is processed on a separate machine. Each machine finds a set of heavy vertices, H_i, by estimating the global degree of each vertex of V_i. It is not hard to argue (using a standard concentration bound) that there is enough randomness in the initial partition so that local degrees in each induced subgraph roughly correspond to the global degrees. Hence, after the described partitioning, sets H and $\bigcup_{i\in[m]} H_i$ have very similar properties. This observation crucially relies on the fact that initially the vertices are distributed *independently* and *uniformly* at random.

However, if one attempts to execute more than one phase without randomly reassigning vertices to sets, the remaining vertices are no longer distributed independently and uniformly at random. This can in principle lead to a situation in which there are many edges going between sets V_i in the partition, but

proportionally very few edges are present in graphs H_i. In other words, after inspecting the neighborhood of every vertex locally and making a decision based on it, the randomness of the initial random partition may significantly decrease. We now discuss in more detail how we preserve two crucial properties of our vertex assignments throughout the execution of multiple phases:

Independence: A key and counter-intuitive step in our approach is to *estimate* even *local degrees* of vertices (in contrast to computing them exactly). To implement this feature, we sample a small set of vertices on each machine, called *reference sets*, and use the set to estimate the local degrees. Very crucially, all the vertices that are used in computing a matching in one emulated phase (including the reference sets) are discarded at the end of the phase, even if they do not participate in the obtained matching. Intuitively, this way we secure an independent distribution of the vertices across the machines in the next phase.

We also note, without going into details, that obtaining full independence required modifying how the set of friends is selected, compared to the approach of Onak and Rubinfeld. In their approach, each heavy vertex selected one friend at random. This, however, introduces dependencies between vertices that have not been selected. In our `GlobalAlg`, every vertex selects itself as a friend independently and proportionally to the number of high degree vertices. The final properties of the obtained sets in either approach are very similar.

Uniformity: A very desired property in the task of emulating multiple phases of `GlobalAlg` is a uniform distribution of vertices across all the machines at every phase. For such a distribution, we know the expected number of neighbors of each desired type assigned to the same machine. Obtaining perfect uniformity seems difficult—if not impossible in our setting—and we therefore settle for *near* uniformity of vertex assignments. The probability of the assignment of each vertex to each machine is allowed to differ slightly from that in the uniform distribution. Initially, the distribution of each vertex is uniform and with every phase it can deviate more and more from the uniform distribution. We bound the rate of the decay with high probability and execute multiple rounds as long as the deviation from the uniform distribution is negligible. More precisely, in the execution of the entire parallel algorithm, the sufficiently uniform distribution is on average kept over $\Omega\left(\frac{\log n}{(\log\log n)^2}\right)$ phases of the emulation of `GlobalAlg`.

In order to achieve the near uniformity, we modify the procedure for selecting H, the set of high-degree vertices, is selected. Instead of a hard threshold on the degrees of vertices that are included in H as in the sequential algorithm, we randomize the selection by using a carefully crafted threshold function μ_H. This function specifies the probability with which a vertex is included in H. It takes as input the ratio of the vertex's degree to the current threshold and it smoothly transitions from 0 to 1 in the neighborhood of the original hard threshold (see Figure 1). The main intuition behind the introduction of this function is that we want to ensure that a vertex is *not* selected for H with almost the same probability, independently of the machine on which it resides. For a hard threshold, it could happen that due to the composition of reference sets, it becomes clear after even a single phase that a given vertex that has not been removed could not have been on a given machine, because it would have landed above the threshold and would have been removed. At this point the distribution is clearly no longer uniform.

Function μ_H has further useful properties that we extensively exploit in our analysis. We just note that in order to ensure near uniformity with high probability, we also have to ensure that each vertex is selected for F, the set of friends, with roughly the same probability on each machine.

1.5 Future challenges

We show a parallel matching algorithm in the MPC model by taking an algorithm that can be seen as a distributed algorithm in the so-called LOCAL model. This algorithm requires $\Theta(\log n)$ rounds and can be simulated in $\Theta(\log n)$ MPC rounds relatively easily with $n^{\Omega(1)}$ space per machine. We develop an approximate version of the algorithm that uses much fewer rounds by repeatedly compressing a superconstant number of rounds of the original algorithm to $O(1)$ rounds. It is a great question if a this kind of speedup can be obtained for other—either distributed or PRAM—algorithms.

As for the specific problem considered in this paper, an obvious question is whether our round complexity is optimal. We conjecture that there is a better algorithm that requires $O(\log \log n)$ rounds, the square root of our complexity. Unfortunately, a factor of $\log n$ in one of our functions (see the logarithmic factor in α, a parameter defined later in the paper) propagates to the round complexity, where it imposes a penalty of $\log \log n$.

Note also that as opposed to the paper of Onak and Rubinfeld [OR10], we do not obtain an $O(1)$-approximation to vertex cover. This stems from the fact that we discard so-called reference sets, which can be much bigger than the minimum vertex cover. This is unfortunately necessary in our analysis. Is there a way to fix this shortcoming of our approach?

Finally, we suspect that there is a simpler algorithm for the problem that avoids the intricacies of our approach and proceeds by simply greedily matching high degree vertices on induced subgraphs without a sophisticated sampling in every phase. Unfortunately, we do not know how to analyze this kind of approach.

1.6 Paper organization

We start by giving a description of `GlobalAlg` in Section 2. Section 3 describes our emulation of a single phase of `GlobalAlg` in the MPC model. Section 4 gives and analyzes our parallel algorithm by putting together components developed in the previous sections. Section 5 describes additional implementation details in the MPC model.

1.7 Notation

Neighbor set. In our algorithms and analysis, we write $N(v)$ to denote the set of neighbors of a vertex v in the input graph.

Induced subgraphs. For a graph $G = (V, E)$ and $V' \subseteq V$, we write $G[V']$ to denote the subgraph of G induced by V'. Formally, $G[V'] \stackrel{\text{def}}{=} (V', E \cap (V' \times V'))$.

Concise range notation. Multiple times throughout a paper, we want to denote a range around some value. Instead of writing, say, $[x - \delta, x + \delta]$, we introduce a more concise notation. In this specific case, we would simply write $[\![x \pm \delta]\!]$. More formally, let E be a numerical expression that apart from standard operations also contains a single application of the binary or unary operator $\pm$. We create two standard numerical expressions from E: E_- and E_+ that replace $\pm$ with $-$ and $+$, respectively. Now we define $[\![E]\!] \stackrel{\text{def}}{=} [\min\{E_-, E_+\}, \max\{E_-, E_+\}]$.

As another example, consider $E = \sqrt{101 \pm 20}$. We have $E_- = \sqrt{101 - 20} = 9$ and $E_+ = \sqrt{101 + 20} = 11$. Hence $\left[\!\left[\sqrt{101 \pm 20}\right]\!\right] = [\min\{9, 11\}, \max\{9, 11\}] = [9, 11]$.

2 Global Algorithm

2.1 Overview

The starting point of our result is a peeling algorithm `GlobalAlg` that takes as input a graph G, and removes from it vertices of lower and lower degree until no edge is left. See Algorithm 1 for its pseudocode. We use the term *phase* to refer to an iteration of the main loop in Lines 2–6.

Each phase is associated with a threshold Δ. Initially, Δ equals $\widetilde{\Delta}$, the upper bound on the maximum vertex degree. In every phase, Δ is divided by two until it becomes less than one and the algorithm stops. Since during the execution of the algorithm we maintain the invariant that the maximum degree in the graph is at most Δ, the graph has no edge left when the algorithm terminates.

In each phase the algorithm matches, in expectation, a constant fraction of the vertices it removes. We use this fact to prove that, across all the phases, the algorithm computes a constant-factor approximate matching.

Algorithm 1: `GlobalAlg`$(G, \widetilde{\Delta})$
Global matching algorithm

Input: Graph $G = (V, E)$ of maximum degree at most $\widetilde{\Delta}$
Output: A matching in G

1 $\Delta \leftarrow \widetilde{\Delta}$, $M \leftarrow \emptyset$, $V' \leftarrow V$
2 **while** $\Delta \geq 1$ **do**
  `/* Invariant: the maximum degree in` $G[V']$ `is at most` Δ `*/`
3 Let $H \subset V'$ be a set of vertices of degree at least $\Delta/2$ in $G[V']$. We call vertices in H *heavy*.
4 Create a set F of *friends* by selecting each vertex $v \in V'$ independently with probability $|N(v) \cap H|/4\Delta$.
5 Compute a matching $\widetilde{M}$ in $G[H \cup F]$ using `MatchHeavy`(H, F) and add it to M.
6 $V' \leftarrow V' \setminus (H \cup F)$, $\Delta \leftarrow \Delta/2$
7 **return** M

Algorithm 2: `MatchHeavy`(H, F)
Computing a matching in $G[H \cup F]$

Input: set H of heavy vertices and set F of friends
Output: a matching in $G[H \cup F]$

1 For every vertex $v \in F$ pick uniformly at random a heavy neighbor $v_\star$ in $N(v) \cap H$.
2 Independently at random color each vertex in $H \cup F$ either red or blue.
3 Select the following subset of edges: $E_\star \leftarrow \{(v, v_\star) : v \in F \wedge v \text{ is red} \wedge v_\star \in H \wedge v_\star \text{ is blue}\}$.
4 For every blue vertex w incident to an edge in $E_\star$, select one such edge and add it to $\widetilde{M}$.
5 **return** $\widetilde{M}$

We now describe in more detail the execution of each phase. First, the algorithm creates H, the set of vertices that have degree at least $\Delta/2$ (Line 3). We call these vertices *heavy*. Then, the algorithm uses randomness to create F, a set of *friends* (Line 4). Each vertex v is independently included in F with probability equal to the number of its heavy neighbors divided by 4Δ. We show that $\mathbb{E}[|F|] = O(|H|)$ and $G[H \cup F]$ contains a matching of expected size $\Omega(|H|)$. This kind of matching is likely found by `MatchHeavy` in Line 5.

Note that `GlobalAlg` could as well compute a maximal matching in $G[H \cup F]$ instead of calling `MatchHeavy`. However, for the purpose of the analysis, using `MatchHeavy` is simpler, as we can directly relate the size of the obtained matching to the size of H. In addition, we later give a parallel version of `GlobalAlg`, and `MatchHeavy` is easy to parallelize.

At the end of the phase, vertices in both H and F are removed from the graph, while the matching found in $G[H \cup F]$ is added to the global matching being constructed. It is easy to see, that by removing H, the algorithm ensures that no vertex of degree larger than $\Delta/2$ remains in the graph, and therefore the bound on the maximum degree decreases by a factor of two.

2.2 Analysis

We start our analysis of the algorithm by showing that the execution of `MatchHeavy` in each phase of `GlobalAlg` finds a relatively large matching in expectation.

Lemma 2.1. *Consider one phase of* `GlobalAlg`*. Let H be the set of heavy vertices.* `MatchHeavy` *finds a matching $\widetilde{M}$ such that $\mathbb{E}\left[\left|\widetilde{M}\right|\right] \geq \frac{1}{40}|H|$.*

Proof. Observe that the set $E_\star$ is a collection of vertex-disjoint stars: each edge connects a red vertex with a blue vertex and the red vertices have degree 1. Thus, a subset of $E_\star$ forms a valid matching as long as no blue vertex is incident to two matched edges. Note that this is guaranteed by how edges are added to $\widetilde{M}$ in Line 4.

The size of the computed matching is the number of blue vertices in H that have at least one incident edge in $E_\star$. Let us now lower bound the number of such vertices. Consider an arbitrary $u \in H$. It has the desired properties exactly when the following three independent events happen: some $v \in F$ selects u in Line 1, u is colored blue, and v is colored red. The joint probability of the two latter events is exactly $\frac{1}{4}$. The probability that u is *not* selected by some $v \in F$ is

$$\left(1 - \frac{1}{4\Delta}\right)^{|N(u)\cap V'|} \leq \left(1 - \frac{1}{4\Delta}\right)^{\Delta/2} \leq \exp\left(-\frac{1}{4\Delta} \cdot \frac{\Delta}{2}\right) \leq \exp\left(-\frac{1}{8}\right) \leq \frac{9}{10}.$$

This implies that u is selected by a neighbor $v \in F$ with probability at least $\frac{1}{10}$. Therefore, with probability at least $\frac{1}{10} \cdot \frac{1}{4} = \frac{1}{40}$, u is blue and incident to an edge in $E_\star$. Hence, $\mathbb{E}\left[\left|\widetilde{M}\right|\right] \geq \frac{1}{40}|H|$. □

Next we show an upper bound on the expected size of F, the set of friends.

Lemma 2.2. *Let H be the set of heavy vertices selected in a phase of* `GlobalAlg`*. The following bound holds on the expected size of F, the set of friends, created in the same phase: $\mathbb{E}[|F|] \leq \frac{1}{4}|H|$.*

Proof. At the beginning of a phase, every vertex $u \in V'$—including those in H—has its degree, $|N(u)\cap V'|$, bounded by Δ. Reversing the order of the summation and applying this fact, we get:

$$\mathbb{E}[|F|] = \sum_{v \in V'} \frac{|N(v) \cap H|}{4\Delta} = \sum_{u \in H} \frac{|N(u) \cap V'|}{4\Delta} \leq \frac{|H| \cdot \Delta}{4\Delta} = \frac{|H|}{4}.$$ □

We combine the last two bounds to lower bound the expected size of the matching computed by `GlobalAlg`.

Lemma 2.3. *Consider an input graph G with an upper bound $\widetilde{\Delta}$ on the maximum vertex degree.* `GlobalAlg`$(G, \widetilde{\Delta})$ *executes $T \stackrel{\text{def}}{=} \lfloor \log \widetilde{\Delta} \rfloor + 1$ phases. Let H_i, F_i, and $\widetilde{M_i}$ be the sets H, F, and $\widetilde{M}$ constructed in phase i for $i \in [T]$. The following relationship holds on the expected sizes of these sets:*

$$\sum_{i=1}^{T} \mathbb{E}\left[\left|\widetilde{M_i}\right|\right] \geq \frac{1}{50} \sum_{i=1}^{T} \mathbb{E}\left[|H_i| + |F_i|\right]$$

Proof. For each phase $i \in [T]$, by applying the expectation over all possible settings of the set H_i, we learn from Lemmas 2.1 and 2.2 that

$$\mathbb{E}\left[\left|\widetilde{M_i}\right|\right] \geq \frac{1}{40}\,\mathbb{E}\left[|H_i|\right] \qquad \text{and} \qquad \mathbb{E}\left[|F_i|\right] \leq \frac{1}{4}\,\mathbb{E}\left[|H_i|\right].$$

It follows that

$$\frac{1}{50}\,\mathbb{E}\left[|H_i| + |F_i|\right] \leq \frac{1}{50}\,\mathbb{E}\left[|H_i|\right] + \frac{1}{200}\,\mathbb{E}\left[|H_i|\right] = \frac{1}{40}\,\mathbb{E}\left[|H_i|\right] \leq \mathbb{E}\left[\left|\widetilde{M_i}\right|\right],$$

and the statement of the lemma follows by summing over all phases. □

We do not use this fact directly in our paper, but note that the last lemma can be used to show that `GlobalAlg` can be used to find a large matching.

Corollary 2.4. `GlobalAlg` *computes a constant factor approximation to the maximum matching with $\Omega(1)$ probability.*

Proof. First, note that `GlobalAlg` finds a correct matching, i.e., no two different edges in M share an endpoint. This is implied by the fact that M is extended in every phase by a matching on a disjoint set of vertices.

Let T and sets H_i, F_i, and $\widetilde{M_i}$ for $i \in [T]$ be defined as in the statement of Lemma 2.3. Let M_{OPT} be a maximum matching in the graph. Observe that at the end of the algorithm execution, the remaining graph is empty. This implies that the size of the maximum matching can be bounded by the total number of removed vertices, because each removed vertex decreases the maximum matching size by at most one:

$$\sum_{i=1}^{T} |H_i| + |F_i| \geq |M_{\text{OPT}}|.$$

Hence, using Lemma 2.3,

$$\mathbb{E}\left[|M|\right] = \sum_{i=1}^{T} \mathbb{E}\left[\left|\widetilde{M_i}\right|\right] \geq \frac{1}{50} \sum_{i=1}^{T} \mathbb{E}\left[|H_i| + |F_i|\right] \geq \frac{1}{50}\,|M_{\text{OPT}}|.$$

Since $|M| \leq |M_{\text{OPT}}|$, $|M| \geq \frac{1}{100}\,|M_{\text{OPT}}|$ with probability at least $\frac{1}{100}$. Otherwise, $\mathbb{E}\left[|M|\right]$ would be strictly less than $\frac{1}{100} \cdot |M_{\text{OPT}}| + 1 \cdot \frac{1}{100}\,|M_{\text{OPT}}| = \frac{1}{50}\,|M_{\text{OPT}}|$, which is not possible. □

3 Emulation of a Phase in a Randomly Partitioned Graph

In this section, we introduce a modified version of a single phase (one iteration of the main loop) of `GlobalAlg`. Our modifications later allow for implementing the algorithm in the MPC model. The pseudocode of the new procedure, `EmulatePhase`, is presented as Algorithm 3. We partition the vertices of the current graph into m sets V_i, $1 \leq i \leq m$. Each vertex is assigned independently and *almost* uniformly at random to one of the sets. For each set V_i, we run a subroutine `LocalPhase` (presented as Algorithm 4). This subroutine runs a carefully crafted approximate version of one phase of `GlobalAlg` with an appropriately rescaled threshold Δ. More precisely, the threshold passed to the subroutine is scaled down by a factor of m, which corresponds to how approximately vertex degrees decrease in subgraphs induced by each of the sets. The main intuition behind this modification is that we hope to break the problem up into smaller subproblems on disjoint induced subgraph, and obtain similar *global* properties by solving the problem approximately on each smaller part. Later, in Section 4, we design an algorithm that assigns the subproblems to different machines and solves them in parallel.

Algorithm 3: `EmulatePhase`$(\Delta, G_\star, m, \mathcal{D})$
Emulation of a single phase in a randomly partitioned graph

Input:
- threshold Δ
- induced subgraph $G_\star = (V_\star, E_\star)$ of maximum degree $\frac{3}{2}\Delta$
- number m of subgraphs
- ϵ-near uniform and independent distribution $\mathcal{D}$ on assignments of $V_\star$ to $[m]$

Output: Remaining vertices and a matching

1 Pick a random assignment $\Phi : V_\star \to [m]$ from $\mathcal{D}$
2 **for** $i \in [m]$ **do**
3 $\quad V_i \leftarrow \{v \in V_\star : \Phi(v) = i\}$
4 $\quad (V_i', M_i) \leftarrow$ `LocalPhase`$(i, G_\star[V_i], \Delta/m)$ `/* LocalPhase = Algorithm 4 */`
5 **return** $(\bigcup_{i=1}^{m} V_i', \bigcup_{i=1}^{m} M_i)$

We now discuss `LocalPhase` (i.e., Algorithm 4) in more detail. Table 2 introduces two parameters, α and μ_R, and two functions, μ_H and μ_F, which are used in `LocalPhase`. Note first that α is a parameter used in the definition of μ_H but it is not used in the pseudocode of `LocalPhase` (or `EmulatePhase`) for anything else. It is, however, a convenient abbreviation in the analysis and the later parallel algorithm. The other three mathematical objects specify probabilities with which vertices are included in sets that are created in an execution of `LocalPhase`.

Apart from creating its own versions of H, the set of heavy vertices, and F, the set of friends, `LocalPhase` constructs also a set R_i, which we refer to as a *reference set*. In Line 1, the algorithm puts each vertex in R_i independently and with the same probability μ_R. The reference set is used to estimate the degrees of other vertices in the same induced subgraph in Line 2. For each vertex v_i, its estimate $\widehat{d}_v$ is defined as the number of v's neighbors in R_i multiplied by μ_R^{-1} to compensate for sampling. Next, in Line 3, the algorithm uses the estimates to create H_i, the set of heavy vertices. Recall that `GlobalAlg` uses a sharp threshold for selecting heavy vertices: all vertices of degree at least $\Delta/2$ are placed in H_i. `LocalPhase` works differently. It divides the degree estimate by the current threshold $\Delta_\star$ and uses function μ_H to decide with what probability the corresponding vertex is included in H_i. A sketch of the function can be seen in Figure 1. The function transitions from almost 0 to almost 1 in the neighborhood of $\frac{1}{2}$ at a limited pace. As a result vertices of degrees smaller than, say, $\frac{1}{4}\Delta$ are very unlikely to be included in H_i and vertices of

A multiplicative constant used in the exponent of μ_H:

$$\alpha \stackrel{\text{def}}{=} 96 \ln n.$$

The probability of the selection for a reference set:

$$\mu_R \stackrel{\text{def}}{=} \left(10^6 \cdot \log n\right)^{-1}.$$

The probability of the selection for a heavy set (used with r equal to the ratio of the estimated degree to the current threshold):

$$\mu_H(r) \stackrel{\text{def}}{=} \begin{cases} \frac{1}{2}\exp\left(\frac{\alpha}{2}\left(r-1/2\right)\right) & \text{if } r \leq 1/2, \\ 1-\frac{1}{2}\exp\left(-\frac{\alpha}{2}\left(r-1/2\right)\right) & \text{if } r > 1/2. \end{cases}$$

The probability of the selection for the set of friends (used with r equal to the ratio of the number of heavy neighbors to the current threshold):

$$\mu_F(r) \stackrel{\text{def}}{=} \begin{cases} \max\{r/4, 0\} & \text{if } r \leq 4, \\ 1 & \text{if } r > 4. \end{cases}$$

Table 2: Global parameters $\alpha \in (1, \infty)$ and $\mu_R \in (0, 1)$ and functions $\mu_H : \mathbb{R} \to [0, 1]$ and $\mu_F : \mathbb{R} \to [0, 1]$ used in the parallel algorithm. α, μ_R, and μ_H depend on n, the total number of vertices in the graph.

degree greater than $\frac{3}{4}\Delta$ are very likely to be included in H_i. `GlobalAlg` can be seen as an algorithm that instead of μ_H, uses a step function that equals 0 for arguments less than $\frac{1}{2}$ and abruptly jumps to 1 for larger arguments. Observe that without μ_H, the vertices whose degrees barely qualify them as heavy could behave very differently depending on which set they were assigned to. We use μ_H to guarantee a smooth behavior in such cases. That is one of the key ingredients that we need for making sure that a set of vertices that remains on one machine after a phase has almost the same statistical properties as a set of vertices obtained by new random partitioning.

Finally, in Line 4, `LocalPhase` creates a set of friends. This step is almost identical to what happens in the global algorithm. The only difference is that this time we have no upper bound on the number of heavy neighbors of a vertex. As a result that number divided by $4\Delta_\star$ can be greater than 1, in which case we have to replace it with 1 in order to obtain a proper probability. This is taken care of by function μ_F. Once H_i and F_i have been created, the algorithm finds a maximal matching M_i in the subgraph induced by the union of these two sets. The algorithm discards from the further consideration not only H_i and F_i, but also R_i. This eliminates dependencies in the possible distribution of assignments of vertices that have not been removed yet if we condition this distribution on the configuration of sets that have been removed. Intuitively, the probability of a vertex's inclusion in any of these sets depends only on R_i and H_i but not on any other vertices. Hence, once we fix the sets of removed vertices, the assignment of the remaining vertices to subgraphs is fully independent.[2] The output of `LocalPhase` is a subset of V_i to be considered in later phases and a matching M_i, which is used to expand the matching that we construct for the entire input

[2]By way of comparison, consider observing an experiment in which we toss the same coin twice. The bias of the coin is not fixed but comes from a random distribution. If we do not know the bias, the outcomes of the coin tosses are not independent. However, if we do know the bias, the outcomes are independent, even though they have the same bias.

Algorithm 4: `LocalPhase`$(i, G_i, \Delta_\star)$
Emulation of a single phase on an induced subgraph

Input:
- induced subgraph number i (useful only for the analysis)
- induced subgraph $G_i = (V_i, E_i)$
- threshold $\Delta_\star \in \mathbb{R}_+$

Output: Remaining vertices and a matching on V_i

1 Create a *reference set* R_i by independently selecting each vertex in V_i with probability μ_R.
2 For each $v \in V_i$, $\widehat{d}_v \leftarrow |N(v) \cap R_i|/\mu_R$.
3 Create a set H_i of *heavy vertices* by independently selecting each $v \in V_i$ with probability $\mu_H\left(\widehat{d}_v/\Delta_\star\right)$.
4 Create a set F_i of *friends* by independently selecting each vertex in $v \in V_i$ with probability $\mu_F\left(|N(v) \cap H_i|/\Delta_\star\right)$.
5 Compute a maximal matching M_i in $G[H_i \cup F_i]$.
6 **return** $(V_i \setminus (R_i \cup H_i \cup F_i), M_i)$

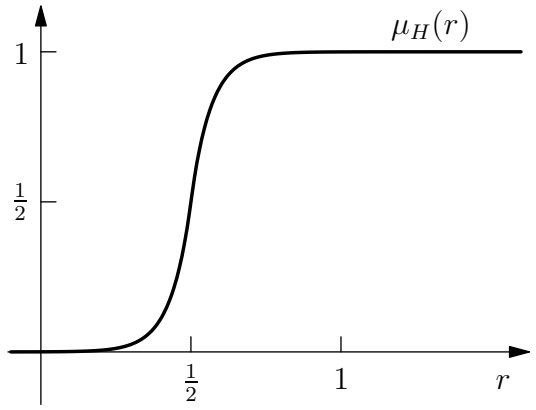

Figure 1: An idealized version of $\mu_H : \mathbb{R} \to [0, 1]$, in which n was fixed to a small constant and the multiplicative constant inside the exponentiation operator was lowered.

graph. We now introduce additional concepts and notation. They are useful for describing and analyzing properties of the algorithm. A configuration describes sets R_i, H_i, and F_i, for $1 \leq i \leq m$, constructed in an execution of `EmulatePhase`. We use it for conditioning a distribution of vertex assignments as described in the previous paragraph. We also formally define two important properties of distributions of vertex assignments: independence and near uniformity.

Configurations. Let m and $V_\star$ be the parameters to `EmulatePhase`: the number of subgraphs and the set of vertices in the graph to be partitioned, respectively. We say that

$$\mathcal{C} = (\{R_i\}_{i\in[m]}, \{H_i\}_{i\in[m]}, \{F_i\}_{i\in[m]})$$

is an *m-configuration* if it represents a configuration of sets R_i, H_i, and F_i created by `EmulatePhase` in the simulation of a phase. Recall that for any $i \in [m]$, R_i, H_i, and F_i are the sets created (and removed) by the execution of `LocalPhase` for V_i, the i-th subset of vertices.

We say that a vertex v is *fixed* by $\mathcal{C}$ if it belongs to one of the sets in the configuration, i.e.,

$$v \in \bigcup_{i \in [m]} (R_i \cup H_i \cup F_i)\,.$$

Conditional distribution. Let $\mathcal{D}$ be a distribution on assignments $\varphi : V_\star \to [m]$. Suppose that we execute `EmulatePhase` for $\mathcal{D}$ and let $\mathcal{C}$ be a *non-zero probability* m-configuration—composed of sets R_i, H_i, and F_i for $i \in [m]$—that can be created in this setting. Let $V'_\star$ be the set of vertices in $V_\star$ that are *not* fixed by $\mathcal{C}$. We write $\mathcal{D}[\mathcal{C}]$ to denote the conditional distribution of possible assignments of vertices in $V'_\star$ to $[m]$, given that for all $i \in [m]$, R_i, H_i, and F_i in $\mathcal{C}$ were the sets constructed by `LocalPhase` for the i-th induced subgraph.

Near uniformity and independence. Let $\mathcal{D}$ be a distribution on assignments $\varphi : \widetilde{V} \to [m]$ for some set $\widetilde{V}$ and m. For each vertex $v \in \widetilde{V}$, let $p_v : [m] \to [0,1]$ be the probability mass function of the marginal distribution of v's assignment. For any $\epsilon \geq 0$, we say that $\mathcal{D}$ is *ϵ-near uniform* if for every vertex v and every $i \in [m]$, $p_v(i) \in [\![(1 \pm \epsilon)/m]\!]$. We say that $\mathcal{D}$ is an *independent* distribution if the probability of every assignment φ in $\mathcal{D}$ equals exactly $\prod_{v \in V'} p_v(\varphi(v))$.

Concentration inequality. We use the following version of the Chernoff bound that depends on an upper bound on the expectation of the underlying independent random variables. It can be shown by combining two applications of the more standard version.

Lemma 3.1 (Chernoff bound)**.** *Let $X_1, \ldots, X_k$ be independently distributed random variables taking values in $[0,1]$. Let $X \stackrel{\text{def}}{=} X_1 + \cdots + X_k$ and let $U \geq 0$ be an upper bound on the expectation of X, i.e., $\mathbb{E}[X] \leq U$. For any $\delta \in [0,1]$, $\Pr(|X - \mathbb{E}[X]| > \delta U) \leq 2\exp(-\delta^2 U/3)$.*

We now show the properties of `EmulatePhase` that we use to obtain our final parallel algorithm.

3.1 Expected matching size

First, we show that `EmulatePhase` computes a large matching. Each vertex belonging to H_i or F_i that `EmulatePhase` removes in the calls to `LocalPhase` can decrease the maximum matching size in the graph induced by the remaining vertices by one. We show that the matching that `EmulatePhase` constructs in the process captures on average at least a constant fraction of that loss. We also show that the effect of removing R_i is negligible.

We start the proof by showing that the expected total size of sets H_i and F_i is not significantly impacted by relatively low-degree vertices classified as heavy or by an unlucky assignment of vertices to subgraphs resulting in local vertex degrees not corresponding to global degrees. Namely, we show that the expected number of friends a heavy vertex adds is $O(1)$ and at the same time the probability that the vertex gets matched is $\Omega(1)$.

Lemma 3.2. *Let Δ, $G_\star = (V_\star, E_\star)$, m, and $\mathcal{D}$ be parameters for* `EmulatePhase` *such that*

- *$\mathcal{D}$ is an independent and ϵ-near uniform distribution on assignments of vertices $V_\star$ to $[m]$ for $\epsilon \in [0, 1/200]$,*
- *$\frac{\Delta}{m} \geq 4000\mu_R^{-2}\ln^2 n$,*

- *the maximum degree of a vertex in $G_\star$ is at most $\frac{3}{2}\Delta$.*

For each $i \in [m]$, let H_i, F_i, and M_i be the sets constructed by `LocalPhase` *for the i-th induced subgraph. Then, the following relationship holds for their expected sizes:*

$$\sum_{i\in[m]} \mathbb{E}\left[|H_i \cup F_i|\right] \leq n^{-9} + 1200 \sum_{i\in[m]} \mathbb{E}\left[|M_i|\right].$$

Proof. We borrow more notation from `EmulatePhase` and the m executions of `LocalPhase` initiated by it. For $i \in [m]$, V_i is the set inducing the i-th subgraph. Value $\Delta_\star = \frac{\Delta}{m}$ is the rescaled threshold passed to the executions of `LocalPhase`. R_i is the reference set created by `LocalPhase` for the i-th induced subgraph.

For each induced subgraph, `LocalPhase` computes a maximal matching M_i in Line 5. While such a matching is always large—its size is at least half the maximum matching size—it is hard to relate ts size directly to the sizes of H_i and F_i. Therefore, we first analyze the size of a matching that would be created by `MatchHeavy`$(G_\star[H_i \cup F_i], H_i, F_i)$. We refer to this matching as $\widetilde{M}_i$ and we later use the inequality $\left|\widetilde{M}_i\right| \leq 2\left|M_i\right|$.

We partition each H_i, $i \in [m]$, into two sets: H'_i and H''_i. H'_i is the subset of vertices in H_i of degree less than $\frac{1}{8}\Delta$ in $G_\star$. $H''_{i,t+1}$ is its complement, i.e., $H''_i \stackrel{\text{def}}{=} H_i \setminus H'_i$. We start by bounding the expected total size of sets H'_i. What is the probability that a given vertex v of degree less than $\frac{1}{8}\Delta$ is included in $\bigcup_{i\in[m]} H_i$? Suppose that $v \in V_k$, where $k \in [m]$. The expected number of v's neighbors in R_k is at most $(1+\epsilon)\cdot\mu_R\cdot\frac{1}{8}\Delta/m \leq \frac{3}{16}\mu_R\Delta_\star$ due to the independence and ϵ-near uniformity of $\mathcal{D}[\mathcal{C}]$. Using the independence, Lemma 3.1, and the lower bound on $\Delta_\star$, we obtain the following bound:

$$\Pr\left[\mu_R\widehat{d}_v > \frac{1}{4}\mu_R\Delta_\star\right] \leq 2\exp\left(-\frac{1}{3}\cdot\left(\frac{1}{3}\right)^2\cdot\frac{3}{16}\mu_R\Delta_\star\right) \leq 2\exp\left(-27\ln n\right) = 2n^{-27}.$$

If $\widehat{d}_v \leq \frac{1}{4}\Delta_\star$, the probability that v is selected to H_k is at most $\mu_H(\widehat{d}_v/\Delta_\star) \leq \mu_H(1/4) \leq \frac{1}{2}n^{-12}$. Hence v is selected to H_k—and therefore to H'_k—with probability at most $2n^{-27} + \frac{1}{2}n^{-12} \leq n^{-12}$. This implies that $\sum_{i\in[m]}\mathbb{E}\left[|H'_i|\right] \leq n\cdot n^{-12} = n^{-11}$.

We also partition the sets of friends, F_i for $i \in [m]$, into two sets each: F'_i and F''_i. This partition is based on the execution of `MatchHeavy` for the i-th subgraph. In Line 1, this algorithm selects for every vertex $v \in F_i$ a random heavy neighbor $v_\star \in H_i$. If $v_\star \in H'_i$, we assign v to F'_i. Analogously, if $v_\star \in H''_i$, we assign v to F''_i. Obviously, a heavy vertex in H'_i can be selected only if H'_i is non-empty. By Markov's inequality and the upper bound on $\sum_{i\in[m]}\mathbb{E}\left[|H'_i|\right]$, the probability that at least one set H'_i is non-empty is at most n^{-11}. Even if for all $i \in [m]$, all vertices in F_i select a heavy neighbor in H'_i whenever it is available, the total expected number of vertices in sets F'_i is at most $\sum_{i\in[m]}\mathbb{E}\left[\left|F'_{i,t+1}\right|\right] \leq n\cdot n^{-11} = n^{-10}$.

Before we proceed to bounding sizes of the remaining sets, we prove that with high probability, all vertices have a number of neighbors close to the expectation. Let $\varphi : V_\star \to [m]$ be the assignment of vertices to subgraphs. We define $\mathcal{E}$ as the event that for all $v \in V_\star$,

$$\left|\frac{1}{m}\left|N(v)\cap V_\star\right| - \left|N(v)\cap V_{\varphi(v)}\right|\right| \leq \frac{1}{16}\Delta_\star.$$

Consider first one fixed $v \in V_\star$. The degree of v in $G_\star$ is $|N(v)\cap V_\star| \leq \frac{3}{2}\Delta$. Due to the near-uniformity and independence,

$$\left|\frac{1}{m}\left|N(v)\cap V_\star\right| - \mathbb{E}\left[\left|N(v)\cap V_{\varphi(v)}\right|\right]\right| \leq \epsilon\cdot\frac{3}{2}\frac{\Delta}{m} \leq \frac{3}{400}\Delta_\star.$$

This in particular implies that $\mathbb{E}\left[\left|N(v) \cap V_{\varphi(v)}\right|\right] \leq \left(\frac{3}{2} + \frac{3}{400}\right)\Delta_\star \leq 2\Delta_\star$. Using the independence of $\mathcal{D}$, Lemma 3.1, and the lower bound on $\Delta_\star$ (i.e., $\Delta_\star = \frac{\Delta}{m} \geq 4000\mu_R^{-2}\ln^2 n = 4 \cdot 10^{15} \cdot \ln^4 n$),

$$\begin{aligned}\Pr\left[\left|\mathbb{E}\left[\left|N(v) \cap V_{\varphi(v)}\right|\right] - \left|N(v) \cap V_{\varphi(v)}\right|\right| > \frac{1}{20}\Delta_\star\right] &\leq 2\exp\left(-\frac{1}{3} \cdot \left(\frac{1}{20} \cdot \frac{1}{2}\right)^2 \cdot 2\Delta_\star\right) \\ &\leq 2\exp\left(-(10^{12}+3)\ln n\right) \\ &\leq n^{-(10^{12}+2)} \leq n^{-12}.\end{aligned}$$

As a result, with this probability, we have

$$\left|\frac{1}{m}\left|N(v) \cap V_\star\right| - \left|N(v) \cap V_{\varphi(v)}\right|\right| \leq \frac{1}{20}\Delta_\star + \frac{3}{400}\Delta_\star \leq \frac{1}{16}\Delta_\star.$$

By the union bound, this bound holds for all vertices in $V_\star$ simultaneously—and hence $\mathcal{E}$ occurs—with probability at least $1 - n \cdot n^{-12} = 1 - n^{-11}$.

If $\mathcal{E}$ does not occur, we can bound both $\sum_{i\in[m]}|H_i''|$ and $\sum_{i\in[m]}|F_i''|$ by n. This contributes at most $n^{-11}\cdot n = n^{-10}$ to the expected size of each of these quantities. Suppose now that $\mathcal{E}$ occurs. Consider an arbitrary $v \in H_i''$ for some i. The number of neighbors of v in V_i lies in the range $\left[\frac{1}{8}\Delta_\star - \frac{1}{16}\Delta_\star, \frac{3}{2}\Delta_\star + \frac{1}{16}\Delta_\star\right] \subseteq \left[\frac{1}{16}\Delta_\star, 2\Delta_\star\right]$. Moreover, the expected number of vertices $w \in F_i''$ that select v in $w_\star$ in Line 1 of `MatchHeavy` is bounded by $2\Delta_\star \cdot \frac{1}{4\Delta_\star} = \frac{1}{2}$. It follows that $\mathbb{E}\left[|F_i''|\right] \leq \frac{1}{2}\mathbb{E}\left[|H_i''|\right]$, given $\mathcal{E}$. We now lower bound the expected size of $\widetilde{M}_i$ given $\mathcal{E}$. What is the probability that some vertex $w \in F_i$ selects v as $w_\star$ in `MatchHeavy` and (v, w) is added to $\widetilde{M}_i$?

This occurs if one of v's neighbors w is added to F_i and selects v as $w_\star$, and additionally, v and w are colored blue and red, respectively. The number of v's neighbors is at least $\frac{1}{16}\Delta_\star$. Since each vertex w in V_i has at most $2\Delta_\star$ neighbors, the number of heavy neighbors of w is bounded by the same number. This implies that in the process of selecting F_i, only the first branch in the definition of μ_F is used and each vertex w is included with probability exactly equal to the number of its neighbors in H_i divided by $4\Delta_{t+1}$. Then each heavy neighbor of w is selected as $w_\star$ with probability one over the number of heavy neighbors of w. What this implies is that each neighbor w of v is selected for F_i *and* selects v as $w_\star$ with probability exactly $(4\Delta_\star)^{-1}$. Hence the probability that v is *not* selected as $w_\star$ by any of its at least $\frac{1}{16}\Delta_\star$ neighbors w can be bounded by

$$\left(1 - \frac{1}{4\Delta_\star}\right)^{\frac{1}{16}\Delta_\star} \leq \exp\left(-\frac{1}{4\Delta_\star} \cdot \frac{1}{16}\Delta_\star\right) = e^{-1/64}.$$

Therefore the probability that v is selected by some vertex $w \in F_i$ as $w_\star$ is at least $1 - e^{-1/64} \geq 1/100$. Then with probability $1/4$, these two vertices have appropriate colors and this or another edge incident to v with the same properties is added to $\widetilde{M}_i$. In summary, the probability that an edge (v, w) for some w as described is added to $\widetilde{M}_i$ is at least $1/400$. Since we do not count any edge in the matching twice for two heavy vertices, by the linearity of expectation $\mathbb{E}\left[\left|\widetilde{M}_i\right|\right] \geq \frac{1}{400}\mathbb{E}\left[|H_i''|\right]$ given $\mathcal{E}$. Overall, given $\mathcal{E}$, we have

$$\sum_{i\in[m]}\mathbb{E}\left[\left|H_i''\right| + \left|F_i''\right|\right] \leq \frac{3}{2}\sum_{i\in[m]}\mathbb{E}\left[\left|H_i''\right|\right] \leq 600\sum_{i\in[m]}\mathbb{E}\left[\left|\widetilde{M}_i\right|\right].$$

In general, without conditioning on $\mathcal{E}$,

$$\sum_{i\in[m]}\mathbb{E}\left[\left|H_i''\right| + \left|F_i''\right|\right] \leq 2 \cdot n^{-10} + 600\sum_{i\in[m]}\mathbb{E}\left[\left|\widetilde{M}_i\right|\right].$$

We now combine bounds on all terms to finish the proof of the lemma.

$$\begin{aligned}\sum_{i\in[m]} \mathbb{E}\left[|H_i \cup F_i|\right] &\le \sum_{i\in[m]} \mathbb{E}\left[\left|H_i'\right| + \left|F_i'\right| + \left|H_i''\right| + \left|F_i''\right|\right] \\ &\le n^{-11} + n^{-10} + 2n^{-10} + 600 \sum_{i\in[m]} \mathbb{E}\left[\left|\widetilde{M}_i\right|\right] \\ &\le n^{-9} + 1200 \sum_{i\in[m]} \mathbb{E}\left[|M_i|\right].\end{aligned}$$

□

3.2 Independence

Note that Lemma 3.2 requires that the vertices are distributed independently and near uniformly in the m sets. This is trivially the case right after the vertices are partitioned independently at random. However, in the final algorithm, after we partition the vertices, we run *multiple* phases on each machine. In the rest of this section we show that running a single phase *preserves* independence of vertex distribution and only slightly disturbs the uniformity (Lemma 3.3 and Lemma 3.6). As we have mentioned before, independence stems from the fact that we use reference sets to estimate vertex degrees. We discard them at the end and condition on them, which leads to the independence of the distribution of vertices that are not removed.

Lemma 3.3. *Let $\mathcal{D}$ be an independent distribution of assignments of vertices in $V_\star$ to $[m]$. Let $\mathcal{C}$ be a non-zero probability m-configuration that can be constructed by* `EmulatePhase` *for $\mathcal{D}$. Let $V_\star'$ be the set of vertices of $V_\star$ that are not fixed by $\mathcal{C}$. Then $\mathcal{D}[\mathcal{C}]$ is an independent distribution of vertices in $V_\star'$ on $[m]$.*

Now we prove Lemma 3.3. We start with an auxiliary lemma that gives a simple criterion under which an independent distribution remains independent after conditioning on a random event. Consider a random vector with independently distributed coordinates. Suppose that for any value of the vector, a random event $\mathcal{E}$ occurs when all coordinates "cooperate," where each coordinate cooperates independently with probability that depends only on the value of that coordinate. We then show that the distribution of the vector's coordinates given $\mathcal{E}$ remains independent.

Lemma 3.4. *Let k be a positive integer and A an arbitrary finite set. Let $X = (X_1, \ldots, X_k)$ be a random vector in A^k with independently distributed coordinates. Let $\mathcal{E}$ be a random event of non-zero probability. If there exist functions $p_i : A \to [0, 1]$, for $i \in [k]$, such that for any $x = (x_1, \ldots, x_k) \in A^k$ appearing with non-zero probability,*

$$\Pr[\mathcal{E}|X = x] = \prod_{i=1}^{k} p_i(x_i),$$

then the conditional distribution of coordinates in X given $\mathcal{E}$ is independent as well.

Proof. Since the distribution of coordinates in X is independent, there are k probability mass functions $p_i' : A \to [0, 1]$, $i \in [k]$, such that for every $x = (x_1, \ldots, x_k) \in A^k$, $\Pr[X = x] = \prod_{i=1}^{k} p_i'(x_i)$. The probability of $\mathcal{E}$ can be expressed as

$$\begin{aligned}\Pr[\mathcal{E}] &= \sum_{x=(x_1,\ldots,x_k)\in A^k} \Pr[\mathcal{E} \wedge X = x] = \sum_{\substack{x=(x_1,\ldots,x_k)\in A^k \\ \Pr[X=x]>0}} \Pr[\mathcal{E}|X = x] \cdot \Pr[X = x] \\ &= \sum_{x=(x_1,\ldots,x_k)\in A^k} \prod_{i=1}^{k} p_i(x_i)p_i'(x_i) = \prod_{i=1}^{k} \sum_{y\in A} p_i(y)p_i'(y).\end{aligned}$$

Note that since the probability of $\mathcal{E}$ is positive, each multiplicative term $\sum_{y\in A} p_i(y)p'_i(y)$, $i \in [k]$, in the above expression is positive. We can express the probability of any vector $x = (x_1, \ldots, x_k) \in A^k$ given $\mathcal{E}$ as follows:

$$\Pr[X = x|\mathcal{E}] = \frac{\Pr[\mathcal{E} \wedge X = x]}{\Pr[\mathcal{E}]} = \frac{\Pr[\mathcal{E}|X = x] \cdot \Pr[X = x]}{\Pr[\mathcal{E}]}$$
$$= \frac{\prod_{i=1}^{k} p_i(x_i)p'_i(x_i)}{\prod_{i=1}^{k} \sum_{y\in A} p_i(y)p'_i(y)} = \prod_{i=1}^{k} \frac{p_i(x_i)p'_i(x_i)}{\sum_{y\in A} p_i(y)p'_i(y)}.$$

We define $p''_i : A \to [0, 1]$ as $p''_i(x) \stackrel{\text{def}}{=} p_i(x_i)p'_i(x_i)/\sum_{y\in A} p_i(y)p'_i(y)$ for each $i \in [k]$. Each p''_i is a valid probability mass function on A. As a result we have $\Pr[X = x|\mathcal{E}] = \prod_{i=1}^{k} p''_i(x_i)$, which proves that the distribution of coordinates in X given $\mathcal{E}$ is still independent with each coordinate distributed according to its probability mass function p''_i. □

We now prove Lemma 3.3 by applying Lemma 3.4 thrice. We refer to functions p_i, which describe the probability of each coordinate cooperating, as *cooperation probability functions*.

of Lemma 3.3. $\mathcal{C}$ can be expressed as

$$\mathcal{C} = \left(\{R^\star_i\}_{i\in[m]}, \{H^\star_i\}_{i\in[m]}, \{F^\star_i\}_{i\in[m]}\right)$$

for some subsets $R^\star_i$, $H^\star_i$, and $F^\star_i$ of $V_\star$, where $i \in [m]$. We write Φ to denote the random assignment of vertices to sets selected in Line 1 of `EmulatePhase`. Φ is a random variable distributed according to $\mathcal{D}$.

Let $\mathcal{E}_R$ be the event that for all $i \in [m]$, the reference set R_i generated for the i-th induced subgraph by `LocalPhase` equals exactly $R^\star_i$. A vertex v that is assigned to a set V_i is included in R_i with probability exactly μ_R, independently of other vertices. Hence once we fix an assignment $\varphi : V_\star \to [m]$ of vertices to sets V_i, we can express the probability of $\mathcal{E}_R$ as a product of probabilities that each vertex cooperates. More formally, $\Pr[\mathcal{E}_R|\Phi = \varphi] = \prod_{v\in V_\star} q_v(\varphi(v))$ for cooperation probability functions $q_v : [m] \to [0, 1]$ defined as follows.

- If $v \in \bigcup_{i\in[m]} R^\star_i$, there is exactly one $i \in [m]$ such that $v \in R^\star_i$. If v is not assigned to V_i, $\mathcal{E}_R$ cannot occur. If it is, v cooperates with $\mathcal{E}_R$ with probability exactly μ_R, i.e., the probability of the selection for R_i. For this kind of v, the cooperation probability function is

$$q_v(i) \stackrel{\text{def}}{=} \begin{cases} \mu_R & \text{if } v \in R^\star_i, \\ 0 & \text{if } v \notin R^\star_i. \end{cases}$$

- If $v \notin \bigcup_{i\in[m]} R^\star_i$, v cooperates with $\mathcal{E}_R$ if it is not selected for $R_{\varphi(v)}$, independently of its assignment $\varphi(v)$, which happens with probability exactly $1 - \mu_R$. Therefore, the cooperation probability can be defined as $q_v(i) \stackrel{\text{def}}{=} 1 - \mu_R$ for all $i \in [m]$.

We invoke Lemma 3.4 to conclude that the conditional distribution of values of Φ given $\mathcal{E}_R$ is independent as well.

We now define an event $\mathcal{E}_H$ that both $\mathcal{E}_R$ occurs and for all $i \in [m]$, H_i, the set of heavy vertices constructed for the i-th subgraph equals exactly $H^\star_i$. We want to show that the conditional distribution of values of Φ given $\mathcal{E}_H$ is independent. Note that if Φ is selected from the conditional distribution given $\mathcal{E}_R$ (i.e., all sets R_i are as expected) and we fix the assignment $\phi : V_\star \to [m]$ of vertices to sets V_i, then each

vertex $v \in V_\star$ is assigned to $H_{\phi(v)}$—this the only set H_i to which it can be assigned—independently of other vertices. As a result, we can express the probability of $\mathcal{E}_H$ given $\mathcal{E}_R$ and φ being the assignment as a product of cooperation probabilities for each vertex. More precisely, $\Pr[\mathcal{E}_H|\Phi = \varphi, \mathcal{E}_R] = \prod_{v \in V_\star} q'_v(\varphi(v))$ for cooperation probability functions $q'_v : [m] \to [0,1]$ defined as follows, where $\Delta_\star$ is the threshold used in the m executions of `LocalPhase`.

- If $v \in \bigcup_{i \in [m]} H^\star_i$, then there is exactly one i such that $v \in H^\star_i$. $\mathcal{E}_H$ can only occur if v is included in the corresponding H_i. This cannot happen if v is not assigned to the corresponding V_i by φ. If v is assigned to this V_i, it has to be selected for H_i, which happens with probability $\mu_H\left(|N(v) \cap R^\star_i|/(\mu_R \Delta_\star)\right)$. The cooperation probability function can be written in this case as

$$q'_v(i) \stackrel{\text{def}}{=} \begin{cases} \mu_H(|N(v) \cap R^\star_i|/(\mu_R \Delta_\star)) & \text{if } v \in H^\star_i, \\ 0 & \text{if } v \notin H^\star_i. \end{cases}$$

- If $v \notin \bigcup_{i \in [m]} H^\star_i$, v cannot be included in H_i corresponding to the set V_i to which it is assigned for $\mathcal{E}_H$ to occur. This happens with probability $1 - \mu_H(|N(v) \cap R^\star_i|/(\mu_R \Delta_\star))$. Hence, we can define $q'_v(i) \stackrel{\text{def}}{=} 1 - \mu_H(|N(v) \cap R^\star_i|/(\mu_R \Delta_\star))$ for all $i \in [m]$.

We can now invoke Lemma 3.4 to conclude that the distribution of values of Φ given $\mathcal{E}_H$ is independent.

Finally, we define $\mathcal{E}_F$ to be the event that both $\mathcal{E}_H$ occurs and for each $i \in [m]$, F_i, the set of friends selected for the i-th induced subgraph, equals exactly $F^\star_i$. We observe that once Φ is fixed to a specific assignment $\varphi : V_\star \to [m]$ and $\mathcal{E}_H$ occurs (i.e., all sets R_i and H_i are as in $\mathcal{C}$), then each vertex is independently included in $F_{\varphi(v)}$ with some specific probability that depends only on $H_{\varphi(v)}$, which is already fixed. In this setting, we can therefore express the probability of $\mathcal{E}_F$, which exactly specifies the composition of sets F_i, as a product of values provided by some cooperation probability functions $q''_v : [m] \to [0,1]$. More precisely, $\Pr[\mathcal{E}_F|\Phi = \varphi, \mathcal{E}_H] = \prod_{v \in V_\star} q''_v(\varphi(v))$ for q''_v that we define next.

- If $v \in \bigcup_{i \in [m]} F^\star_i$, then there is exactly one i such that $v \in F^\star_i$. $\mathcal{E}_F$ cannot occur if v is not assigned to V_i and selected for F_i. Hence, the cooperation probability function for v is

$$q''_v(i) \stackrel{\text{def}}{=} \begin{cases} \mu_F(|N(v) \cap H^\star_i|/\Delta_\star) & \text{if } v \in F^\star_i, \\ 0 & \text{if } v \notin F^\star_i. \end{cases}$$

- If $v \notin \bigcup_{i \in [m]} F^\star_i$, to whichever set V_i vertex v is assigned, it should not be included in F_i in order for $\mathcal{E}_F$ to occur. Hence, $q''_v(i) \stackrel{\text{def}}{=} 1 - \mu_F(|N(v) \cap H^\star_{i_\star,t}|/\Delta_t)$.

We invoke Lemma 3.4 to conclude that the distribution of values of Φ given $\mathcal{E}_F$ is independent as well. This is a distribution on assignments for the entire set $V_\star$. If we restrict it to assignments of $V'_\star \subseteq V_\star$, we obtain a distribution that first, is independent as well, and second, equals exactly $\mathcal{D}[\mathcal{C}]$. □

3.3 Near Uniformity

The proof of near uniformity is the most involved proof in this paper. In a nutshell, the proof is structured as follows. We pick an arbitrary vertex v that has not been removed and show that with high probability it has the same number of neighbors in all sets R_i. The same property holds for v's neighbors in all sets H_i. We use this to show that the probability of a fixed configuration of sets removed in a single phase is roughly

the same for all assignments of v to subgraphs. In other words, if v was distributed nearly uniformly before the execution of `EmulatePhase`, it is distributed only slightly less uniformly after the execution.

We begin by proving a useful property of μ_H (see Table 2 for definition). Recall that `GlobalAlg` selects H, the set of heavy vertices, by taking all vertices of degree at least $\Delta/2$. In `LocalPhase` the degree estimate of each vertex depends on the number of neighbors in the reference set in the vertex's induced subgraph. We want the decision taken for each vertex to be approximately the same, independently of which subgraph it is assigned to. Therefore, we use μ_H—which specifies the probability of the inclusion in the set of heavy vertices—which is relatively insensitive to small argument changes. The next lemma proves that this is indeed the case. Small additive changes to the parameter x to μ_H have small multiplicative impact on both $\mu_H(x)$ and $1 - \mu_H(x)$.

Lemma 3.5 (Insensitivity of μ_H)**.** *Let $\delta \in [0, (\alpha/2)^{-1}] = [0, (48 \ln n)^{-1}]$. For any pair x and x' of real numbers such that $|x - x'| \leq \delta$,*

$$\mu_H(x') \in [\![\mu_H(x)(1 \pm \alpha\delta)]\!]$$

and

$$1 - \mu_H(x') \in [\![(1 - \mu_H(x))(1 \pm \alpha\delta)]\!]\,.$$

Proof. We define an auxiliary function $f : \mathbb{R} \to [0, 1]$:

$$f(r) \stackrel{\text{def}}{=} \begin{cases} \frac{1}{2}\exp\left(\frac{\alpha}{2}r\right) & \text{if } r \leq 0, \\ 1 - \frac{1}{2}\exp\left(-\frac{\alpha}{2}r\right) & \text{if } r > 0. \end{cases}$$

It is easy to verify that for all $r \in \mathbb{R}$, $\mu_H(r) = f(r - 1/2)$ and $1 - \mu_H(r) = f(-(r - 1/2))$. Therefore, in order to prove the lemma, it suffices to prove that for any r and r' such that $|r - r'| \leq \delta$,

$$f(r)(1 - \alpha\delta) \leq f(r') \leq f(r)(1 + \alpha\delta), \tag{1}$$

i.e., a small additive change to the argument of f has a limited multiplicative impact on the value of f.

Note that f is differentiable in both $(-\infty, 0)$ and $(0, \infty)$. Additionally, it is continuous in the entire range—the left and right branch of the function meet at 0—and both the left and right derivatives at 0 are equal. This implies that it is differentiable at 0 as well. Its derivative is

$$f'(r) = \begin{cases} \frac{\alpha}{4} \cdot \exp\left(\frac{\alpha}{2}r\right) & \text{if } r \leq 0, \\ \frac{\alpha}{4} \cdot \exp\left(-\frac{\alpha}{2}r\right) & \text{if } r > 0, \end{cases}$$

which is positive for all r, and therefore, f is strictly increasing. Note that f' is increasing in $(-\infty, 0]$ and decreasing in $[0, \infty)$. Hence the global maximum of f' equals $f'(0) = \alpha/4$.

In order to prove Inequality 1 for all r and r' such that $|r - r'| \leq \delta$, we consider two cases. Suppose first that $r \geq 0$. By the upper bound on the derivative of f,

$$f(r) - \frac{\alpha}{4} \cdot |r - r'| \leq f(r') \leq f(r) + \frac{\alpha}{4} \cdot |r - r'|.$$

Since $r \geq 0$, $f(r) \geq 1/2$. This leads to

$$f(r) - f(r) \cdot \frac{\alpha}{2} \cdot |r - r'| \leq f(r') \leq f(r) + f(r) \cdot \frac{\alpha}{2} \cdot |r - r'|.$$

By the bound on $|r - r'|$,

$$f(r)(1 - \alpha\delta) \leq f(r') \leq f(r)(1 + \alpha\delta),$$

which finishes the proof in the first case.

Suppose now that $r < 0$. Since f is increasing, it suffices to bound the value of f from below at $r - \delta$ and from above and at $r + \delta$. For $r - \delta$, we obtain

$$\begin{aligned} f(r-\delta) &= \frac{1}{2}\exp\left(\frac{\alpha}{2}(r-\delta)\right) = f(r)\exp\left(-\frac{\alpha}{2}\delta\right) \\ &\geq f(r)\left(1-\frac{\alpha}{2}\delta\right) \geq f(r)(1-\alpha\delta). \end{aligned}$$

For $r + \delta$, let us first define a function $g : \mathbb{R} \to \mathbb{R}$ as

$$g(y) \stackrel{\text{def}}{=} \frac{1}{2}\exp\left(\frac{\alpha}{2}y\right).$$

For $y \leq 0$, $f(y) = g(y)$. For $y > 0$, $g'(y) \geq f'(y)$ and hence, for any $y \in \mathbb{R}$, $g(y) \geq f(y)$. As a result, we obtain

$$f(r+\delta) \leq g(r+\delta) = \frac{1}{2}\exp\left(\frac{\alpha}{2}(r+\delta)\right) = f(r)\cdot\exp\left(\frac{\alpha}{2}\delta\right).$$

By the bound on δ in the lemma statement, $\frac{\alpha}{2}\delta \leq 1$. It follows from the convexity of the exponential function that for any $y \in [0,1]$, $\exp(y) \leq y \cdot \exp(1) + (1-y)\cdot\exp(0) \leq 3y + (1-y) = 1 + 2y$. Continuing the reasoning,

$$f(r+\delta) \leq f(r)\cdot\left(1 + 2\cdot\frac{\alpha}{2}\delta\right) = f(r)(1+\alpha\delta),$$

which finishes the proof of Inequality (1). □

The main result of this section is Lemma 3.6 that states that if a distribution $\mathcal{D}$ of vertex assignments is near uniform, then `EmulatePhase` constructs a configuration $\mathcal{C}$ such that $\mathcal{D}[\mathcal{C}]$ is near uniform as well, and also, the maximum degree in the graph induced by the vertices not removed by `EmulatePhase` is bounded.

Lemma 3.6. *Let Δ, $G_\star = (V_\star, E_\star)$, m, and $\mathcal{D}$ be parameters for* `EmulatePhase` *such that*

- *$\mathcal{D}$ is an independent and ϵ-near uniform distribution on assignments of vertices $V_\star$ to $[m]$ for $\epsilon \in [0, (200\ln n)^{-1}]$,*
- *$\frac{\Delta}{m} \geq 4000\mu_R^{-2}\ln^2 n$.*

Let $\mathcal{C}$ be an m-configuration constructed by `EmulatePhase`*. With probability at least $1 - n^{-4}$ both the following properties hold:*

- *The maximum degree in the graph induced by the vertices not fixed in $\mathcal{C}$ is bounded by $\frac{3}{4}\Delta$.*
- *$\mathcal{D}[\mathcal{C}]$ is $60\alpha\left(\left(\frac{\Delta}{m}\right)^{-1/4} + \epsilon\right)$-near uniform.*

Proof overview. This is the most intricate proof of the entire paper. We therefore provide a short overview. First, we list again the variables in `EmulatePhase` and `LocalPhase` to which we refer in the proof and define additional convenient symbols. Then we introduce five simple random events (Events 1–5) that capture properties needed to prove Lemma 3.6. In Claim 3.7, we show that the probability of all these events occurring simultaneously is high. The proof of the claim follows mostly from a repetitive application of the Chernoff bound. In the next claim, Claim 3.8, we show that the occurrence of all the events has a

few helpful consequences. First, high degree vertices get removed in the execution of `EmulatePhase` (which is one of our final desired properties). Second, each vertex v that is not fixed in $\mathcal{C}$ has a very similar number of neighbors in all sets R_i and it has a very similar number of neighbors in all sets H_i. In the final proof of Lemma 3.6, we use the fact that this implies that to whichever set V_i vertex v was assigned in `EmulatePhase`, the probability of its removal in `EmulatePhase` was more or less the same. This leads to the conclusion that if v was distributed nearly uniformly in $\mathcal{D}$, it is distributed only slightly less uniformly in $\mathcal{D}[\mathcal{C}]$.

Notation. To simplify the presentation, for the rest of Section 3.3, we assume that Δ, $G_\star = (V_\star, E_\star)$, m, and $\mathcal{D}$ are the parameters to `EmulatePhase` as in the statement of Lemma 3.6. Additionally, for each $i \in [m]$, R_i, H_i, and F_i are the sets constructed by `LocalPhase` for the i-th subgraph in the execution of `EmulatePhase`. We also write $\mathcal{C}$ denote the corresponding m-configuration, i.e., $\mathcal{C} = \left(\{R_i\}_{i\in[m]}, \{H_i\}_{i\in[m]}, \{F_i\}_{i\in[m]}\right)$. Furthermore, for each $v \in V_\star$, $\widehat{d}_v$ is the estimate of v's degree in the subgraph to which it was assigned. This estimate is computed in Line 2 of `LocalPhase`. We also use $\Delta_\star$ to denote the rescaled threshold passed in all calls to `LocalPhase`, i.e., $\Delta_\star = \frac{\Delta}{m}$.

We also introduce additional notation, not present in `EmulatePhase` or `LocalPhase`. For each $v \in V_\star$, $d_v \stackrel{\text{def}}{=} |N(v) \cap V_\star|$, i.e., d_v is the degree of v in $G_\star$. For each vertex $v \in V_\star$, we also introduce a notion of its *weight*: $w_v \stackrel{\text{def}}{=} \mu_H(d_v/\Delta)$, which can be seen as a very rough approximation of v's probability of being selected for the set of heavy vertices. For any $v \in V_\star$ and $U \subseteq V_\star$, we also introduce notation for the total weight of v's neighbors in U:

$$W_v(U) \stackrel{\text{def}}{=} \sum_{u \in N(v) \cap U} w_u.$$

Finally, for all $i \in [m]$ and $v \in V_\star$, we also introduce a slightly less intuitive notion of the expected number of heavy neighbors of v in the i-th subgraph after the degree estimates are fixed in Line 2 of `LocalPhase` and before vertices are assigned to the heavy set in Line 3:

$$h_{v,i} \stackrel{\text{def}}{=} \sum_{u \in N(v) \cap V_i} \mu_H\left(\widehat{d}_u/\Delta_\star\right).$$

Obviously, each $h_{v,i}$ is a random variable.

Convenient random events. We now list five random events that we hope all to occur simultaneously with high probability. The first event intuitively is the event that high-degree vertices are likely to be included in the set of heavy vertices in Line 3 of `LocalPhase`.

Event 1

For each vertex $v \in V_\star$ such that $d_v \geq \frac{3}{4}\Delta$,

$$\mu_H\left(\widehat{d}_v/\Delta_\star\right) \geq 1 - \frac{1}{2}n^{-6}.$$

Another way to define this event would be to state that $\widehat{d}_v$ for such vertices v is high, but this form is more suitable for our applications later. The next event is the event that all such vertices are in fact classified as

heavy.

Event 2

Each vertex $v \in V_\star$ such that $d_v \geq \frac{3}{4}\Delta$ belongs to $\bigcup_{i\in[m]} H_i$.

The next event is the event that low-degree vertices have a number of neighbors in each set R_i close to the mean. This implies that if we were able to move a low-degree vertex v to V_i, for any $i \in [m]$, its estimated degree $\widehat{d}_v$ would not change significantly.

Event 3

For each vertex $v \in V_\star$ such that $d_v < \frac{3}{4}\Delta$ and each $i \in [m]$,

$$\left|\frac{1}{\mu_R}\left|N(v) \cap R_i\right| - \frac{d_v}{m}\right| \leq \Delta_\star^{3/4} + \frac{3}{4}\epsilon\Delta_\star.$$

The next event is the event that low-degree vertices have a number of neighbors in each set R_i close to the mean. This implies that if we were able to move a low-degree vertex v to V_i, for any $i \in [m]$, its estimated degree $\widehat{d}_v$ would not change significantly.

Event 4

For each vertex $v \in V_\star$ such that $d_v < \frac{3}{4}\Delta$ and each $i \in [m]$,

$$|W_v(V_i) - W_v(V_\star)/m| \leq \Delta_\star^{3/4} + \frac{3}{4}\epsilon\Delta_\star.$$

Recall that $h_{v,i}$ intuitively expresses the expected number of v's neighbors in the i-th induced subgraph at some specific stage in the execution of `LocalPhase` for the i-th induced subgraph. The final event is the event that for all bounded $h_{v,i}$, the actual number of v's neighbors in H_i does not deviate significantly from $h_{v,i}$.

Event 5

For each vertex $v \in V_\star$ and each $i \in [m]$, if $h_{v,i} \leq 2\Delta_\star$, then

$$||N(v) \cap H_i| - h_{v,i}| \leq \Delta_\star^{3/4}.$$

High probability of the random events. We now show that the probability of all the events occurring is high. The proof follows mostly via elementary applications of the Chernoff bound.

Claim 3.7. *If $\epsilon \in [0, 1/100]$ and $\frac{\Delta}{m} \geq 4000\mu_R^{-2}\ln^2 n$, then Events 1–5 occur simultaneously with probability at least $1 - n^{-4}$.*

Proof. We consider all events in order and later show by the union bound that all of them hold simultaneously with high probability. In the proof of the lemma, we extensively use the fact that $\Delta_\star = \frac{\Delta}{m} \geq 4000\mu_R^{-2}\ln^2 n = 4 \cdot 10^{15} \cdot \ln^4 n$.

First, we consider Event 1 and Event 2, which we handle together. Consider a vertex v such that $d_v \geq \frac{3}{4}\Delta$. Let $i_\star$ be the index of the set to which it is assigned. Since $\mathcal{D}$ is ϵ-near uniform, the expectation of $|N(v) \cap R_{i_\star}|$, the number of v's neighbors in $R_{i_\star}$, is at least $(1-\epsilon)\frac{3}{4}\mu_R\frac{\Delta}{m} \geq \frac{297}{400}\mu_R\Delta_\star$. Since vertices are both assigned to machines independently and included in the reference set independently as well, we can

apply Lemma 3.1 to bound the deviation with high probability. The probability that the number of neighbors is smaller than $\frac{9}{10} \cdot \frac{297}{400}\mu_R\Delta_\star \geq \frac{5}{8}\mu_R\Delta_\star$ is at most

$$2\exp\left(-\frac{1}{3}\cdot\left(\frac{1}{10}\right)^2\cdot\frac{297}{400}\mu_R\Delta_\star\right) \leq 2\exp\left(-\frac{1}{405}\mu_R\Delta_\star\right) \leq 2n^{-9} \leq \frac{1}{2}n^{-6}.$$

Hence with probability at least $1-\frac{1}{2}n^{-6}$, $\widehat{d}_v \geq \frac{5}{8}\Delta_\star$ and $\mu_H\left(\widehat{d}_v/\Delta_\star\right) \geq 1-\frac{1}{2}n^{-6}$. If this is the case, v is not included in the set of heavy vertices in Line 3 of `LocalPhase` with probability at most $\frac{1}{2}n^{-6}$. Therefore, v has the desired value of $\mu_H\left(\widehat{d}_v/\Delta_\star\right)$ and belongs to $H_{i_\star}$ with probability at least $1-n^{-6}$. By the union bound, this occurs for all high degree vertices with probability at least $1-n^{-5}$, in which case both Event 1 and Event 2 occur.

We now show that Event 3 occurs with high probability. Let v be an arbitrary vertex such that $d_v < \frac{3}{4}\Delta$ and let $i \in [m]$. Let $X_{v,i} \stackrel{\text{def}}{=} |N(v)\cap R_i|$. $X_{v,i}$ is a random variable. Since $\mathcal{D}$ is ϵ-near uniform, $\mathbb{E}\left[X_{v,i}\right] \in [\![(1\pm\epsilon)\mu_R d_v/m]\!]$. In particular, due to the bounds on d_v and ϵ, $E[X_{v,i}] \leq \mu_R\Delta_\star$. Due to the independence, we can use Lemma 3.1 to bound the deviation of $X_{v,i}$ from its expectation. We have

$$\begin{aligned}\Pr\left(|X_{v,i}-\mathbb{E}[X_{v,i}]| > \mu_R\Delta_\star^{3/4}\right) &\leq 2\exp\left(-\frac{1}{3}\cdot\left(\frac{1}{\Delta_\star^{1/4}}\right)^2\cdot\mu_R\Delta_\star\right)\\ &= 2\exp\left(-\frac{1}{3}\mu_R\Delta_\star^{1/2}\right) \leq 2n^{-21}.\end{aligned}$$

Hence with probability $1-2n^{-21}$, we have

$$\begin{aligned}\left|X_{v,i}-\mu_R\frac{d_v}{m}\right| &\leq |X_{v,i}-\mathbb{E}[X_{v,i}]| + \left|\mathbb{E}[X_{v,i}]-\mu_R\frac{d_v}{m}\right| \leq \mu_R\Delta_\star^{3/4}+\epsilon\mu_R\frac{d_v}{m}\\ &\leq \mu_R\Delta_\star^{3/4}+\frac{3}{4}\epsilon\mu_R\Delta_\star.\end{aligned}$$

By dividing both sides by μ_R, we obtain the desired bound

$$\left|\frac{X_{v,i}}{\mu_R}-\frac{d_v}{m}\right| = \left|\frac{1}{\mu_R}|N(v)\cap R_i|-\frac{d_v}{m}\right| \leq \Delta_\star^{3/4}+\frac{3}{4}\epsilon\Delta_\star.$$

By the union bound, this holds for all v and i of interest—and therefore, Event 3 occurs—with probability at least $1-|V_\star|\cdot m\cdot 2n^{-21} \geq 1-n^{-5}$.

We now move on to Event 4. Consider a vertex v such that $d_v < \frac{3}{4}\Delta$ and $i \in [m]$. Note that since the weight of every vertex is at most 1, $W_v(V_\star)/m \leq d_v/m < \frac{3}{4}\Delta_\star$. Since $\mathcal{D}[\mathcal{C}]$ is ϵ-near uniform, $\mathbb{E}\left[W_v(V_i)\right] \in [\![(1\pm\epsilon)W_v(V_\star)/m]\!]$. In particular, $\mathbb{E}\left[W_v(V_i)\right] \leq \frac{101}{100}W_v(V_\star)/m \leq \frac{101}{100}\cdot\frac{3}{4}\Delta_\star \leq \Delta_\star$. Since vertices are assigned to machines independently, we can apply Lemma 3.1 to bound the deviation of $W_v(V_i)$ from the expectation:

$$\begin{aligned}\Pr\left(|W_v(V_i)-\mathbb{E}\left[W_v(V_i)\right]| > \Delta_\star^{3/4}\right) &\leq 2\exp\left(-\frac{1}{3}\cdot\left(\frac{1}{\Delta_\star^{1/4}}\right)^2\cdot\Delta_\star\right)\\ &= 2\exp\left(-\frac{1}{3}\Delta_\star^{1/2}\right) \leq 2n^{-21}.\end{aligned}$$

As a result, with probability at least $1 - 2n^{-21}$,

$$\begin{aligned}|W_v(V_i) - W_v(V_\star)/m| &\leq |W_v(V_i) - \mathbb{E}\left[W_v(V_i)\right]| + |\mathbb{E}\left[W_v(V_i)\right] - W_v(V_\star)/m| \\ &\leq \Delta_\star^{3/4} + \epsilon W_v(V_\star)/m \leq \Delta_\star^{3/4} + \epsilon d_v/m \leq \Delta_\star^{3/4} + \frac{3}{4}\epsilon\Delta_\star.\end{aligned}$$

By the union bound, this holds for all v and i of interest—and therefore, Event 4 occurs—with probability at least $1 - |V_\star| \cdot m \cdot 2n^{-21} \geq 1 - n^{-5}$.

To show that Event 5 occurrs with high probability, recall first that $h_{v,i}$ is the expected number of v's neighbors to be added in Line 3 to H_i in the execution of `LocalPhase` for the i-th subgraph. Note that the decision of adding a vertex to H_i is made independently for each neighbor of v. Fix a $v \in V_\star$ and $i \in [m]$ such that $h_{v,i} \leq 2\Delta_\star$. We apply Lemma 3.1 to bound the probability of a large deviation from the expectation:

$$\begin{aligned}\Pr\left(||N(v) \cap H_i| - h_{v,i}| > \Delta_\star^{3/4}\right) &\leq 2\exp\left(-\frac{1}{3} \cdot \left(\frac{1}{2\Delta_\star^{1/4}}\right)^2 \cdot 2\Delta_\star\right) \\ &= 2\exp\left(-\frac{1}{6}\Delta_\star^{1/2}\right) \leq 2n^{-10}.\end{aligned}$$

By the union bound the probability that this bound does not hold for some v and i such that $h_{v,i} \leq 2\Delta_\star$ is by the union bound at most $|V_\star| \cdot m \cdot 2n^{-10} \leq n^{-5}$. Hence, Event 5 occurs with probability at least $1 - n^{-5}$.

In summary, Events 1–5 occur simultaneously with probability at least $1 - 4 \cdot n^{-5} \geq 1 - n^{-4}$ by another application of the union bound. □

Consequences of the random events. We now show that if all the random events occur, then a few helpful properties hold for every vertex v that is not fixed by the constructed configuration $\mathcal{C}$. Namely, v's degree is at most $\frac{3}{4}\Delta$, the number of v's neighbors is similar in all sets R_i is approximately the same, and the number of v's neighbors is similar in all sets H_i.

Claim 3.8. *If Events 1–5 occur for $\epsilon \in [0, (200\ln n)^{-1}]$ and $\frac{\Delta}{m} \geq 4000\mu_R^{-2}\ln^2 n$, then the following properties hold for every vertex $v \in V_\star$ that is not fixed by $\mathcal{C}$:*

1. *$d_v < \frac{3}{4}\Delta$.*
2. *There exists χ_v such that for all $i \in [m]$,*

$$|N(v) \cap R_i| / \mu_R \in \left[\!\left[\chi_v \pm \left(\Delta_\star^{3/4} + \frac{3}{4}\epsilon\Delta_\star\right)\right]\!\right].$$

3. *There exists $\psi_v \in \left[0, \frac{3}{4}\Delta_\star\right]$ such that for all $i \in [m]$,*

$$|N(v) \cap H_i| \in \left[\!\left[\psi_v \pm \alpha\left(\Delta_\star^{3/4} + \epsilon\Delta_\star\right)\right]\!\right].$$

Proof. We use in the proof of the claim the fact that $\Delta_\star = \frac{\Delta}{m} \geq 4000\mu_R^{-2}\ln^2 n = 4 \cdot 10^{15} \cdot \ln^4 n$. To prove the lemma, we fix a vertex v that is not fixed by $\mathcal{C}$. The first property is directly implied by Event 2. Suppose that $d_v \geq \frac{3}{4}\Delta$. Then v is included in the H_i corresponding to the subgraph to which it has been assigned and v is fixed by $\mathcal{C}$. We obtain a contradiction that implies that $d_v < \frac{3}{4}\Delta$.

For the second property, we now know that $d_v < \frac{3}{4}\Delta$. The property follows then directly from Event 3 with $\chi_v \stackrel{\text{def}}{=} d_v/m$.

The last property requires a more complicated reasoning. We set $\psi_v \stackrel{\text{def}}{=} W_v(V_\star)/m < \frac{3}{4}\Delta_\star$. Consider any $i \in [m]$. By Event 4,

$$W_v(V_i) \in \left[\!\!\left[\psi_v \pm \left(\Delta_\star^{3/4} + \frac{3}{4}\epsilon\Delta_\star\right)\right]\!\!\right]. \tag{2}$$

Consider now an arbitrary $u \in V_\star$. We bound the difference between $w_u = \mu_H\left(d_u/\Delta\right)$, which can be seen as the ideal probability of the inclusion in the set of heavy vertices, and $\mu_H\left(\widehat{d}_u/\Delta_\star\right)$, the actual probability of this event in Line 3 of the appropriate execution of `LocalPhase`. Let $\delta_\star \stackrel{\text{def}}{=} \alpha\left(\Delta_\star^{-1/4} + \frac{3}{4}\epsilon\right)$. We consider two cases.

- If $d_u < \frac{3}{4}\Delta$, by Event 3, the monotonicity of μ_H, and Lemma 3.5,

$$\begin{aligned}\mu_H\left(\widehat{d}_u/\Delta_\star\right) &\in \left[\!\!\left[\mu_H\left(\frac{d_u}{\Delta} \pm \left(\Delta_\star^{-1/4} + \frac{3}{4}\epsilon\right)\right)\right]\!\!\right] \\ &\subseteq \left[\!\left[w_u \cdot (1 \pm \delta_\star)\right]\!\right].\end{aligned}$$

  Note that Lemma 3.5 is applied properly because $\Delta_\star^{-1/4} + \frac{3}{4}\epsilon \leq (200\ln n)^{-1} + (200\ln n)^{-1} \leq (48\ln n)^{-1}$.

- If $d_u \geq \frac{3}{4}\Delta$, by Event 1, $\mu_H\left(\widehat{d}_u/\Delta_\star\right) \in \left[1 - \frac{1}{2}n^{-6}, 1\right]$. Concurrently, $w_u \in [\mu_H(3/4), 1] = \left[1 - \frac{1}{2}n^{-12}, 1\right]$. Because $\Delta_\star$ is relatively small, i.e., $\Delta_\star \leq n$,

$$\mu_H\left(\widehat{d}_u/\Delta_\star\right) \in \left[\!\!\left[w_u\left(1 \pm \Delta_\star^{-1/4}\right)\right]\!\!\right] \subseteq \left[\!\left[w_u \cdot (1 \pm \delta_\star)\right]\!\right],$$

  which is the same bound as in the previous case.

It follows from the bound that we just obtained and the definitions of W_v and $h_{v,i}$ that

$$\begin{aligned}h_{v,i} = \sum_{u \in N(v) \cap V_i} \mu_H\left(\widehat{d}_u/\Delta_\star\right) &\in \left[\!\!\left[(1 \pm \delta_\star) \cdot \sum_{u \in N(v)\cap V_i} w_u\right]\!\!\right] \\ &= \left[\!\left[W_v\left(V_i\right) \cdot (1 \pm \delta_\star)\right]\!\right].\end{aligned} \tag{3}$$

We now combine bounds (2) and (3):

$$\begin{aligned}h_{v,i} &\in \left[\psi_v\left(1 - \delta_\star\right) - \left(\Delta_\star^{3/4} + \frac{3}{4}\epsilon\Delta_\star\right)\left(1 + \delta_\star\right), \psi_v\left(1 + \delta_\star\right) + \left(\Delta_\star^{3/4} + \frac{3}{4}\epsilon\Delta_\star\right)\left(1 + \delta_\star\right)\right] \\ &\subseteq \left[\!\!\left[\psi_v \pm \left(\psi_v\delta_\star + \left(\Delta_\star^{3/4} + \frac{3}{4}\epsilon\Delta_\star\right)\left(1 + \delta_\star\right)\right)\right]\!\!\right].\end{aligned}$$

Due to the lower bound on $\Delta_\star$, we obtain $\delta_\star \leq \alpha\left((200\ln n)^{-1} + (200\ln n)^{-1}\right) \leq 1$. This enables us to

simplify and further transform the bound on $h_{v,i}$:

$$\begin{aligned} h_{v,i} &\in \left[\!\!\left[\psi_v \pm \left(\psi_v \delta_\star + 2 \left(\Delta_\star^{3/4} + \frac{3}{4}\epsilon\Delta_\star \right) \right) \right]\!\!\right] \\ &\subseteq \left[\!\!\left[\psi_v \pm \left(\frac{3}{4}\alpha\Delta_\star^{3/4} + \frac{9}{16}\alpha\epsilon\Delta_\star + 2\Delta_\star^{3/4} + \frac{3}{2}\epsilon\Delta_\star \right) \right]\!\!\right] \\ &\subseteq \left[\!\!\left[\psi_v \pm \alpha \left(\frac{4}{5}\Delta_\star^{3/4} + \epsilon\Delta_\star \right) \right]\!\!\right]. \end{aligned}$$

By applying the bound on $\Delta_\star$ again, we obtain a bound on the magnitude of the second term in the above bound:

$$\alpha \left(\frac{4}{5}\Delta_\star^{3/4} + \epsilon\Delta_\star \right) = \alpha \left(\frac{4}{5}\Delta_\star^{-1/4} + \epsilon \right) \Delta_\star \leq 96 \ln n \left(\frac{1}{200 \ln n} + \frac{1}{200 \ln n} \right) \Delta_\star \leq \Delta_\star.$$

This implies that $h_{v,i} \leq \psi_v + \Delta_\star \leq 2\Delta_\star$. The condition in Event 5 holds, and therefore, $||N(v) \cap H_i| - h_{v,i}| \leq \Delta_\star^{3/4}$. We combine this with the bound on $h_{v,i}$ to obtain

$$|N(v) \cap H_i| \in \left[\!\!\left[\psi_v \pm \left(\alpha\frac{4}{5}\Delta_\star^{3/4} + \alpha\epsilon\Delta_\star + \Delta_\star^{3/4} \right) \right]\!\!\right] \subseteq \left[\!\!\left[\psi_v \pm \alpha \left(\Delta_\star^{3/4} + \epsilon\Delta_\star \right) \right]\!\!\right]. \qquad \square$$

Wrapping up the proof of near uniformity. We now finally prove Lemma 3.6. Recall that it states that an ϵ-near uniform $\mathcal{D}$ is very likely to result in a near uniform $\mathcal{D}[\mathcal{C}]$ with a slightly worse parameter and that all vertices not fixed by $\mathcal{C}$ have bounded degree. The proof combines the last two claims: Claim 3.7 and Claim 3.8. We learn that $\mathcal{C}$, the m-configuration constructed in the process is very likely to have the properties listed in Claim 3.8. One of those properties is exactly the property that all vertices not fixed by $\mathcal{C}$ have bounded degree. Hence we have to prove only the near uniformity property. We accomplish this by observing that the probability of $\mathcal{C}$ equal to a specific m-configuration $\mathcal{C}_\star$ with good properties—those in Claim 3.8—does not depend significantly on to which induced subgraph a given vertex v not fixed in $\mathcal{C}_\star$ is assigned. This can be used to show that the conditional distribution of v given that $\mathcal{C} = \mathcal{C}_\star$ is near uniform as desired.

Proof of Lemma 3.6. By combining Claim 3.7 and Claim 3.8, we learn that with probability at least $1 - n^{-4}$, all properties listed in the statement of Claim 3.8 hold for $\mathcal{C}$, the configuration constructed by `EmulatePhase`. Since one of the properties is exactly the same as in the statement of Lemma 3.6, it suffices to prove the other one: that $\mathcal{D}[\mathcal{C}]$ is $60\alpha \left(\Delta_\star^{-1/4} + \epsilon \right)$-near uniform for $\mathcal{C}$ with this set of properties.

Fix $\widetilde{\mathcal{C}} = \left(\{\widetilde{R}_i\}_{i\in[m]}, \{\widetilde{H}_i\}_{i\in[m]}, \{\widetilde{F}_i\}_{i\in[m]} \right)$ to be an m-configuration that has non-zero probability when `EmulatePhase` is ran for $\mathcal{D}$ and has the properties specified by Claim 3.8. Consider an arbitrary vertex $v \in V_\star$. In order to prove the near uniformity of $\mathcal{D}\left[\widetilde{\mathcal{C}}\right]$, we show that v is assigned by it almost uniformly to $[m]$. Let $\mathcal{E}$ be the event that `EmulatePhase` constructs $\widetilde{\mathcal{C}}$, i.e., $\mathcal{C} = \widetilde{\mathcal{C}}$. For each $i \in [m]$, let $\mathcal{E}_{\to i}$ be the event that v is assigned to the i-th induced subgraph. Let $p : [m] \to [0, 1]$ be the probability mass function describing the probability of the assignment of v to each of the m subgraphs in $\mathcal{D}$. Obviously, $p(i) = \Pr[\mathcal{E}_{\to i}]$ for all $i \in [m]$. Due to the ϵ-near uniformity of $\mathcal{D}$, $p(i) = \left[\!\!\left[\frac{1}{m}(1 \pm \epsilon) \right]\!\!\right]$.

For each $i \in [m]$, let $q_i \stackrel{\text{def}}{=} \Pr[\mathcal{E}|\mathcal{E}_{\to i}]$. In order to express all q_i's in a suitable form, we conduct a thought experiment. Suppose v were not present in the graph, but the distribution of all the other vertices in

the modified $\mathcal{D}$ remained the same. Let $q_\star$ be the probability of $\mathcal{E}$, i.e., $\mathcal{C} = \widetilde{\mathcal{C}}$, in this modified scenario. How does the probability of $\mathcal{E}$ change if we add v back and condition on its assignment to a machine i? Note first that conditioning on $\mathcal{E}_{\to i}$ does not impact the distribution of the other vertices, because vertices are assigned to machines independently in $\mathcal{D}$. In order for $\mathcal{E}$ still to occur in this scenario, v cannot be assigned to any of R_i, H_i, or F_i, for which it is considered. Additionally, as long as this the case, v does not impact the behavior of other vertices which only depends on the content of these sets and independent randomized decisions to include vertices. As a result we can express q_i as a product of $q_\star$ and three probabilities: of v not being included in sets R_i, H_i, or F_i.

$$q_i = q_\star \cdot (1-\mu_R) \cdot \left(1 - \mu_H\left(\frac{\left|N(v) \cap \widetilde{R}_i\right| / \mu_R}{\Delta_\star}\right)\right) \cdot \left(1 - \mu_F\left(\frac{\left|N(v) \cap \widetilde{H}_i\right|}{\Delta_\star}\right)\right). \tag{4}$$

Using the properties listed in Claim 3.8, we have

$$\left|N(v) \cap \widetilde{R}_i\right| / \mu_R \in \left[\!\left[\chi_v \pm \left(\Delta_\star^{3/4} + \frac{3}{4}\epsilon\Delta_\star\right)\right]\!\right],$$

and

$$\left|N(v) \cap \widetilde{H}_i\right| \in \left[\!\left[\psi_v \pm \alpha\left(\Delta_\star^{3/4} + \epsilon\Delta_\star\right)\right]\!\right],$$

where χ_v and ψ_v are constants independent of machine i to which v has been assigned and $\psi \le \frac{3}{4}\Delta_\star$. In the next step, we use these bounds to derive bounds on the multiplicative terms in Equation (4) that may depend on i. We also repeatedly use the bounds $\Delta_\star = \frac{\Delta}{m} \ge 4000\mu_R^{-2}\ln^2 n = 4 \cdot 10^{15} \cdot \ln^4 n$ and $\epsilon \le (200\ln n)^{-1}$ from the lemma statement. First, due to Lemma 3.5,

$$\begin{aligned}1 - \mu_H\left(\frac{\left|N(v) \cap \widetilde{R}_i\right| / \mu_R}{\Delta_\star}\right) &\in \left[\!\left[1 - \mu_H\left(\frac{\chi_v}{\Delta_\star} \pm \left(\Delta_\star^{-1/4} + \frac{3}{4}\epsilon\right)\right)\right]\!\right] \\ &\subseteq \left[\!\left[\left(1 - \mu_H\left(\frac{\chi_v}{\Delta_\star}\right)\right) \cdot \left(1 \pm \alpha\left(\Delta_\star^{-1/4} + \frac{3}{4}\epsilon\right)\right)\right]\!\right].\end{aligned}$$

(Note that the application of Lemma 3.5 was correct, because $\Delta_\star^{-1/4} + \frac{3}{4}\epsilon \le (200\ln n)^{-1} + (200\ln n)^{-1} < (96\ln n)^{-1}$.) Second,

$$1 - \mu_F\left(\frac{\left|N(v) \cap \widetilde{H}_i\right|}{\Delta_\star}\right) \in \left[\!\left[1 - \mu_F\left(\frac{\psi_v}{\Delta_\star} \pm \alpha\left(\Delta_\star^{-1/4} + \epsilon\right)\right)\right]\!\right].$$

Since $\psi_v/\Delta_\star \le \frac{3}{4}$ and $\alpha\left(\Delta_\star^{-1/4} + \epsilon\right) \le (96\ln n) \cdot \left((200\ln n)^{-1} + (200\ln n)^{-1}\right) < 1$, the argument to μ_F in the above bound is always less than 4, and therefore, only one branch of μ_F's definitions gets applied. Hence, we can eliminate μ_F:

$$1 - \mu_F\left(\frac{\left|N(v) \cap \widetilde{H}_i\right|}{\Delta_\star}\right) \in \left[\!\left[1 - \frac{\psi_v}{4\Delta_\star} \pm \frac{\alpha}{4}\left(\Delta_\star^{-1/4} + \epsilon\right)\right]\!\right].$$

Since $1 - \frac{\psi_v}{4\Delta_\star} \geq \frac{3}{4}$, we can further transform the bound to

$$1 - \mu_F\left(\frac{\left|N(v) \cap \widetilde{H}_i\right|}{\Delta_\star}\right) \in \left[\!\left[\left(1 - \frac{\psi_v}{4\Delta_\star}\right)\left(1 \pm \frac{\alpha}{3}\left(\Delta_\star^{-1/4} + \epsilon\right)\right)\right]\!\right].$$

Let $\delta_1 \stackrel{\text{def}}{=} \alpha\left(\Delta_\star^{-1/4} + \frac{3}{4}\epsilon\right)$ and $\delta_2 \stackrel{\text{def}}{=} \frac{\alpha}{3}\left(\Delta_\star^{-1/4} + \epsilon\right)$ As a result, every q_i can be expressed as $q_i = \eta_v \lambda_i \lambda_i'$, where η_v is a constant independent of i, $\lambda_i \in [\![1 \pm \delta_1]\!]$, and $\lambda_i' \in [\![1 \pm \delta_2]\!]$. For every i, we can also write

$$\Pr[\mathcal{E} \wedge \mathcal{E}_{\to i}] = \Pr[\mathcal{E}|\mathcal{E}_{\to i}] \cdot \Pr[\mathcal{E}_{\to i}] = \eta_v \lambda_i \lambda_i' \cdot p(i) = \frac{\eta_v}{m} \lambda_i \lambda_i' \lambda_i'',$$

where $\lambda_i'' \in [\![1 \pm \epsilon]\!]$. We now express the conditional probability of v being assigned to the i-th subgraph in $\mathcal{D}$ given $\mathcal{E}$:

$$\Pr[\mathcal{E}_{\to i}|\mathcal{E}] = \frac{\Pr[\mathcal{E} \wedge \mathcal{E}_{\to i}]}{\sum_{j=1}^m \Pr[\mathcal{E} \wedge \mathcal{E}_{\to j}]} = \frac{\lambda_i \lambda_i' \lambda_i''}{\sum_{j=1}^m \lambda_j \lambda_j' \lambda_j''}.$$

Note that for any i, this implies that

$$\frac{1}{m} \cdot \frac{(1-\delta_1)(1-\delta_2)(1-\epsilon)}{(1+\delta_1)(1+\delta_2)(1+\epsilon)} \leq \Pr[\mathcal{E}_{\to i}|\mathcal{E}] \leq \frac{1}{m} \cdot \frac{(1+\delta_1)(1+\delta_2)(1+\epsilon)}{(1-\delta_1)(1-\delta_2)(1-\epsilon)}. \tag{5}$$

Observe that

$$\delta_1 \leq (96 \ln n) \cdot \left((7000 \ln n)^{-1} + (250 \ln n)^{-1}\right) < 1/2,$$

and

$$\delta_2 \leq \frac{1}{3} \cdot (96 \ln n) \cdot \left((7000 \ln n)^{-1} + (200 \ln n)^{-1}\right) < 1/2.$$

Hence all of δ_1, δ_2, and ϵ are at most $1/2$. We can therefore transform (5) to

$$\frac{1}{m} \cdot (1-\delta_1)^2(1-\delta_2)^2(1-\epsilon)^2 \leq \Pr[\mathcal{E}_{\to i}|\mathcal{E}] \leq \frac{1}{m} \cdot (1+\delta_1)(1+\delta_2)(1+\epsilon)(1+2\delta_1)(1+2\delta_2)(1+2\epsilon),$$

and then

$$\frac{1}{m} \cdot (1 - 2\delta_1 - 2\delta_2 - 2\epsilon) \leq \Pr[\mathcal{E}_{\to i}|\mathcal{E}] \leq \frac{1}{m} \cdot (1 + 45\delta_1 + 45\delta_2 + 45\epsilon).$$

Hence

$$\Pr[\mathcal{E}_{\to i}|\mathcal{E}] \in \left[\!\left[\frac{1}{m} \cdot (1 \pm 45(\delta_1 + \delta_2 + \epsilon))\right]\!\right] \subseteq \left[\!\left[\frac{1}{m} \cdot \left(1 \pm 60\alpha\left(\Delta_\star^{-1/4} + \epsilon\right)\right)\right]\!\right],$$

which finishes the proof that $\mathcal{D}\left[\widetilde{\mathcal{C}}\right]$ is $60\alpha\left(\Delta_\star^{-1/4} + \epsilon\right)$-near uniform. □

4 Parallel Algorithm

In this section, we introduce our main parallel algorithm. It builds on the ideas introduced in `EmulatePhase`. `EmulatePhase` randomly partitions the graph into m induced subgraphs and runs on each of them `LocalPhase`, which resembles a phase of `GlobalAlg`. As we have seen, the algorithm performs well even if vertices are assigned to subgraphs not exactly uniformly so long as the assignment is fully independent. Additionally, with high probability, if we condition on the configuration of sets R_i, H_i, and F_i

that were removed, the distribution of assignments of the remaining vertices is still nearly uniform and also independent.

These properties allow for the main idea behind the final parallel algorithm. We partition vertices randomly into m induced subgraphs and then run `LocalPhase` multiple times on each of them with no repartitioning in the meantime. In each iteration, for a given subgraph, we halve the local threshold $\Delta_\star$. This corresponds to multiple phases of the original global algorithm. As long as we can show that this approach leads to finding a large matching, the obvious gain is that *multiple* phases of the original algorithm translate to $O(1)$ parallel rounds. This approach enables our main result: the parallel round complexity reduction from $O(\log n)$ to $O((\log \log n)^2)$.

Algorithm 5: `ParallelAlg`(G, S)
The final parallel matching algorithm

Input:
- graph $G = (V, E)$ on n vertices
- parameter $S \in \mathbb{Z}_+$ such that $S \leq n$ and $S = n^{\Omega(1)}$ (each machine uses $O(S)$ space)

Output: matching in G

1 $\Delta \leftarrow n$, $V' \leftarrow V$, $M \leftarrow \emptyset$
2 **while** $\Delta \geq \frac{n}{S}(200 \ln n)^{32}$ **do**
  `/* High-probability invariant: maximum degree in` $G[V']$ `bounded by` $\frac{3}{2}\Delta$ `*/`
3 $m \leftarrow \left\lfloor \sqrt{\frac{n\Delta}{S}} \right\rfloor$ `/* number of machines used */`
4 $\tau \leftarrow \left\lceil \frac{1}{16} \log_{120\alpha}(\Delta/m) \right\rceil$ `/* number of phases to emulate */`
5 Partition V' into m sets $V_1, \ldots, V_m$ by assigning each vertex independently uniformly at random.
6 **foreach** $i \in [m]$ **do in parallel**
7 If the number of edges in $G[V_i]$ is greater than $8S$, $V_i \leftarrow \emptyset$.
8 **for** $j \in [\tau]$ **do** $(V_i, M_{i,j}) \leftarrow$ `LocalPhase` $\left(i, G[V_i], \Delta/\left(2^{j-1}m\right)\right)$
9 $V' \leftarrow \bigcup_{i=1}^{m} V_i$
10 $M \leftarrow M \cup \bigcup_{i=1}^{m} \bigcup_{j=1}^{\tau} M_{i,j}$
11 $\Delta \leftarrow \Delta/2^\tau$
12 Compute degrees of vertices V' in $G[V']$ and remove from V' vertices of degree at least 2Δ.
13 Directly simulate $M' \leftarrow$ `GlobalAlg`$(G[V'], 2\Delta)$, using $O(1)$ rounds per phase.
14 **return** $M \cup M'$

We present `ParallelAlg`, our parallel algorithm, as Algorithm 5. We write S to denote a parameter specifying the amount of space per machine. After the initialization of variables, the algorithm enters the main loop in Lines 2–11. The loop is executed as long as Δ, an approximate upper bound on the maximum degree in the remaining graph, is large enough. The loop implements the idea of running multiple iterations of `LocalPhase` on each induced subgraph in a random partition. At the beginning of the loop, the algorithm decides on m, the number of machines, and τ, the number of phases to be emulated. Then it creates a random partition of the current set of vertices that results in m induced subgraphs. Next for each subgraph, the algorithm first runs a security check that the set of edges fits onto a single machine (see Line 7). If it does not, which is highly unlikely, the entire subgraph is removed from the graph. Otherwise, the entire subgraph is sent to a single machine that runs τ consecutive iterations of `LocalPhase`. Then the vertices not removed in the executions of `LocalPhase` are collected for further computation and new matching edges are added to the matching being constructed. During the execution of the loop, the maximum degree in the graph induced by V', the set of vertices to be considered is bounded by $\frac{3}{2}\Delta$ with high probability. Once

the loop finishes, we remove from the graph vertices of degree higher than 2Δ—there should be none—and we directly simulate `GlobalAlg` on the remaining graph, using $O(1)$ rounds per phase.

4.1 Preliminaries

Before we analyze the behavior of the algorithm, we observe that the value of $\frac{\Delta}{m}$ inside the main loop is at least polylogarithmic and that the same property holds for the rescaled threshold that is passed to `LocalPhase`.

Lemma 4.1. *Consider a single iteration of the main loop of* `ParallelAlg` *(Lines 2–11). Let Δ and m be set as in this iteration. The following two properties hold:*

- $\Delta/m \geq (200 \log n)^{16}$.
- *The threshold $\Delta/\left(2^{j-1}m\right)$ passed to* `LocalPhase` *in Line 8 is always at least $(\Delta/m)^{15/16} \geq 4000\mu_R^{-2} \ln^2 n$.*

Proof. Let τ be also as in this iteration of the loop. The smallest threshold passed to `LocalPhase` is $\Delta/(2^{\tau-1}m)$. Let $\lambda \stackrel{\text{def}}{=} S\Delta/n$, where S is the parameter to `ParallelAlg`. Due to the condition in Line 2, $\lambda \geq (200 \ln n)^{32}$. Note that $\Delta = \lambda n/S$. Hence $m \leq \sqrt{n\Delta/S} = \frac{n}{S}\sqrt{\lambda}$. This implies that $\Delta/m \geq \sqrt{\lambda} \geq (200 \ln n)^{16}$, which proves the first claim. Due to the definition of τ,

$$2^{\tau-1} \leq (120\alpha)^{\tau-1} \leq (\Delta/m)^{1/16}.$$

This implies that

$$\Delta/(2^{\tau-1}m) \geq (\Delta/m)^{15/16} \geq (200 \ln n)^{15} > 4 \cdot 10^{15} \cdot \ln^4 n = 4000\mu_R^{-2} \ln^2 n. \qquad \square$$

We also observe that the probability of any set of vertices deleted by the security check in Line 7 of `ParallelAlg` is low as long as the maximum degree in the graph induced by the remaining vertices is bounded.

Lemma 4.2. *Consider a single iteration of the main loop of* `ParallelAlg` *and let Δ and V' be as in that iteration. If the maximum degree in $G[V']$ is bounded by $\frac{3}{2}\Delta$, then the probability of any subset of vertices deleted in Line 7 is n^{-8}.*

Proof. Let m be as in the same iteration of the main loop of `ParallelAlg`. Consider a single vertex $v \in V'$. The expected number of v's neighbors assigned to the same subgraph is at most $\frac{3}{2}\Delta/m$. Recall that due to Lemma 4.1, $\frac{\Delta}{m} \geq 200 \ln n$. Since the assignment of vertices to machines is fully independent, by Lemma 3.1 (i.e., the Chernoff bound), the probability that v has more than $2\Delta/m$ neighbors is bounded by

$$2\exp\left(-\frac{1}{3} \cdot \left(\frac{1}{3}\right)^2 \cdot \frac{3}{2} \cdot \frac{\Delta}{m}\right) \leq 2\exp\left(-\frac{1}{18} \cdot 200 \ln n\right) \leq n^{-10}.$$

Therefore, by the union bound, with probability $1 - n^{-9}$, no vertex has more than 2Δ neighbors in the same induced subgraph. As $|V'| \leq n$, the expected number of vertices in each set V_i constructed in the iteration of the main loop is at most $n/m \geq \Delta/m \geq 200 \ln n$. What is the probability that $|V_i| > 2n/m$? By the independence of vertex assignments and Lemma 3.1, the probability of such event is at most

$$2\exp\left(-\frac{1}{3} \cdot \frac{n}{m}\right) \leq 2\exp\left(-\frac{1}{3} \cdot 200 \ln n\right) \leq n^{-10}.$$

Again by the union bound, the event $|V_i| \leq 2n/m$, for all i simultaneously, occurs with probability at least $1 - n^{-9}$. Combining both bounds, with probability at least $1 - 2n^{-9} \geq 1 - n^{-8}$, all induced subgraphs have at most $2n/m$ vertices and the degree of every vertex is bounded by $2\Delta/m$. Hence the number of edges in one induced subgraph is at most $\frac{1}{2} \cdot \frac{2n}{m} \cdot \frac{2\Delta}{m} = \frac{2n\Delta}{m^2}$. By the definition of m and the setting of parameters in the algorithm, $m \geq \frac{1}{2}\sqrt{\frac{n\Delta}{S}}$, where S is the parameter to `ParallelAlg`. This implies that the number of edges is at most $2n\Delta / \left(\frac{1}{2}\sqrt{\frac{n\Delta}{S}}\right)^2 = 8S$ in every induced subgraph with probability $1 - n^{-8}$, in which case no set V_i is deleted in Line 7 of `ParallelAlg`. □

4.2 Matching Size Analysis

The parallel algorithm runs multiple iterations of `LocalPhase` on each induced subgraph, without repartitioning. A single iteration on all subgraphs corresponds to running `EmulatePhase` once. We now show that in most cases, the global algorithm simulates `EmulatePhase` on a well behaved distribution with independently assigned vertices and all vertices distributed nearly uniformly conditioned on the configurations of the previously removed sets R_i, H_i, and F_i. We also show that the maximum degree in the remaining graph is likely to decrease gracefully during the process.

Lemma 4.3. *With probability at least* $1 - n^{-3}$*:*

- *all parallel iterations of* `LocalPhase` *in* `ParallelAlg` *on each induced subgraph correspond to running* `EmulatePhase` *on independent and* $(200 \ln n)^{-1}$*-near uniform distributions of assignments,*
- *the maximum degree of the graph induced by the remaining vertices after the* k*-th simulation of* `EmulatePhase` *is* $\frac{3}{2}\Delta/2^k$*.*

Proof. We first consider a single iteration of the main loop in `ParallelAlg`. Let Δ, τ, and m be set as in this iteration of the loop. For $j \in [\tau]$, let $\Delta_j \stackrel{\text{def}}{=} \Delta / \left(2^{j-1}m\right)$ be the threshold passed to `LocalPhase` for the j-th iteration of `LocalPhase` on each of the induced subgraphs. The parallel algorithm assigns vertices to subgraphs and then iteratively runs `LocalPhase` on each of them. In this analysis we ignore the exact assignment of vertices to subgraphs until they get removed as a member of one of sets R_i, H_i, or F_i. Instead we look at the conditional distribution on assignments given the configurations of sets R_i, H_i, and F_i removed in the previous iterations corresponding to `EmulatePhase`. We write $\mathcal{D}_j$, $1 \leq j \leq \tau$, to denote this distribution of assignments before the execution of j-th iteration of `LocalPhase` on the induced subgraphs, which corresponds to the j-th iteration of `EmulatePhase` for this iteration of the main loop of `ParallelAlg`. Additionally, we write $\mathcal{D}_{\tau+1}$ to denote the same distribution after the τ-th iteration, i.e., at the end of the execution of the parallel block in Lines 6–8 of `ParallelAlg`. Due to Lemma 3.3, the distributions of assignments are all independent. We define ϵ_j, $j \in [\tau+1]$, to be the minimum positive value such that $\mathcal{D}_j$ is ϵ_j-near uniform. Obviously, $\epsilon_1 = 0$, since the first distribution corresponds to a perfectly uniform assignment. We want to apply Lemma 3.6 inductively to bound the value of ϵ_{j+1} as a function of ϵ_j with high probability. The lemma lists two conditions: ϵ_j must be at most $(200 \ln n)^{-1}$ and the threshold passed to `EmulatePhase` has to be at least $4000\mu_H^{-2} \ln^2 n$. The latter condition holds due to Lemma 4.1. Hence as long as ϵ_j is sufficiently small, Lemma 3.6 implies that with probability at least $1 - n^{-4}$,

$$\epsilon_{j+1} \leq 60\alpha \left(\left(\frac{\Delta}{2^{\tau-1}m} \right)^{-1/4} + \epsilon_j \right) \leq 60\alpha \left(\left(\frac{\Delta}{m} \right)^{-15/64} + \epsilon_j \right),$$

and no high degree vertex survives in the remaining graph. One can easily show by induction that if this recursion is satisfied for all $1 \leq j \leq \tau$, then $\epsilon_j \leq (120\alpha)^{j-1} \cdot \left(\frac{\Delta}{m}\right)^{-15/64}$ for all $j \in [\tau+1]$. In particular, by the definition of τ and Lemma 4.1, for any $j \in [\tau]$,

$$\epsilon_j \leq (120\alpha)^{\tau-1} \cdot \left(\frac{\Delta}{m}\right)^{-15/64} \leq \left(\frac{\Delta}{m}\right)^{1/16} \cdot \left(\frac{\Delta}{m}\right)^{-15/64} \leq \left(\frac{\Delta}{m}\right)^{-11/64} \leq (200 \ln n)^{-1},$$

This implies that as long the unlikely events specified in Lemma 3.6 do not occur for any phase in any iteration of the main loop of `ParallelAlg`, we obtain the desired properties: all intermediate distributions of possible assignments are $(200 \ln n)^{-1}$-near uniform and the maximum degree in the graph decreases at the expected rate. It remains to bound the probability of those unlikely events occurring for any phase. By the union bound, their total probability is at most $\log n \cdot n^{-4} \leq n^{-3}$. □

We now prove that the algorithm finds a large matching with constant probability.

Theorem 4.4. *Let M_{OPT} be an arbitrary maximum matching in a graph G. With $\Omega(1)$ probability,* `ParallelAlg` *constructs a matching of size $\Omega(|M_{\mathrm{OPT}}|)$.*

Proof. By combining Lemma 4.2 and Lemma 4.3, we learn that with probability at least $1 - n \cdot n^{-8} - n^{-3} \geq 1 - 2n^{-3}$, we obtain a few useful properties. First, all relevant distributions corresponding to iterations of `EmulatePhase` are independent and $(200 \ln n)^{-1}$-near uniform. Second, the maximum degree in the graph induced by vertices still under consideration is bounded by $\frac{3}{2}\Delta$ before and after every simulated execution of `EmulatePhase`, where Δ is the corresponding. As a result, no vertex is deleted in Lines 7 or 12 due to the security checks.

We now use Lemma 3.2 to lower bound the expected size of the matching created in every `EmulatePhase` simulation. Let $\tau_\star$ be the number of phases we simulate this way. We have $\tau_\star \leq \log n$. Let $\mathbf{H}_j$, $\mathbf{F}_j$, and $\mathbf{M}_j$ be random variables equal to the total size of sets H_i, F_i, and M_i created in the j-th phase. If the corresponding distribution in the j-th phase is near uniform and the maximum is bounded, Lemma 3.2 yields

$$\mathbb{E}\left[\mathbf{H}_j + \mathbf{F}_j\right] \leq n^{-9} + 1200 \cdot \mathbb{E}\left[\mathbf{M}_j\right],$$

i.e.,

$$\mathbb{E}\left[\mathbf{M}_j\right] \geq \frac{1}{1200}\left(\mathbb{E}\left[\mathbf{H}_j + \mathbf{F}_j\right] - n^{-9}\right).$$

Overall, without the assumption that the conditions of Lemma 3.2 are always met, we obtain a lower bound

$$\sum_{j \in [\tau_\star]} \mathbb{E}\left[\mathbf{M}_j\right] \geq \sum_{j \in [\tau_\star]} \frac{1}{1200}\left(\mathbb{E}\left[\mathbf{H}_j + \mathbf{F}_j\right] - n^{-9}\right) - 2n^{-3} \cdot \frac{n}{2},$$

in which we consider the worst case scenario that we lose as much as $n/2$ edges in the size of the constructed matching when the unlikely negative events happen. `ParallelAlg` continues the construction of a matching by directly simulating the global algorithm. Let $\tau'_\star$ be the number of phases in that part of the algorithm. We define $\mathbf{H}'_j$, $\mathbf{F}'_j$, and $\mathbf{M}'_j$, for $j \in [\tau'_\star]$, to be random variables equal to the size of sets H, F, and $\widetilde{M}$ in `GlobalAlg` in the j-th phase of the simulation. By Lemma 2.3, we have

$$\sum_{j \in [\tau'_\star]} \mathbb{E}\left[\mathbf{M}'_j\right] \geq \sum_{j \in [\tau'_\star]} \frac{1}{50}\left(\mathbb{E}\left[\mathbf{H}'_j + \mathbf{F}'_j\right]\right).$$

By combining both bounds we obtain a lower bound on the size of the constructed matching. Let

$$\mathbf{M}_\star \stackrel{\text{def}}{=} \sum_{j\in[\tau_\star]} \mathbb{E}\left[\mathbf{M}_j\right] + \sum_{j\in[\tau'_\star]} \mathbb{E}\left[\mathbf{M}'_j\right]$$

be the expected matching size, and let

$$\mathbf{V}_\star \stackrel{\text{def}}{=} \sum_{j\in[\tau_\star]} \mathbb{E}\left[\mathbf{H}_j + \mathbf{F}_j\right] + \sum_{j\in[\tau'_\star]} \mathbb{E}\left[\mathbf{H}'_j + \mathbf{F}'_j\right].$$

We have

$$\mathbf{M}_\star \geq \frac{1}{1200}\mathbf{V}_\star - \frac{1}{n^2}.$$

Consider a maximum matching M_{OPT}. At the end of the algorithm, the graph is empty. The expected number of edges in M_{OPT} incident to a vertex in one of the reference sets is bounded by $|M_{\mathrm{OPT}}| \cdot 2\mu_R \cdot \log n \leq 10^{-5}|M_{\mathrm{OPT}}|$. The expected number of edges removed by the security checks is bounded by $\frac{n}{2}\cdot n^{-3}$. Hence the expected number of edges in M_{OPT} deleted as incident to vertices that are heavy or are friends is at least $(1-10^{-5})|M_{\mathrm{OPT}}| - 1/(2n^2)$. Since we can assume without the loss of generality that the graph is non-empty, it is at least $\frac{1}{2}|M_{\mathrm{OPT}}|$. Hence $\mathbf{V}_\star \geq \frac{1}{2}|M_{\mathrm{OPT}}|$, and $\mathbf{M}_\star \geq \frac{1}{2400}|M_{\mathrm{OPT}}| - \frac{1}{n^2}$. For sufficiently large n (say, $n \geq 50$), $\mathbf{M}_\star \geq \Omega\left(|M_{\mathrm{OPT}}|\right)$ and by an averaging argument, `ParallelAlg` has to output an $O(1)$-multiplicative approximation to the maximum matching with $\Omega(1)$ probability. For smaller n, it is not difficult to show that at least one edge is output by the algorithm with constant probability as long as it is not empty. □

Finally, we want to argue that the above procedure can be used to compute $2+\epsilon$ approximation to maximum matching at the cost of increasing the running time by a factor of $\log(1/\epsilon)$. The idea is to; execute algorithm `ParallelAlg` to compute constant approximate matching; remove this matching from the graph; and repeat.

Corollary 4.5. *Let M_{OPT} be an arbitrary maximum matching in a graph G. For any $\epsilon > 0$, executing* `ParallelAlg` *on G and removing a constructed matching repetitively, $O(\log(1/\epsilon))$ times, finds a multiplicative $(2+\epsilon)$-approximation to maximum matching, with $\Omega(1)$ probability.*

Proof. Assume that the `ParallelAlg` succeeds with probability p and computes c-approximate matching. Observe that each successful execution of `ParallelAlg` finds a matching M_c of size at least $\frac{1}{c}|M_{\mathrm{OPT}}|$. Removal of M_c from the graph decreases the size of optimal matching by at least $\frac{1}{c}|M_{\mathrm{OPT}}|$ and at most by $\frac{2}{c}|M_{\mathrm{OPT}}|$, because each edge of M_c can be incident to at most two edges of M_{OPT}. Hence, when the size of the remaining matching drops to at most $\epsilon|M_{\mathrm{OPT}}|$, we have an $2+\epsilon$-multiplicative approximation to maximum matching constructed. The number t of successful applications of `ParallelAlg` need to satisfy.

$$\left(1-\frac{1}{c}\right)^t \leq \epsilon.$$

This gives $t = O(\log(1/\epsilon))$. In $\lceil t/p \rceil = O(\log(1/\epsilon))$ executions, we have t successes with probability at least $1/2$ by the properties of the median of the binomial distribution. □

5 MPC Implementation

In this section we present an MPC implementation of our algorithm and analyze its round and space complexity. In the description we heavily use some of the subroutines described in [GSZ11]. While the model used there is different, the properties of the distributed model used in [GSZ11] also hold in the MPC model. Thus, the results carry over to the MPC model.

The results of [GSZ11] allow us to sort a set A of $O(N)$ key-value pairs of size $O(1)$ and for every element of a sorted list, compute its index. Moreover, we can also do a parallel search: given a collection A of $O(N)$ key-value pairs and a collection of $O(N)$ queries, each containing a key of an element of A, we can annotate each query with the corresponding key-value pair from A. Note that multiple queries may ask for the same key, which is nontrivial to parallelize. If $S = n^{\Omega(1)}$, all the above operations can be implemented in $O(1)$ rounds.

The search operation allows us to broadcast information from vertices to their incident edges. Namely, we can build a collection of key-value pairs, where each key is a vertex and the value is the corresponding information. Then, each edge $\{u, v\}$ may issue two queries to obtain the information associated with u and v.

5.1 `GlobalAlg`

We first show how to implement `GlobalAlg`, which is called in Line 13 of `ParallelAlg`.

Lemma 5.1. *Let $S = n^{\Omega(1)}$. There exists an implementation of* `GlobalAlg` *in the MPC model, which with high probability executes $O(\ln \widetilde{\Delta})$ rounds and uses $O(S)$ space per machine.*

Proof. We first describe how to solve the following subproblem. Given a set X of marked vertices, for each vertex v compute $|N(v) \cap X|$. When all vertices are marked, this just computes the degree of every vertex.

The subproblem can be solved as follows. Create a set $A_X = \{(u, v) \mid u \in V, v \in X, \{u, v\} \in E\} \cup \{(v, -\infty), (v, \infty) \mid v \in V\}$, and sort its elements lexicographically. Denote the sorted sequence by Q_X. Then, for each element of A_X compute its index in Q_A.

Note that $|N(v) \cap X|$ is equal to the number of elements in Q_X between $(v, -\infty)$ and (v, ∞). Thus, having computed the indices of these two elements, we can compute $|N(v) \cap X|$.

Let us now describe how to implement `GlobalAlg`. We can compute the degrees of all vertices, as described above. Once we know the degrees, we can trivially mark the vertices in H. The next step is to compute F and for that we need to obtain $|N(v) \cap H|$, which can be done as described above.

After that, `GlobalAlg` computes a matching in $G[H \cup F]$ by calling `MatchHeavy` (see Algorithm 2). In the first step, `MatchHeavy` assigns to every $v \in F$ a random neighbor $v_\star$ in H. This can again be easily done by using the sequence Q_H (i.e. Q_X build for $X = H$). Note that for each $v \in F$ we know the number of neighbors of v that belong to H. Thus, each vertex v can pick an integer $r_v \in [1, |N(v) \cap H|]$ uniformly at random. Then, by adding r_v and the index of $(v, -\infty)$ in Q_H, we obtain the index in Q_H, which corresponds to an edge between v and its random neighbor in H. The remaining lines of `MatchHeavy` are straightforward to implement. The vertices can trivially pick their colors. After that, the set $E_\star$ can be easily computed by transmitting data from vertices to their adjacent edges. Implementing the following steps of `MatchHeavy` is straightforward. Finally, picking the edges to be matched is analogous to the step, when for each $v \in F$ we picked a random neighbor in H.

Overall, each phase of `GlobalAlg` (that is, iteration of the main loop) is executed in $O(1)$ rounds. Thus, by Lemma 2.3, `GlobalAlg` can be simulated in $O(\ln \widetilde{\Delta})$ rounds as advertised. □

5.2 Vertex and edge partitioning

We now show how to implement Line 5 and compute the set of edges that are used in each call to `LocalPhase` in Line 8 of `ParallelAlg`. Our goal is to annotate each edge with the machine number it is supposed to go to. To that end, once the vertices pick their machine numbers, we broadcast them to their adjacent edges. Every edge that receives two equal numbers x is assigned to machine x.
In the implementation we do not check whether a machine is assigned too many edges (Line 7), but rather show in Lemma 4.2 that not too many edges are assigned with high probability.

5.3 `LocalPhase`

We now discuss the implementation of `LocalPhase`. Observe that `LocalPhase` is executed locally. Therefore, the for loop at Line 8 of `ParallelAlg` can also be executed locally on each machine. Thus, we only explain how to process the output of `LocalPhase`.

Instead of returning the set of vertices and matched edges at Line 6 of `LocalPhase`, each vertex that should be returned is marked as `discarded`, and each matched edge is marked as `matched`. After that, we need to discard edges, whose at least one endpoint has been discarded. This can be done by broadcasting information from vertices to adjacent edges. Note that some of the discarded edges might be also marked as `matched`.

5.4 Putting all together

Lines 5, 7 and 8 can be implemented as described in sections 5.2 and 5.3. Lines 9 and 10 do not need an actual implementation, as by that point all the vertices that are not marked as `discarded` constitute V', and all the edges incident to $V \setminus V'$ will be marked as `discarded`. Similarly, all the matched edges will be marked as `matched` by the implementation of `LocalPhase`. All the edges and vertices that are marked as `discarded` will be ignored in further processing. After all the rounds are over, the matching consists of the edges marked as `matched`.

Let $\Delta_\star$ be the value of Δ at Line 12, and hence the value of Δ at the end of the last while loop iteration. Let Δ' be the value of Δ just before the last iteration, i.e. $\Delta_\star = \Delta'/2^\tau$, for the corresponding τ. Now consider the last call of `LocalPhase` at Line 8. The last invocation has $\Delta'/(2^{\tau-1})$ as a parameter. On the other hand, by Claim 3.7 and Claim 3.8 we know that after the last invocation of `LocalPhase` with high probability there is no vertex that has degree greater then $\frac{3}{4}\Delta'/(2^{\tau-1}) < 2\Delta_\star$. Therefore, with high probability there is no vertex that should be removed at Line 12, and hence we do not implement that line either.

An implementation of Line 13 is described in Section 5.1. Finally, we can state the following result.

Lemma 5.2. *There exists an implementation of* `ParallelAlg` *in the MPC model that with high probability executes* $O\left((\log\log n)^2 + \max\left(\log\frac{n}{S}, 0\right)\right)$ *rounds.*

Proof. In the proof we analyze the case $S \leq n$. Otherwise, for the case $S > n$, we think of each machine being split into $\lfloor S/n \rfloor$ "smaller" machines, each of the smaller machines having space n.

We will analyze the number of iterations of the while loop `ParallelAlg` performs. Let Δ_i and τ_i be the value of Δ and τ at the end of iteration i, respectively. Then, from Line 3 and Line 4 we have

$$\tau_i = \left\lceil \frac{1}{16}\log_{120\alpha}\left(\Delta_{i-1}/m\right) \right\rceil \geq \frac{1}{16}\log_{120\alpha}\left(\Delta_{i-1}/m\right) \geq \frac{1}{16}\log_{120\alpha}\sqrt{\frac{S\Delta_{i-1}}{n}}.$$

Define $\gamma := \frac{1}{32\log_2 120\alpha}$. By plugging in the above bound on τ_i, from Line 11, we derive

$$\Delta_i = \Delta_{i-1} \cdot 2^{\tau_i} \leq \Delta_{i-1} \cdot 2^{-\frac{1}{16}\log_{120\alpha}\sqrt{\frac{S\Delta_{i-1}}{n}}} = \Delta_{i-1} \cdot 2^{-\frac{\log_2 \frac{S\Delta_{i-1}}{n}}{32\log_2 120\alpha}} = \Delta_{i-1}^{1-\gamma}\left(\frac{n}{S}\right)^{\gamma} \tag{6}$$

To obtain the number of iterations the while loop of `ParallelAlg` performs, we derive for which $i \geq 1$ the condition at Line 2 does not hold.

Unraveling Δ_{i-1} further from (6) gives

$$\Delta_i \leq \Delta_0^{(1-\gamma)^i}\left(\frac{n}{S}\right)^{\gamma\sum_{j=0}^{i-1}(1-\gamma)^j} \leq n^{(1-\gamma)^i}\left(\frac{n}{S}\right)^{\gamma\frac{1-(1-\gamma)^i}{1-(1-\gamma)}} = n^{(1-\gamma)^i}\left(\frac{n}{S}\right)^{1-(1-\gamma)^i} \tag{7}$$

Observe that $(c\log\log n)^{-1} \leq \gamma \leq (32\log\log n)^{-1} < 1/2$, for an absolute constant c and $n \geq 4$.

For $S \leq n$ and as $\gamma < 1/2$ we have

$$\left(\frac{n}{S}\right)^{1-(1-\gamma)^i} \leq \frac{n}{S}. \tag{8}$$

On the other hand, for $i_\star = \frac{\log\log n}{\gamma} \leq c(\log\log n)^2$ we have

$$n^{(1-\gamma)^{i_\star}} < \log n. \tag{9}$$

Now putting together (7), (8), and (9) we conclude

$$\Delta_{i_\star} < \frac{n}{S}\ln n,$$

and hence the number of iteration the while loop of `ParallelAlg` performs is $O\left((\log\log n)^2\right)$.

Total round complexity. Every iteration of the while loop can be executed in $O(1)$ MPC rounds with probability at least $1 - 1/n^3$. Since there are $O\left((\log\log n)^2\right)$ iterations of the while loop, all the iterations of the loop can be performed in $O\left((\log\log n)^2\right)$ many rounds with probability at least $1 - 1/n^2$.

On the other hand, by Lemma 5.1 and the condition at Line 2 of `ParallelAlg`, the computation of Line 13 of `ParallelAlg` can be performed in $O\left(\log\left(\frac{n}{S}(\ln n)^{32}\right)\right)$ rounds. Putting the both bounds together we conclude that the round complexity of `ParallelAlg` is $O\left((\log\log n)^2 + \log\frac{n}{S}\right)$ for the case $S \leq n$. For the case $S > n$ (recall that in this regime we assume that each machine is divided into machines of memory n) the round complexity is $O\left((\log\log n)^2\right)$. □

References

[ABI86] 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.

[AG15] 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 on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 202–211, 2015.

[ANOY14] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 574–583, 2014.

[BCH16] Sayan Bhattacharya, Deeparnab Chakrabarty, and Monika Henzinger. Deterministic fully dynamic approximate vertex cover and fractional matching in $o(1)$ amortized update time. *CoRR*, abs/1611.00198, 2016.

[BH87] P. Beame and J. Hastad. Optimal bounds for decision problems on the crcw pram. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 83–93, New York, NY, USA, 1987. ACM.

[BHI15] Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. Deterministic fully dynamic data structures for vertex cover and matching. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 785–804, 2015.

[BHN16] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. New deterministic approximation algorithms for fully dynamic matching. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 398–411, 2016.

[BHN17] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. Fully dynamic approximate maximum matching and minimum vertex cover in $O(\log^3 n)$ worst case update time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 470–489, 2017.

[BKS13] 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 2013, New York, NY, USA - June 22 - 27, 2013*, pages 273–284, 2013.

[BKS14] Paul Beame, Paraschos Koutris, and Dan Suciu. Skew in parallel query processing. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 212–223, 2014.

[DG04] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[DG08] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[Edm65] Jack Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, pages 449–467, 1965.

[FG17] Manuela Fischer and Mohsen Ghaffari. Deterministic distributed matching: Simpler, faster, better. *CoRR*, abs/1703.00900, 2017.

[FMS+10] Jon Feldman, S. Muthukrishnan, Anastasios Sidiropoulos, Clifford Stein, and Zoya Svitkina. On distributing symmetric streaming computations. *ACM Trans. Algorithms*, 6(4):66:1–66:19, 2010.

[GSZ11] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the MapReduce framework. In *International Symposium on Algorithms and Computation*, pages 374–383. Springer, 2011.

[HKP99] Michał Hańćkowiak, Michał Karoński, and Alessandro Panconesi. A faster distributed algorithm for computing maximal matchings deterministically. In *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '99, pages 219–228, New York, NY, USA, 1999. ACM.

[HKP01] Michał Hańćkowiak, Michał Karoński, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. *SIAM Journal on Discrete Mathematics*, 15(1):41–57, 2001.

[HRVZ15] Zengfeng Huang, Božidar Radunović, Milan Vojnović, and Qin Zhang. Communication complexity of approximate matching in distributed graphs. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 460–473, 2015.

[IBY+07] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: Distributed data-parallel programs from sequential building blocks. *SIGOPS Oper. Syst. Rev.*, 41(3):59–72, March 2007.

[II86] Amos Israeli and Alon Itai. A fast and simple randomized parallel algorithm for maximal matching. *Inf. Process. Lett.*, 22(2):77–80, 1986.

[IS86] Amos Israeli and Yossi Shiloach. An improved parallel algorithm for maximal matching. *Inf. Process. Lett.*, 22(2):57–60, 1986.

[KKS14] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751, 2014.

[KMW06] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 980–989, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics.

[KSV10] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 938–948, 2010.

[LMSV11] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in MapReduce. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Co-located with FCRC 2011)*, pages 85–94, 2011.

[LPP15] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. *J. ACM*, 62(5):38:1–38:17, November 2015.

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

[OR10] Krzysztof Onak and Ronitt Rubinfeld. Maintaining a large matching and a small vertex cover. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 457–464, 2010.

[PR07] Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theor. Comput. Sci.*, 381(1-3):183–196, 2007.

[RVW16] Tim Roughgarden, Sergei Vassilvitskii, and Joshua R. Wang. Shuffles and circuits: On lower bounds for modern parallel computation. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific Grove, CA, USA, July 11-13, 2016*, pages 1–12, 2016.

[Whi12] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 2012.

[ZCF+10] 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'10, Boston, MA, USA, June 22, 2010*, 2010.