Distributed Approximate Maximum Matching in the CONGEST Model

Mohamad Ahmadi*
University of Freiburg
mahmadi@cs.uni-freiburg.de

Fabian Kuhn*
University of Freiburg
kuhn@cs.uni-freiburg.de

Rotem Oshman
Tel Aviv University
roshman@tau.ac.il

Abstract

We study distributed algorithms for the maximum matching problem in the CONGEST model, where each message must be bounded in size. We give new deterministic upper bounds, and a new lower bound on the problem.

We begin by giving a distributed algorithm that computes an exact maximum (unweighted) matching in bipartite graphs, in $O(n \log n)$ rounds. Next, we give a distributed algorithm that approximates the fractional weighted maximum matching problem in general graphs. In a graph with maximum degree at most Δ, the algorithm computes a $(1-\varepsilon)$-approximation for the problem in time $O\big(\log(\Delta W)/\varepsilon^2\big)$, where W is a bound on the ratio between the largest and the smallest edge weight. Next, we show a slightly improved and generalized version of the deterministic rounding algorithm of Fischer [DISC '17]. Given a fractional weighted maximum matching solution of value f for a given graph G, we show that in time $O((\log^2(\Delta)+\log^* n)/\varepsilon)$, the fractional solution can be turned into an integer solution of value at least $(1-\varepsilon)f$ for bipartite graphs and $(1-\varepsilon)\cdot\frac{g-1}{g}\cdot f$ for general graphs, where g is the length of the shortest odd cycle of G. Together with the above fractional maximum matching algorithm, this implies a deterministic algorithm that computes a $(1-\varepsilon)\cdot\frac{g-1}{g}$-approximation for the weighted maximum matching problem in time $O\big(\log(\Delta W)/\varepsilon^2 + (\log^2(\Delta)+\log^* n)/\varepsilon\big)$.

On the lower-bound front, we show that even for unweighted fractional maximum matching in bipartite graphs, computing an $(1-O(1/\sqrt{n}))$-approximate solution requires at least $\tilde{\Omega}(D+\sqrt{n})$ rounds in CONGEST. This lower bound requires the introduction of a new 2-party communication problem, for which we prove a tight lower bound.

1 Introduction

In the *maximum matching* problem, we are given a graph G, and asked to find a maximum-size set of edges of G which do not share any vertices. In the *weighted* version of the problem, the graph edges are associated with weights, and our goal is to find a set of vertex-disjoint edges that maximizes the total weight. Maximum matching is a fundamental graph optimization problem, extensively studied in the classical centralized setting, as well as in other settings such as streaming algorithms (e.g., [25]) and sublinear-time approximation (e.g., [30]). The problem has also received significant attention from the distributed computing community, so far focusing on approximation algorithms (cf. Section 2).

In this paper we study maximum matching in the CONGEST model, a synchronous network communication model where messages are bounded in size. We consider both exact and approximate maximum matching, weighted and unweighted, and give new upper bounds and a lower bound.

*Supported by ERC Grant No. 336495 (ACDC)

Our upper bounds are deterministic, while the lower bound holds for randomized algorithms as well. Our contributions are as follows.

1.1 Exact Unweighted Maximum Matching in Bipartite Graphs

In the sequential world, the fastest-known algorithm for finding a maximum matching in unweighted bipartite graphs is the Hopcroft-Karp algorithm [18]. Its running time is $O(m \cdot \sqrt{n})$ on graphs with n nodes and m edges. Its central building block is a fast way, using breadth-first-search, to find a maximal set of node-disjoint *augmenting paths*: paths of alternating matching and non-matching edges, used to increase the size of the matching.

A naive implementation of the Hopcroft-Karp algorithm in the CONGEST model would yield an algorithm requiring $O(n^{3/2})$ rounds. Taking inspiration and ideas from Hopcroft-Karp, we are able to instead give an algorithm that takes only $O(n \log n)$ rounds. More specifically, we obtain the following result.

Theorem 1.1. *The deterministic round complexity in the* CONGEST *model of computing an exact maximum matching in unweighted, bipartite graphs is $O(s^* \log s^*)$, where s^* is the size of a maximum matching.*

Note that the algorithm is not assumed to initially know the value s^*.

The core of our algorithm is a procedure that finds a single augmenting path of length k in $O(k)$ rounds. Together with the well-known fact that if we are given a matching of size $s^* - \ell$, we are guaranteed to have an augmenting path of length at most $O(s^*/\ell)$, this procedure implies the above result. To our knowledge, this is the first non-trivial algorithm for exact bipartite maximum matching in the CONGEST model.

1.2 Approximate Fractional Weighted Maximum Matching

One strategy for computing an approximate maximum matching is to first solve the *fractional* version of the problem, and then *round* the solution to obtain an integral matching. A *fractional* matching is the natural linear programming (LP) relaxation of the notion of a matching, where instead of taking a *set* of edges (where each edge is "taken" or "not taken"), we instead assign each edge $e \in E$ a value $y_e \in [0, 1]$. Whereas before, we required that each node participate in at most one edge of the matching, we now require that for each node v, the sum of the values of v's edges must be at most 1. This is a linear constraint: $\sum_{u \in N(v)} y_{\{u,v\}} \leq 1$.

To compute a fractional matching, we can thus bring to bear the powerful machinery of linear programming (LP). In particular, the fractional maximum matching problem is a *packing LP*. Packing LPs and their duals, *covering LPs*, are a class of LPs for which there are particularly efficient distributed solutions (e.g., [20, 27]). In this paper, we extend an approach that was developed by Eisenbrand, Funke, Garg, and Könemann [9] to solve the fractional set cover problem. We prove the following theorem.

Theorem 1.2. *Let $G = (V, E, w)$ be a weighted graph. Assume that Δ is the maximum degree of G, and let W denote the ratio between the largest and smallest edge weight. Then, for every $\varepsilon > 0$, there is a deterministic $O\big(\log(\Delta W)/\varepsilon^2\big)$-time* CONGEST *algorithm to compute a $(1-\varepsilon)$-approximation for the maximum weighted fractional matching problem in G.*

The algorithm is based on another distributed implementation of the algorithm of [9], which appeared in [20]. The algorithm of [20] is general: it approximates general covering and packing LPs. When applied to the weighted fractional matching problem, the algorithm of [20] computes a

$(1-\varepsilon)$-approximation in time $O\big(\log(\Delta W)/\varepsilon^4\big)$, which was the best $(1-\varepsilon)$-approximation for the problem in the CONGEST model prior to the present work.

As we are only interested in the matching problem, our algorithm is simpler than the algorithm of [20], and more importantly, our algorithm significantly improves the ε-dependency of computing a $(1-\varepsilon)$-approximate fractional matching in the CONGEST model.

1.3 Deterministic Rounding of Fractional Matchings

After computing a fractional matching, we wish to *round* the edge values to $\{0,1\}$, to obtain an integral matching with roughly the same weight.

Randomized rounding of LP solutions, in order to obtain approximate solutions of the corresponding integer LPs, has been used for a while, even in the distributed context (e.g., [19, 20]). However, *deterministic* distributed rounding algorithm have only been studied recently. In [11], Fischer gave an amazingly simple and elegant deterministic $O(\log^2\Delta)$-time algorithm, which rounds a fractional unweighted matching into an integral matching that is smaller by only a constant factor. Repeating this rounding step $O(\log n)$ times, Fischer obtains a maximal matching in deterministic time $O(\log^2\Delta\log n)$.[1]

At its core, the approach of Fischer [11] solves the problem on bipartite graphs, and it decomposes the problem of rounding a fractional matching to the problem of rounding fractional matchings on paths and even cycles. Our contribution in this part of the paper is two-fold. First, while Fischer loses a constant factor when rounding the matching, we show that a simple change in the algorithm allows us to only lose a factor $(1-\varepsilon)$ on bipartite graphs. Second, we generalize the technique to also work for weighted (fractional) matching.

Theorem 1.3. *Let $G=(V,E,w)$ be a weighted graph, $\boldsymbol{y}$ be a fractional matching of G, and $\varepsilon>0$ be a parameter. There is a deterministic $O\big(\frac{\log^2(\Delta/\varepsilon)+\log^* n}{\varepsilon}\big)$-time* CONGEST *algorithm that computes an matching M of G such that the ratio between the total weight of M and the value of the given fractional weighted matching $\boldsymbol{y}$ is at least $1-\varepsilon$ if G is bipartite, and at least $(1-\varepsilon)\cdot\frac{g-1}{g}$ if G is not bipartite and g is the length of the shortest odd cycle of G.*

In combination with Theorem 1.2, we obtain a deterministic CONGEST algorithm to compute a $(1-\varepsilon)$-approximate maximum weighted matching in bipartite graphs in time $O\big(\frac{\log(\Delta W)}{\varepsilon^2}+\frac{\log^2(\Delta/\varepsilon)+\log^* n}{\varepsilon}\big)$. For general graphs, we obtain a $(2/3-\varepsilon)$-approximate maximum weighted matching in the same asymptotic time. To the best of our knowledge, this is the first CONGEST algorithm that obtains an approximation ratio better than $1/2$ for the weighted maximum matching problem in general graphs.

1.4 Lower Bound for $(1-O(1/\sqrt{n}))$-Approximate Fractional Matching

As we said above, in this paper we show that a $(1-\varepsilon)$-approximate maximum matching in bipartite graphs can be computed in time $\tilde{O}(1/\varepsilon^2)$ (ignoring the logarithmic terms in n,Δ and W). Is this dependence on ϵ optimal? We do not yet know, but we are able to show that $\tilde{O}(1/\epsilon)$ rounds *are* necessary, for sufficiently small ϵ:

Theorem 1.4. *There exists a constant $\alpha\in(0,1)$, such that any randomized algorithm that computes a $(1-\alpha/\sqrt{n})$-multiplicative approximation to the maximum fractional matching in unweighted, bipartite graphs with diameter $O(\log n)$ requires $\Omega(\sqrt{n}/\log(n))$ rounds.*

[1] Actually, the earlier polylog-time deterministic algorithms for computing a maximal matching [14, 15] can also be interpreted as approximate rounding algorithms. However, these algorithms are not explicitly phrased in this way.

The lower bound is based on the framework of [28], and it is shown by reduction from two-party communication complexity. Given a fast algorithm A for approximate fractional matching, we construct a protocol for two players, Alice and Bob, to solve a communication complexity problem, by simulating the execution of A in a network that the players construct.

In contrast to [28], here we are not interested in a *verification* problem. In [28], in addition to the network graph, there is a set of *marked edges*, and the goal is to check whether the marked edges satisfy some property. Thus, we can give the algorithm a "hard subgraph to check", even if the corresponding *search problem* is easy: e.g., [28] shows that checking if the marked edges form a spanning tree is hard ($\tilde{\Omega}(\sqrt{n} + D)$ rounds), even though *constructing* a spanning tree is easy ($O(D)$ rounds). Here, we do not give the algorithm a set of marked edges, and instead we allow the algorithm to compute any feasible fractional matching.

To prove the lower bound, we argue that a good approximation to the maximum matching on odd paths "looks different" from one on even paths, and this difference allows us to solve a communication complexity problem, PBXA, that we introduce for this purpose. We prove, using information complexity [3], that the randomized communication complexity of PBXA is linear. One unusual feature of this lower bound is that at the end of the simulation, each player only knows part of the matching constructed. Thus, we cannot guarantee that both players will "see" the difference between odd and even paths, but at least one of them will. The problem PBXA reflects this: instead of asking the players to *agree* on an output, each player produces its own output, and at least one of them must "be correct".

2 Related Work

We survey here only the most directly relevant work. In particular, we mostly focus on the CONGEST model, and we discuss only some of the work for the LOCAL model, where messages do not need to be of bounded size.

The first polynomial-time algorithm for unweighted maximum matching in general graphs was given by Edmonds [7, 8]. It was preceded by the algorithm of Hopcroft and Karp [18], which is restricted to bipartite graphs. Our exact algorithm for bipartite graphs is inspired by and uses insights from the Hopcroft-Karp algorithm.

Because exact maximum matching is a "global problem", work on distributed algorithms has mostly been focused on approximation algorithms. The first ones were for the *maximal* matching problem; in the unweighted case, a maximal matching is also a 1/2-approximation to the maximum matching. Even in the 80s, simple and elegant solutions for maximal matching in $O(\log n)$ rounds were known [1, 17, 24]. (These papers give *PRAM* algorithms, but they translate to the CONGEST model easily.) The best randomized distributed algorithm for maximal matching is due to Barenboim et al. [4], and has time complexity $O(\log \Delta + \log^3 \log n)$.

On the deterministic side, maximal matching was first shown to be solvable in polylogarithmic distributed time, $O(\log^4 n)$ rounds, in [14, 15]. While they do not explicitly analyze the message size, we believe that their algorithm can be implemented in the CONGEST model. Currently, the best deterministic algorithm (in CONGEST and LOCAL) is from [11], and requires $O(\log^2 \Delta \log n)$ rounds. As one of our algorithms heavily builds on the techniques of [11], we discuss them in more detail in Section 6. The best lower bound for maximal matching, and more generally, for obtaining constant or polylogarithmic approximations for unweighted maximum matching, is $\Omega\big(\min\big\{\frac{\log \Delta}{\log\log \Delta}, \sqrt{\frac{\log n}{\log\log n}}\big\}\big)$ [21]. The lower bound even holds for randomized algorithms in the LOCAL model.

Beyond the simple 1/2-approximation provided by a maximal matching, there is series of works

on the distributed complexity of obtaining a $(1-\varepsilon)$-approximate maximum cardinality matching. All are based on the framework of Hopcroft and Karp [18], of repeatedly computing a (nearly) maximal vertex-disjoint set of short augmenting paths. The first such algorithm is [22], a randomized CONGEST algorithm with time complexity $O(\log n)$ for every constant $\varepsilon > 0$; however, the dependence on ε is exponential in $1/\varepsilon$. This was recently improved in [2], which gives a randomized algorithm with time complexity $O\big(\operatorname{poly}(1/\varepsilon) \cdot \frac{\log \Delta}{\log \log \Delta}\big)$. Note that the Δ-dependency of the running time matches the lower bound of [21]. There are also deterministic distributed algorithms to obtain a $(1-\varepsilon)$-approximate maximum cardinality matching in polylogarithmic time [6, 10, 12, 13], but they require the LOCAL model.

As for weighted matching, fhe first paper to explicitly study distributed approximation of the weighted maximum matching is [29]. They give a randomized $O(\log^2 n)$-time algorithm with an approximation ratio of $1/5$. This result for the weighted case was later improved in [23] and in [22], which give $O(\log n)$-round randomized CONGEST algorithms with approximation ratios $(1/4-\varepsilon)$ and $(1/2-\varepsilon)$, respectively. In [2], Bar-Yehuda et al. improve the running time and provide a $(1/2-\varepsilon)$-approximation in time $O(\log \Delta / \log \log \Delta)$. The only known polylog-time deterministic CONGEST algorithm for approximate weighted maximum matching in general graphs is the $(2-\varepsilon)$-approximation algorithm by Fischer [11], which runs in $O\big(\log^2 \Delta \cdot \log \frac{1}{\varepsilon}\big)$ rounds.

3 Model and Definitions

Communication model: Our algorithms and lower bounds are designed for the CONGEST model [26]. The network is modeled as a an undirected n-node graph $G = (V, E)$, where each node has a unique $O(\log n)$-bit identifier. Time is divided into synchronous rounds; in each round, each node can send an $O(\log n)$-bit message to each of its neighbors in G. We are interested in the *time complexity* of an algorithm, which is defined as the number of rounds that are required until all nodes terminate.

For simplicity, we assume that all nodes know the maximum degree Δ of G. In all our algorithms, one can replace the value of Δ by a polynomial upper bound, without changing the asymptotic results. We note that at the cost of a slightly more complicated algorithm, the knowledge of n and Δ can also be dropped completely. If the edges of G have weights, we assume that $w_e > 0$ is the weight of edge e. We assume that the weights are normalized such that for all $e \in E$, we have $0 < w_e \leq 1$. We further assume that the nodes know a value W such that the smallest weight is at least $1/W$.

Distributed matching: When we say that a distributed algorithm *computes a matching*, we mean that when the algorithm terminates, each node of the graph knows which of its edges is in the matching (if any). Since the graph is undirected, both endpoints of an edge must agree about whether it is in the matching or not. For fractional matching, each node knows the value of all of its edges, and again, both endpoints of the edge agree on its value.

Notation: Let $G = (V, E)$ be an undirected graph. The ***bipartite double cover*** of G is the graph $G_2 := G \times K_2 = (V \times \{0,1\}, E_2)$, where there is an edge between two nodes (u, i) and (v, j) in E_2 if and only if $\{u, v\} \in E$ and $i \neq j$. Hence, in G_2, every node u of G is replaced by two nodes $(u, 0)$ and $(u, 1)$ and every edge $\{u, v\}$ of G is replaced by the two edges $\{(u, 0), (v, 1)\}$ and $\{(u, 1), (v, 0)\}$. If G is a weighted graph with edge weights w_e for $e \in E$, we assume that the bipartite double cover G_2 is also weighted and each edge of G_2 has the same weight as the underlying edge in G. Throughout the paper, log refers to the logarithm to base 2.

4 Exact Maximum Matching in Bipartite Graphs

Here we present an $O(n \log n)$-round deterministic algorithm to compute a maximum matching for a given n-node bipartite graph B. The algorithm is based on the following observation, which forms the basis for the celebrated Hopcroft-Karp centralized algorithm for maximum matching in bipartite graphs:

Lemma 4.1 ([18]). *Consider a graph G, and let M^* be a maximum matching in G. Then for any positive integer ℓ and any matching M in G, if $|M| \leq (1 - 1/\ell)|M^*|$, then there is an augmenting path of length less than 2ℓ in G w.r.t. M.*

From Lemma 4.1 we get an upper bound on the length of the shortest augmenting path remaining for a matching of given size:

Corollary 4.2. *If the maximum matching in G has size s^*, and M is a matching of size $|M| = i$, then M has an augmenting path of length less than $2\lceil s^*/(s^* - i)\rceil$.*

Proof. We can write:

$$|M| = i = s^* - (s^* - i) = s^* \left(1 - \frac{1}{\frac{s^*}{s^*-i}}\right) \leq s^* \left(1 - \frac{1}{\lceil \frac{s^*}{s^*-i} \rceil}\right).$$

Therefore, by Lemma 4.1, there is an augmenting path of length less than $2\lceil s^*/(s^* - i)\rceil$. □

Note that the length of the shortest remaining augmenting path depends on the size s^* of the maximum matching, which we do not know; therefore we use exponentially-increasing guesses for s^*. If $s \geq s^*$, then $s/(s - i) \geq s^*/(s^* - i)$, so we can safely use s in place of s^* when computing an upper bound on the length of the shortest augmenting path.

The core of our algorithm is a procedure called `SetupPath`: given an upper bound k on the length of the shortest augmenting path, `SetupPath`(k) finds an augmenting path in $O(k)$ rounds. We describe this procedure below, but before showing how we find an augmenting path, let us describe the overall structure of the algorithm.

Let s^* be the size of the maximum matching in G. Our strategy is as follows: we use exponentially-increasing guesses, $s = 1, 2, 4, \ldots, 2^{\lceil \log n \rceil}$, for the size of the maximum matching. For each guess s, we try to construct a maximum matching, under the "assumption" that $s^* \geq s$. At each point we store the largest matching found so far. For each value of s, we start with an empty matching M, and improve it by searching for augmenting paths one-by-one: for each $i = 1, 2, \ldots, n - 1$, we call `SetupPath`$(2\lceil s/(s - i)\rceil)$, spending $O(s/(s - i))$ rounds searching for an augmenting path of length $O(s/(s - i))$; if we find one, we apply it to M to increase its size by at least 1. Note that by Corollary 4.2, if $s^* \geq s$ and $|M| = i$, then indeed there is an augmenting path of length less than $2\lceil s/(s - i)\rceil$. Finally, we compute the size of the resulting matching, replace the previously-stored matching if the new matching is larger, and move on to the next guess s.

For each value of s, the time spent constructing a matching is bounded by:

$$\sum_{i=1}^{n-1} O\left(\frac{s}{s-i}\right) = O(s \log s) = O(s \log n).$$

Therefore, the total running time of the algorithm is $\sum_{t=1}^{\lceil \log n \rceil} O\left(2^t \log n\right) = O\left(n \log n\right)$.

Now let us explain how we find each augmenting path.

4.1 Setting Up an Augmenting Path: Procedure `SetupPath`

Our goal now is, given an upper bound $2k+1$ on the length of the shortest augmenting path, to find an augmenting path, in $O(k)$ rounds. The algorithm uses k rounds to find an augmenting path, another k rounds to inform all nodes on the path, and a final k rounds to confirm that the path was successfully set up.

4.1.1 Finding the path

In this part of the algorithm, free nodes try to propagate their IDs along alternating paths until they "meet" another free node; at this point an augmenting path is detected.

Each node u maintains variables $src_u, pred_u$ which keep track of the source of an alternating path going through u, and the predecessor of u along that path, respectively. Initially, if u is a free node, then $src_u = u$, and otherwise $src_u = \perp$. We also initialize $pred_u = \perp$ (regardless of whether u is free or not).

Each node also has a state, $state_u$, initially set to *active*.

In each round $r = 1, \ldots, k$, each node u with $state_u = active$ does the following:

- If $src_u \neq \perp$:
  - If r is odd, then u sends src_u along all its free edges.
  - If r is even, then u sends src_u along its matched edge, if it has one.

  In both cases, node u sets $state_u \leftarrow inactive$.
- If $src_u = \perp$, and node u receives messages $w_1, \ldots, w_t$ from neighbors $v_1, \ldots, v_t$, respectively, then it sets $src_u \leftarrow \min_i w_i$ and $pred_u$ to be the neighbor that sent $\min_i w_i$ (or an arbitrary one of them, if there is more than one).
- If in the current round node u sent a message to neighbor v, and also received a message from v, then u sets $state_u \leftarrow detected$. Also, node u sets $partner_u$ to the smallest such neighbor v, and it sets $index_u$ to the round number r. Finally, it sets pid_u to $(r, \{src_u, src_v\}, \{u, v\})$ where $v = partner_u$ and src_v is the value node v sent to u in the current round.

Let π_u be the path defined by tracing backwards along the *pred* variables: $\pi_u = w_0, \ldots, w_\ell$, where $w_0 = u$, for each $1 \leq i < \ell$ we have $w_i = pred_{w_{i-1}}$, and $pred_{w_\ell} = \perp$. (Note that because each node only sends out a message in one round, and $pred_v$ is always a neighbor from which v received a token, the *pred*-pointers are acyclic.)

In the sequel, note that each node u sets each variable $pred_u$ or src_u at most once throughout the whole execution. Moreover, the value of any variable x_u for node u at time t is denoted by $x_u(t)$.

Lemma 4.3. *For any $u, v \in V$, if node u sets src_u to v in round t, then v is free, and π_u is an alternating path of length t from u to v.*

Proof. By induction on rounds.

In the first round, only free nodes sends their IDs to their neighbors. Therefore, the only nodes u that set src_u to a value are the ones that are not free but have a neighboring free node. Therefore, for all such nodes u, π_u is an alternating path of length 1, which is the free edge connecting u to its predecessor (i.e., u's neighboring free node with minimum ID).

Suppose the claim holds for round t, and consider round $t+1$. Then, due to the claim assumption, let node u set src_u to a value v received from its neighbor $w = pred_u$ in round $t+1$. Therefore, by the algorithm, w should have set src_w to v in round t. Due to the induction hypothesis, v is free, and π_w is an alternating path of length t from w to v.

Since $\pi_u = u\pi_w$, it remains to show that $u\pi_w$ is an alternating path from u to v. All the nodes in π_w have set their predecessors before round $t+1$. Then, since u sets $pred_u$ as well as src_u in round $t+1$, node u cannot be in π_w, and consequently $u\pi_w$ is a path. Moreover, based on the algorithm description, the edges over which w receives v and the edges over which it sends v have different types (free/matched). Therefore, the first edge in π_w and edge $\{u, w\}$ have different types. Hence, since π_w is an alternating path, π_u should also be an alternating path. □

Lemma 4.4. *Let $2\ell+1$ be the length of the shortest augmenting path. Let v be a free node with the smallest ID that has an augmenting path of length $2\ell+1$, and let u be the smallest-ID free node to which v has an augmenting path P of length $2\ell+1$. Then, at time $\ell+1$ the two endpoints of the middle edge of P detect a shortest augmenting path.*

Proof. Let us name the nodes in P as $\langle v = v_0, v_1, \ldots, v_\ell = u_{\ell+1}, v_{\ell+1} = u_\ell, \ldots, u_1, u_0 = u\rangle$, where $v < u$. In the sequel, we inductively show that for all t, where $0 \leq t \leq \ell$, nodes v_t and u_t respectively send v and u to v_{t+1} and u_{t+1} in round $t+1$. Then, it follows that v_ℓ and u_ℓ detect an augmenting path at time $\ell+1$ by respectively sending v and u to each other in round $\ell+1$. Since v_0 and u_0 are free, based on the algorithm, they respectively send src_{v_0} and src_{u_0} (i.e., their IDs) over their adjacent free edges and in particular to v_1 and u_1 respectively in the first round. Now, regarding the induction hypothesis, let us assume that for an integer t, where $0 < t < \ell$, v_t and u_t respectively send v and u to v_{t+1} and u_{t+1} in round $t+1$. Then, we prove the claim for v_{t+1} and u_{t+1}.

To do so, we first show that $t+1$ is the first round that v_{t+1} and u_{t+1} receive IDs (i.e., their *src* variables have been $\perp$ while receiving in round $t+1$). For the sake of contradiction, let us assume that round r is the first round that v_{t+1} receives an ID, where $r < t+1$. Then, v_{t+1} sets $src_{v_{t+1}}$ in round r, and due to Lemma 4.3, $\pi_{v_{t+1}}$ is an alternating path of length $r < t+1$ from v_{t+1} to some free node. Since edges $\{v_t, v_{t+1}\}$ and $\{v_{t+1}, v_{t+2}\}$ have different types, it is trivial to see that the concatenation of $\pi_{v_{t+1}}$ and one of paths $P[v, v_{t+1}]$[2] or $P[u, v_{t+1}]$ is an alternating walk of length less than $2\ell+1$ starting and ending with free nodes. Without loss of generality, let us assume that the concatenation of $\pi_{v_{t+1}}$ and $P[v, v_{t+1}]$ is an alternating walk and is denoted by P'. Then, if $src_{v_{t+1}}$ is v, it leads to the existence of an odd cycle and contradicts the fact that the graph is bipartite. Otherwise, if $src_{v_{t+1}} \neq v$, all the nodes in $P[v, v_{t+1}]$ have set their *src* variables to v, while all the nodes in $\pi_{v_{t+1}}$ have set their *src* variables to a different value than v. Therefore, $P[v, v_{t+1}]$ and $\pi_{v_{t+1}}$ do not share any node, and consequently P' is an augmenting path of length less than $2\ell+1$, contradicting the choice of ℓ. It is similar to prove the same for u_{t+1}.

Now let us show that among the IDs received by v_{t+1} and u_{t+1} in round $t+1$, the minimum ones are v and u respectively. For the sake of contradiction, let us assume that u_{t+1} receives w that is smaller than u from its predecessor x in round $t+1$. Therefore, x should have set src_x to w at time t. Then, based on Lemma 4.3, π_x is an alternating path of length t from x to w. Considering the induction hypothesis and the choice of x, node u_{t+1} receives tokens over both edges $\{u_t, u_{t+1}\}$ and $\{x, u_{t+1}\}$ in the same round. Therefore, both the edges should have the same type and in particular are free edges. Hence, if $w = v$, the concatenation of π_x, $\{x, u_{t+1}\}$ and $P[v, u_{t+1}]$ is an alternating walk starting and ending with the same free node, i.e., v. This shows the existence of an alternating cycle of odd length, which contradicts the fact that the graph is bipartite. Otherwise, if $w \neq v$, then the concatenation of π_x, $\{x, u_{t+1}\}$ and $P[v, u_{t+1}]$ is a shortest augmenting path of length $2\ell+1$ between v and w, where $w < u$. This contradicts the choice of u. As a result among the IDs received by u_{t+1}, u is the smallest one. It is also similar to prove that among the IDs received by v_{t+1}, v is the smallest one.

[2]For any nodes a and b in a path P, $P[a, b]$ denotes the subpath of P starting with a and ending with b.

We showed that $t+1$ is the first round that v_{t+1} and u_{t+1} receive IDs, and among which v and u are respectively the minimum IDs they receive in that round. On the other hand, by the induction hypothesis, v_t and u_t send v and u to v_{t+1} and u_{t+1} respectively in round t. Then, since $\{u_t, u_{t+1}\}$ and $\{u_{t+1}, u_{t+2}\}$ have different types and u_{t+1} receives IDs over edges with the same type as $\{u_t, u_{t+1}\}$, then by the algorithm, u_{t+1} sends u over $\{u_{t+1}, u_{t+2}\}$ in round $t+2$. The same is similarly true for v_{t+1}. As a result, v_ℓ and u_ℓ send v and u to each other in round $\ell+1$ and detect a shortest augmenting path. □

4.1.2 Informing the nodes on the path

Once an augmenting path is detected, the center edge attempts to recruit all path nodes and set up the augmenting path, by reversing the BFS through following the *pred* variables. We assume some pre-determined ordering $\preceq$ on unordered pairs of node identifiers, which will be used to break ties between augmenting paths of the same length going through the same node. We extend $\preceq$ to $\mathbb{N} \times V^2 \times V^2$ using the lexicographic order.

This phase takes ℓ rounds. In the first round, each node u with state $state_u = detected$ sends a message $(\mathsf{setup}, u, 2, pid_u)$ to src_u. Each node v that receives a $(\mathsf{setup}, w, i, p)$ message checks if $p \preceq pid_v$, and if so, it sets $pid_v \leftarrow p, index_v \leftarrow i, succ_v \leftarrow w$. In each subsequent round, nodes v that have $pid_v \neq \perp$ send a message $(\mathsf{setup}, v, index_v + 1, p)$ to neighbor src_v, and all nodes update $pid, index, succ$ to according to the path with the smallest path-ID they have heard of.

An easy induction, together with Lemma 4.3, shows that at any time t, if variable $pid_u = (\ell, \{w_1, w_2\}, \{w_3, w_4\})$, then there is a length-ℓ augmenting path between w_1, w_2 whose center edge is $\{w_3, w_4\}$.

Lemma 4.5. *Let ℓ, u, v be as in Lemma 4.4, and let $w_0, \ldots, w_{2\ell+1}$ be the augmenting path of length $2\ell+1$ between u and v that minimizes $\{u, v, w_\ell, w_{\ell+1}\}$ with respect to the ordering $\preceq$. Then at each time $0 \leq t \leq \ell$, each node $w \in \{w_{\ell-t}, w_{\ell+t+1}\}$ has $pid_w(t) = (\ell, \{u, v\}, \{w_\ell, w_{\ell+1}\})$, $index_w = t+1$, and if $t > 0$, then $succ_{w_{\ell-t}}(t) = w_{\ell-t-1}(t)$ and $succ_{w_{\ell+t+1}} = w_{\ell+t}$.*

Proof. By induction on time. For $t = 0$ the claim holds trivially. For $t > 0$, the claim is preserved by the fact that nodes propagate the smallest path ID they have heard of, and increment the index; since there does not exist an augmenting path with smaller path ID in the entire graph, the path $(\ell, \{u, v\}, \{w_\ell, w_{\ell+1}\})$ succeeds in propagating. □

Definition 4.1. *We say that an augmenting path $w_0, \ldots, w_{2\ell+1}$ is* properly set up *if for each $0 \leq t \leq \ell$, each node $w \in \{w_{\ell-t}, w_{\ell+t+1}\}$ has $pid_w = (\ell, \{u, v\}, \{w_\ell, w_{\ell+1}\})$, $index_w = t+1$, and if $t > 0$, then $succ_{w_{\ell-t}} = w_{\ell-t-1}$ and $succ_{w_{\ell+t+1}} = w_{\ell+t}$.*

Corollary 4.6. *Let $\ell, u, v, w_0, \ldots, w_{2\ell+1}$ be the nodes from Lemma 4.5. Then following the setup phase, the path $w_0, \ldots, w_{2\ell+1}$ is properly set up.*

4.1.3 Confirming the path

In the previous phase, some augmenting paths may fail to propagate along their entire length, if they are cut off by an augmenting path with a smaller path ID. To erase such partially-set-up paths, we have one last phase, where the endpoints of a successfully-set-up path inform all nodes on the path that it was successfully set up. To do this, any (free) node u with $pid_u \neq \perp$ sends out a message $(\mathsf{confirm}, u, pid_u)$ to its successor $succ_u$ on the path, and the message is forwarded by all nodes on the path (each node sends it to its own successor), if their path ID matches it. After ℓ rounds, any node v that has $pid_u \neq \perp$ but has *not* received a $(\mathsf{confirm}, pid_v)$ from *both* endpoints of the path (which it can determine from the path ID) deletes the path, setting $pid_v \leftarrow \perp$.

Lemma 4.7. *Let $\ell, u, v, w_0, \ldots, w_{2\ell+1}$ be the nodes from Lemma 4.5. Then during the confirmation phase, no node w_i $(0 \leq i \leq 2\ell+1)$ deletes the path $(\ell, \{u,v\}, \{w_\ell, w_{\ell+1}\})$.*

Proof. By induction on time. This claim follows from Corollary 4.6: since the path is properly set up. □

Corollary 4.8. *At the end of the confirmation phase, for any node w, if $pid_w \neq \perp$, then w is part of a properly set up augmenting path.*

4.2 Augmenting the Matching

At the end of procedure `SetupPath`, each node u of the graph knows whether it participates in a properly set-up augmenting path, and if so, its neighbors along the path ($succ_u$ and $pred_u$, or just $succ_u$ if u is an endpoint of the path). We now augment the matching: for an inner node u with neighbors w, z along the path, one of the edges, say $\{u,w\}$, is in the matching, and the other, $\{u,z\}$ is not. Node u now throws $\{u,w\}$ out of the matching, and adds $\{u,z\}$ instead. (Node z will also add $\{u,z\}$ from its side, preserving consistency.) For a free node u, with a neighbor w along the path, we add $\{u,w\}$ to the matching.

5 Fractional Matching Approximation

We first describe a distributed approximation scheme for the weighted fractional matching problem. The algorithm is based on distributed algorithm for general covering and packing linear programs, which appeared in [20]. Further, the distributed algorithm in [20] itself is based on a sequential fractional set cover algorithm by Eisenbrand, Funke, Garg, and Könemann [9].

Reduction to the Bipartite Case: We first show how to reduce the problem of computing a fractional (weighted) matching for a general graph G to the fractional maximum matching problem on two-colored bipartite graphs.

Lemma 5.1. *Let $G = (V, E)$ be a graph with positive edge weights $w_e \geq 0$ for all $e \in E$ and let $H = (V \times \{0,1\}, E_H)$ be the bipartite double cover of G.*

(1) Let $\boldsymbol{x}$ be a fractional matching of G and let $\boldsymbol{y}$ be an edge vector of H such that for every edge $\{(u,i),(v,1-i)\}$ of H, $y_{\{(u,i),(v,1-i)\}} = x_{\{u,v\}}$. Then, $\boldsymbol{y}$ is a fractional matching of H of size $\sum_{e \in E_H} w_e y_e = 2 \cdot \sum_{e' \in E} w_{e'} x_{e'}$.

(2) Let $\boldsymbol{z}$ be a fractional matching of H and let $\boldsymbol{y}$ be an edge vector of G such that for every edge $\{u,v\}$ of G, $y_{\{u,v\}} = (z_{\{(u,0),(v,1)\}} + z_{\{(u,1),(v,0)\}})/2$. Then $\boldsymbol{y}$ is a fractional matching of G of size $\sum_{e \in E} w_e y_e = \frac{1}{2} \cdot \sum_{e' \in E_H} w_{e'} z_{e'}$.

Proof. Follows immediately from the definition of the bipartite double (cf. Section 3). □

In order to compute an approximate fractional matching on a graph G, it therefore suffices to compute an approximate fractional matchign for the bipartite double cover H of G. Note that communication on H can be efficiently simulated on the actual network graph G and in particular, a CONGEST algorithm on H can be run in the CONGEST model on G.

5.1 Distributed Algorithm for 2-Colored Bipartite Graphs

In the following, we assume that we are given a weighted bipartite graph $B = (V_0 \dot\cup V_1, E, w)$ for which the bipartition is given (i.e., a node knows whether it is in V_0 or in V_1). We further define $V := V_0 \cup V_1$ to be the set of all nodes. Using Lemma 5.1, solving the problem on 2-colored bipartite graphs is sufficient to solving the problem on general graphs.

Formulation as a Linear Program: The maximum weighted fractional matching problem can be phrased as a packing linear program (LP). As it will be convenient to describe our algorithm, we use the following non-standard way to describe the maximum matching problem as an LP. Consider some fractional matching $\boldsymbol{z}$ that assigns a value $z_e \geq 0$ to each edge $e \in E$. Instead of directly computing the variables z_e, we make a simple change of variable and we assign a value $y_e \geq 0$ to each node such that $y_e = w_e \cdot z_e$. In terms of the variables y_e, we then obtain the following packing LP:

$$\max \sum_{e \in E} y_e \quad \text{s.t.} \quad \forall v \in V : \sum_{e \in E : v \in e} \frac{y_e}{w_e} \leq 1 \quad \text{and} \quad \forall e \in E : y_e \geq 0. \tag{1}$$

After solving (1), we obtain a weighted fractional machting $\boldsymbol{z}$ of the same quality by setting $z_e := y_e / w_e$ for each edge $e \in E$. The dual covering LP of (1) is defined as follows:

$$\min \sum_{v \in V} x_v \quad \text{s.t.} \quad \forall e = \{u, v\} \in E : \frac{x_u}{w_e} + \frac{x_v}{w_e} \geq 1 \quad \text{and} \quad \forall v \in V : x_v \geq 0. \tag{2}$$

Note that (2) is a variation of the fractional vertex cover LP. We will design an algorithm that solves (2) and (1) at the same time. The algorithm is based on an adaptation of the greedy set cover algorithm (the vertex cover problem is a special case of the set cover problem). It is therefore most natural to think of the algorithm primarily as an algorithm for solving (2).

The Distributed Fractional Matching Algorithm: Our algorithm has a real-valued parameter $\alpha > 1$ and an integer parameter $f \geq 1$. The values of both parameters will be fixed later. Recall that we assume that all edge weights w_e are normalized and the node know a value $W \geq 1$ such that $1/W < w_e \leq 1$ for all edges $e \in E$.

The algorithm maintains a variable $x_v \geq 0$ for each node $v \in V$ and variables $y_e \geq 0$ and $r_e \in [0, 1]$ for each edges $e \in E$. Initially, we set $x_v := 0$, $y_e := 0$, and $r_e := 1$ for all nodes $v \in V$ and all edges $e \in E$. Throughout the algorithm, the values of x_v and y_e only increase and the value of r_e only decreases. We further define a generalized notion of the degree of a node as follows:

$$\forall v \in V \; : \; \gamma(v) := \sum_{e \in E : v \in e} \frac{r_e}{w_e} \quad \text{and} \quad \hat\gamma(v) := \max_{u \in \{v\} \cup N(v)} \gamma(u). \tag{3}$$

As already mentioned, our fractional matching algorithm is a natural adaptation of the fractional set cover algorithm of Eisenbrand, Funke, Garg, and Könemann [9]. The algorithm consists of phases. A node $v \in V$ participates in the next phase as long as $\gamma(v) > 0$. A node terminates as soon as $\gamma(v) = 0$. Algorithm 1 gives the details of a single phase of the fractional matching algorithm.

Before analyzing the algorithm in detail, we make some simple observations. First note that whenever we increase some variable x_v by 1, in line 8, we make sure that the total increase to the edge variables y_e is also equal to 1. The increase of the variables y_e is proportional to their contribution to the generalized node degree $\gamma(v)$. At the end, we therefore have $\sum_{v \in V} x_v = \sum_{e \in E} y_e$.

Further, consider some node $v \in V$ and some edge e that is incident to v. Each time, we increase x_v by 1, we divide r_e by a factor α^{1/w_e}. We set $r_e = 0$ as soon as r_e becomes less than

```
for i ∈ {0,1} do
    for all v ∈ V_i in parallel do
        if γ(v) > 0 then
            θ_v := γ̂(v)/α;
            while γ(v) ≥ θ_v do
                x_v := x_v + 1;
                for all e ∈ E : v ∈ e do
                    y_e := y_e + (r_e/w_e)/γ(v);
                    r_e := r_e/α^{1/w_e};
                    if r_e ≤ α^{-f} then
                        r_e := 0
```

Algorithm 1: A single phase of the fractional matching algorithm

α^{-f} at the end of the algorithm, for every edge $e = \{u, v\}$, we therefore have $x_u + x_v \geq w_e \cdot f$ and thus $\frac{x_u}{w_e} + \frac{x_v}{w_e} \geq f$. Hence, all inequalities of the LP (2) are "over-satisfied" by a factor at least f and we can therefore obtain a feasible solution $\boldsymbol{x'}$ for LP (2) by setting $x'_v := x_v/f$. The solution $\boldsymbol{y}$ for the fractional matching LP (1) is feasible. In order to obtain a feasible solution $\boldsymbol{y'}$, we compute the value $Y_v := \sum_{e \in E: v \in e} \frac{y_e}{w_e}$ for each nodes v and for each edge $e = \{u, v\}$, we set $y'_e := y_e / \max\{Y_u, Y_v\}$. By LP duality, the optimal solutions of (1) and (2) have the same values and we can therfore lower bound the approximation ratio of our fractional matching algorithm by the ratio $f / \max_{v \in V} Y_v \leq 1$. The following lemma and corollary show that for suitable choices of the parameters α and f, this ratio can be made arbitrarily close to 1.

Lemma 5.2. *At the end of running the above fractional weighted matching algorithm, for all nodes $v \in V$, we have*

$$Y_v \leq \frac{\alpha^2}{\alpha - 1} \cdot \big(\ln(W\Delta) + (f+1)\ln\alpha\big).$$

Proof. Let $v \in V$ be some node of the bipartite graph $B = (V_0 \dot\cup V_1, E)$. We study how the value Y_v increases as the value $\gamma(v)$ decreases. Consider some edge $e = \{u, v\} \in E$ that is incident to v such that the variable y_e contributes to the value of Y_v. The variable y_e is increased whenever either x_u or x_v is incremented by 1. Consider such an event and let y_e^+ be the amount by which y_e increases. Similarly, let $Y_v^+ = y_e^+$ be the amount by which Y_v increases through increasing y_e and let r_e^- be the amoung by which r_e decreases. Finally, let $\gamma^-(v)$ be the amount by which $\gamma(v)$ decreases by incrementing either x_u or x_v. Let $w \in \{u, v\}$ be the node for which x_w is incremented by 1. Note that because the nodes in V_0 and V_1 increment their x-variables in separate parts of a phase (cf. the outer two for-loops), it is not possible that x_u and x_v are incremented at the same time. We have

$$y_e^+ = \frac{r_e/w_e}{\gamma(w)} \leq \alpha \cdot \frac{r_e/w_e}{\gamma(v)} = \frac{\alpha^{1/w_e}}{\alpha^{1/w_e} - 1} \cdot \frac{\alpha}{\gamma(v)} \cdot \frac{r_e^-}{w_e} \leq \frac{\alpha^2}{\alpha - 1} \cdot \frac{1}{\gamma(v)} \cdot \frac{r_e^-}{w_e}.$$

The first inequality follows because either $w = v$ or if $w = u$, u only increases its x_u-variable as long as $\gamma(u) \geq \gamma(v)/\alpha$. The second inequality follows because we assume that all edge weights are between 0 and 1 and therefore $\alpha^{1/w_e}/(\alpha^{1/w_e} - 1) \leq \alpha/(\alpha - 1)$ holds because $\alpha > 1$.

Decreasign r_e by an amount of r_e^- decreases the value of $\gamma(v)$ by r_e^-/w_e. As this is true for every edge that is incident to v, we can rewrite the above inequality as

$$Y_v^+ \leq \frac{\alpha^2}{\alpha - 1} \cdot \frac{1}{\gamma(v)} \cdot \sum_{e \in E: v \in e} \frac{r_e^-}{w_e} = \frac{\alpha^2}{\alpha - 1} \cdot \frac{\gamma^-(v)}{\gamma(v)} \leq \frac{\alpha^2}{\alpha - 1} \cdot \int_{\gamma(v) - \gamma^-(v)}^{\gamma(v)} \frac{ds}{s}.$$

The last inequality follows because the function $1/s$ is monotonically decreasing for $s > 0$. The above integral formulation can now easily be extended to the overall increase of Y_v from the beginning of the algorithm (when $Y_v = 0$) to the end of the algorithm. Right before all r_e-variable of the edges incident to v are set to 0 in line 11, the smallest value that $\gamma(v)$ can have is larger than $\alpha^{-(f+1)}$. At the start of the algorithm, the value of $\gamma(v)$ is upper bounded by $W\Delta$ of B. We can therefore upper bound Y_v at the end of the algorithm by

$$Y_v \leq \frac{\alpha^2}{\alpha-1} \cdot \int_{\alpha^{-(f+1)}}^{W\Delta} \frac{ds}{s} = \frac{\alpha^2}{\alpha-1} \cdot \big(\ln(W\Delta) + (f+1)\ln\alpha\big). \qquad \square$$

Corollary 5.3. *Let $\varepsilon \in (0, 1/2]$ be a parameter. By choosing $\alpha = 1 + \varepsilon/c$ and $f = 2c \cdot \ln(\Delta W)/\varepsilon^2$ for a sufficiently large constant c, the above fractional matching algorithm can be used to compute a $(1-\varepsilon)$-approximate fractional weighted matching in an arbitrary weighted graph $G = (V, E)$.*

Proof. We first use Lemma 5.1 to transform the fractional matching problem on an arbitrary graph G to the fractional matching problem on a 2-colored bipartite graph $B = (V_0 \dot\cup V_1, E_B)$. As discussed above, the approximation ratio of the algorithm is at least $f/\max_{v\in V} Y_v$. By Lemma 5.2, we have

$$\frac{\alpha^2}{\alpha-1} \cdot (\ln(W\Delta) + (f+1)\ln\alpha) \leq e^{2\varepsilon/c} \cdot \left(\frac{\ln(W\Delta)}{\varepsilon/c} + f + 1\right).$$

By choosing c sufficiently large, the above expression is upper bounded by $1 + \varepsilon \geq 1/(1-\varepsilon)$. $\square$

It remains to bound the time complexity of the algorithm in the distributed setting.

Lemma 5.4. *The described fractional weighted matching algorithm can be implemented in $O(f + \log_\alpha(W\Delta))$ rounds in the* CONGEST *model.*

Proof. We first show that a single phase of the algorithm (as described in Alg. 1) can be implemented in $O(1)$ rounds and we then upper bound the required number of phases.

To see the time required to execute a single phase, note that we increase the x-variables of nodes in V_0 and V_1 separately. Consequently, whenever we increase x-variables in parallel, for each edge $\{u, v\} \in E$, at most one of the nodes u and v increases its x-variable. Hence, inside the while loop starting in line 5, node v has complete control over the change to y_e and r_e for its incident edges. The iterations of the while loop can therefore be carried out without communication. The nodes only need to communicate the final changes to the variables y_e and r_e at the end of the while loop. A single phase can therefore be executed in $O(1)$ rounds in the CONGEST model.

To bound the number of phases, we define $\Gamma := \max_{v\in V} \gamma(v)$. Note that in each phase Γ decreases by at least a factor of α. As discussed above (in the proof of Lemma 5.2, at the beginning $\Gamma \leq W\Delta$ and at the very end, before each $\gamma(v) = 0$, we have $\Gamma \geq \alpha^{-(f+1)}$. The required number of phases is therefore $O(\log_\alpha(W\Delta) + f)$. $\square$

Together with Corollay 5.3, Lemma 5.4 directly proves Theorem 1.2.

6 Deterministic Rounding of Fractional Matchings

For rounding the obtained fractional matching from Section 5, we adapt the technique by Manuela Fischer in [11]. In [11], Fischer shows how to round a fractional matching to an integral matching at the cost of losing a non-trivial constant factor (in the unweighted and in the weighted case). We show that a simple adaptation of the algorithm allows to keep the loss within a $(1+\varepsilon)$-factor in

the unweighted bipartite case. We further show that the method can also be generalized to the weighted bipartite case while only losing a $(1+\varepsilon)$-factor in the rounding.

Normalizing the Fractional Matching: As for the fractional maximum matching problem in Section 5, we first solve the problem in 2-colored bipartite graphs, and we then show how to extend the solution to general graphs. The following lemma further shows that we can assume that we start with a *normalized* fractional matching where all the fractional edge values are of the form 2^{-i} for some integer $i \geq 0$.

Lemma 6.1. *At the cost of at most an ε-fraction of an optimal matching, the problem of rounding a weighted fractional matching $\boldsymbol{y}$ of a graph G with maximum degree Δ can be reduced to the problem of rounding a weighted fractional matching $\boldsymbol{y}'$ on a multigraph G' such that for all edges e of G', we have $\boldsymbol{y}'_e = 2^{-i}$ for some non-negative integer $i = O\big(\log(\Delta/\varepsilon)\big)$.*

Proof. Let k be the smallest integer such that $2^k \geq \log(\Delta/\varepsilon)$. Given a weighted graph $G = (V, E)$, we construct a weighted multigraph $G' = (V, E')$ as follows. G' is obtained from G by replacing each edge $e \in E$ with $k+1$ identical copies $e_0, \ldots, e_k$ of e. If e has weight w_e in G, we also set $w_{e_i} = w_e$ for the corresponding $k+1$ parallel edges in G'.

Assume that we are given a fractional matching $\boldsymbol{y}$ assigning a fractional value $y_e \in [0,1]$ to each edge $e \in E$. We obtain a fractional matching $\boldsymbol{z}$ of G' as follows. For each edge $e \in E$, we define $y'_e := \lfloor 2^k \cdot y_e \rfloor / 2^k$. Note that $y'_e \leq y_e$ and thus $\boldsymbol{y}'$ is a valid fractional matching of G. Note further that y'_e can be written as $y'_e := \sum_{i=0}^{k} \beta_i \cdot 2^{-i}$, where $\beta_i \in \{0,1\}$. We define $z_{e_i} := \beta_i \cdot 2^{-i}$. Clearly, $\boldsymbol{z}$ is a fractional matching of G'. The total value of the fractional weighted matching is $\sum_{e\in E} w_e y'_e \geq \sum_{e\in E} w_e (y_e - 2^{-k}) \geq \sum_{e\in E} w_e y_e - \frac{\varepsilon}{\Delta} \cdot \sum_{e\in E} w_e$. The claim of the lemma now follows because an optimal fractional weighted matching of G (and also G') has value at least $\frac{1}{\Delta} \cdot \sum_{e\in E} w_e$. □

Basic Rounding Strategy: We use the same basic rounding approach as Fischer [11]. In the following, we assume that we are given a biparite (multi-)graph $B = (V_0 \dot\cup V_1, E)$ and a normalized fractional matching $\boldsymbol{y}$ that assigns a value $y_e = 2^{-i}$ for some integer $i \geq 0$ to each edge $e \in E$. For convenience, let E_i be the set of edges $e \in E$ for which $y_e = 2^{-i}$ and let $B_i := (V_0 \dot\cup V_1, E_i)$ be the subgraph of B induced by the edges in E_i. Assume further that k is the largest integer such that $E_k \neq \emptyset$, i.e., for which there is some edges $e \in E$ with $y_e = 2^{-k}$. For a given parameter $\delta > 0$, we describe a rounding algorithm that rounds each edge $e \in E_k$ either to value 0 or to value $2^{-(k-1)}$ such that the total value of the fractional (weighted) matching does not decrease by more than a factor $1-\delta$.

In order to do the rounding of the edges in E_k, we define a virtual graph B'_k as follows. For each node $v \in V$, let $d_k(v)$ be the number of edges in E_k that are incident to v. If $d_k(v) \geq 1$, we create $s_v := \lceil d_k(v)/2 \rceil$ virtual nodes $v_1, \ldots, v_{s_v}$ and we arbitrarily divide the $d_k(v)$ edges in E_k that are incident to v among the nodes $v_1, \ldots, v_{s_v}$ such that each node v_i receives at most two such edges (i.e., if $d_k(v)$ is even, all virtual nodes v_i get two edges and if $d_k(v)$ is odd, one of the virtual nodes gets one edge and the others get two edges). Note that the graph B'_k has maximum degree 2 and because B'_k is bipartite, it means that it consists of disjoint paths and even cycles. The next lemma shows that we can use an arbitrary matching of B'_k to select the set of edges in E_k, which are rounded up to value $2^{-(k-1)}$.

Lemma 6.2. *Let M'_k be a matching of the graph B'_k and let M_k the corresponding subset of edges of E_k. Further, let $\boldsymbol{y}'$ be obtained from the fractional matching $\boldsymbol{y}$ of B by setting $y'_e = y_e$ for all $e \notin E_k$, $y'_e = 2y_e$ for all $e \in M_k$ and $y'_e = 0$ for all $e \in E_k \setminus M_k$. Then $\boldsymbol{y}'$ is a valid fractional matching of B.*

Further, if the total weight of M'_k is at least $(1-\delta)/2$ of the total weight of B'_k for some $\delta \geq 0$, the total weight of $\boldsymbol{y}'$ is at most a $(1-\delta)$-factor smaller than the total weight of $\boldsymbol{y}$.

Proof. The second part of the lemma follows immediately because we round up (i.e., double) the fractional matching values corresponding to a $(1-\delta)/2$-fraction of the total weight of B'_k

We need to show that at each node $v \in V$, the sum of the fractional matching values y'_e of the edges e incident to v does not exceed 1. Consider the $s_v = \lceil d_k(v)/2 \rceil$ virtual nodes v_i in B'_k. For each of the $\lfloor d_k(v)/2 \rfloor$ virtual nodes of degree 2, at most one of its incident edges is rounded up to $2^{-(k-1)}$ and at least one of its incident edges is rounded down to 0. Hence, for those virtual nodes, the total sum of the values y'_e is upper bounded by the total sum of the corresponding values y_e. If $d_k(v)$ is odd, there is one virtual node v_i of degree 1 in B'_k and the edge incident to v_i might be rounded up to $2^{-(k-1)}$. Hence, if $d_k(v)$ is odd, we can only guarantee that

$$\sum_{e \in E: v \in e} y'_e \leq (2^{-(k-1)} - 2^{-k}) + \sum_{e \in E: v \in e} y_e = 2^{-k} + \sum_{e \in E: v \in e} y_e.$$

We need to show that this value does not exceed 1. Recall that for all $e \in E$, we have $y_e = 2^{-i}$ for some $i \in \{0, \dots, k\}$. The sum $\sum_{e \in E: v \in e} y_e$ is therefore an integer multiple of 2^{-k} and since $d_k(v)$ is odd, we have $\sum_{e \in E: v \in e} y_e = a \cdot 2^{-k}$ for an odd integer $a > 0$. As $\boldsymbol{y}$ is a valid fractional matching of B, we can therefore conclude that $\sum_{e \in E: v \in e} y_e \leq 1 - 2^{-k}$, which proves the claim of the lemma. □

We note that the above rounding of edges is the main difference between the approach of [11] and our algorithm. In [11], to be on the safe side, the fractional matching value of the edge incident to a virtual node of degree 1 is always rounded down unless the total fractional matching value at the respective node is far from 1. The small and simple change makes the rounding more efficient and it will later also make it easier to argue that the rounded matching is not much smaller than the original matching. The more general rounding provided by Lemma 6.2 will in particular make it much easier to round weighted fractional matchings. Lemma 6.2 implies that rounding fractional matchings to integral matchings essentially boils down to computing almost maximum (weighted) matchings in graphs of maximum degree 2 (i.e., in paths and cycles).

Approximating Maximum Matching in Paths and Cycles: As discussed above, Lemma 6.2 essentially reduces the problem of rounding (weighted) fractional matchings to solving the weighted maximum matching problem in paths and cycles.

Lemma 6.3. *Let $G = (V, E)$ be a weighted n-node graph with maximum degree 2 and assume that W is the total weight of all edges of G. Let $\delta > 0$ be a parameter, and let g be the length of the shortest odd cycle of G.[3] In the* CONGEST *model, a matching of weight at least $\frac{g-1}{g} \cdot (1-\delta) \cdot W/2$ can be computed in time $O\big(\frac{1}{\delta} \cdot \log^* n\big)$.*

If $G = (V_0 \dot\cup V_1, E)$ is a bipartite graph for which the bipartition (V_0, V_1) is given, there is $O(1/\delta)$-time algorithm that computes a matching of total weight at least $(1-\delta)W/2$.

Proof. We can clearly compute an approximate maximum matching independently for each component of G. Let us therefore consider a single component $H = (V_H, E_H)$ of G of size k. The graph H is either a path of length k or a cycle of length k. Note first that if $k = O(1/\delta)$, we can compute an optimal weighted matching in $k = O(1/\delta)$ rounds in the CONGEST model. By using pipelining (in both directions), each node of H can learn the whole structure of H in k rounds. Note that every weighted path and every weighted even cycle has a matching of total weight at least half of the total weight of the path or cycle. For an odd cycle of total weight W, we get a matching of

[3] Note that if G is bipartite, we have $g = \infty$.

weight $(g-1)/2g \cdot W$ by first removing an adge of smallest weight and then computing an optimal weighted matching of the remaining path.

Let us therefore assume that $k \geq C/\delta$ for a sufficiently large constant C. For an integer ℓ, $1 \leq \ell < k$, we define an edge e of H to be *ℓ-light* if there exists a subpath P of H of length ℓ such that $e \in P$ and such that w_e is at most a $1/\ell$-fraction of the total weight of P. Note that every subpath of length 2ℓ of H has at least one ℓ-light edge and thus two ℓ-light edges can be separated by at most 2ℓ hops. Note further that in unweighted graphs, every edge is ℓ-light (for every ℓ).

The idea of approximating maximum weighted matching in long paths and cycles is to select a sufficiently sparse set of light edges, remove the edges from H and solve the maximum matching problem optimally on the remaining paths. To prove the first part of the lemma, we set $\ell := \lceil 1/\delta \rceil$ and we compute a set L of ℓ-light edges satisfying the following properties: Any two edges in L are separated by at least 2ℓ hops and all components (paths) of $H \setminus L$ are of size $O(\ell)$. We can for example compute a maximal subset L of the ℓ-light edges of H such that any two edges in L are separated by at least 2ℓ hops. Such a set L can be computed in time $O(\ell \log^* n) = O(\log^*(n)/\delta)$ by using standard methods based on the techniques in [5]. Let W_L be the total weight of the edges in L. By computing (in time $O(1/\delta)$) an optimal weighted matching for all the paths we get after removing the edges in L, we obtain a matching of weight at least $(W - W_L)/2$. Because the edges in L are sufficiently separated, we know that $W_L \leq W/\ell = \delta W$. The resulting matching thus has size at least $(1-\delta)W/2$.

For the second part of the lemma, we generalize a technique that has been previously been used in [11] and [16] for unweighted matchings. As before, we can restrict the attention to a single path or cycle H of length at least $k \geq C/\delta$ for a sufficiently large constant C. Let U_0 and U_1 be the two color classes of the given 2-coloring of H. In the following, we will orient each edge of H either from U_0 to U_1 or vice versa. For convenience, let us call an edge *blue* if it is oriented from U_0 to U_1 and let us call an edge *red* if it is oriented from U_1 to U_0. We will maintain the total weight of all blue edges is always at least as large as the total weight of all red edges. Let E_L be the set of ℓ-light edges for $\ell := \lceil 4/\delta \rceil$. Initially, we color all edges in E_L blue, i.e., we orient all these edges from U_0 to U_1. The remaining edges form paths of length at most $O(\ell) = O(1/\delta)$. In time $O(1/\delta)$, we can alternatingly color these paths red and blue such that the total weight of the blue edges of each of these paths is at least as large as the total weight of the red edges. Let $\mathcal{P}$ be the maximal consistently oriented subpaths of H (i.e., $\mathcal{P}$ is the set of maximal alternatingly colored paths). We proceed in phases $i = 0, 1, 2, \ldots, \lceil \log 2\ell \rceil$. In phase i, we only work on the maximal consistently oriented subpaths of length at most 2^i. The objective of phase i is to guarantee that at the end of the phase, there are no two adjacent maximal subpaths of length at most 2^i. We prove how to achieve this guarantee by induction on i. Clearly, at the beginning of phase 0, there are no two adjacent subpaths of length at most $2^{-1} < 1$. It is therefore sufficient to prove that if at the beginning of a phase i, there are no two adjacent subpaths of length at most 2^{i-1}, we can guarantee that at the end of the phase, there are no two adjacent subpaths of length at most 2^i.

Let us now decribe phase i. Let $\mathcal{P}_i$ be the set of subpaths of length at most 2^i. We pair adjacent paths of $\mathcal{P}_i$ as follows. We pair any two ajacent paths that are oriented towards each other. This guarantees that for every sequence of adjacent paths in $\mathcal{P}_i$, all paths except possibly the paths at the beginning and the end of the sequence are paired. In each pair of paths, we reverse the direction of one of the paths. We can always do this such that the total weight of the blue edges is still at least as large as the total value of the red edges. Because at the beginning of the phase, there are no two adjacent subpaths of length at most 2^{i-1}, each of the pairs of paths has a total length that exceeds 2^{i-1}. Hence, if we now again consider the new set of subpaths of length at most 2^i, in any sequence of adjacent such paths, all the inner paths are of length more than 2^{i-1}. Further, if the first or last path of such a sequence is shorter, then it is oriented towards the outside

(towards the neighboring long path). If we do the same kind of pairing again and path reversal again, we therefore get to a situation, where there are no two adjacent subpaths of length at most 2^i. The total time to implement phase i in the CONGEST model is linear in 2^i. The total time to implement all $\lceil \log 2\ell \rceil$ phases is therefore linear in ℓ. At the end, there are not two adjacent maximal consistently oriented subpaths of length at most 2ℓ.

We now determine the matching of H as follows. First note that there cannot be more than three adjacent blue edges because this would imply two adjacent maximal consistently oriented subpaths of length 1. Further, for any node v for which both incident edges are not ℓ-light, the two edges are consistently oriented at the beginning and because we always reorient maximal consistently oriented paths, they remain consistently oriented to the end. Hence, one of these edges needs to be blue and the other one needs to be red. Therefore, whenever there are two adjacent blue edges, one of these edges needs to be a light edge. Let us therefore define the matching of H to consist of all the blue edges, except for some ℓ-light blue edges. Among any two adjacent blue edges, we remove an arbitrary ℓ-light one. Further, if there are three consecutive blue edges, if the middle edge is ℓ-light, we remove the middle edge and otherwise, we remove the two outer edges (both of them have to be ℓ-light in this case). Because there are no two adjacent maximal subpaths of length at most 2ℓ, on any subpath of H of length 2ℓ, there are at most 2 ℓ-light blue edges that are removed. By the definition of ℓ-light edges, the total weight that we remove from the blue edges is at most $2/\ell$-fraction of the total weight W of H. The total weight of the matching is therefore at least $W/2 - 2W/\ell \geq (1-\delta)W/2$. □

Putting the Pieces Together: We now have all the tools that are needed for the rounding and we can therefore prove Theorem 1.3.

Proof of Theorem 1.3. First of all, we assume that $\boldsymbol{y}$ is at least a 1/3-approximation. If not, one can directly apply the weighted $(2+\varepsilon)$-approximation algorithm of [11] to obtain the claim of the theorem. Because $\boldsymbol{y}$ is at least a 1/3-approximation and because the optimal fractional matching size is at least $\sum_{e\in E} w_e/\Delta$, we directly round down matching values that are smaller than $\varepsilon/(12\Delta)$, i.e., if $y_e \leq \varepsilon/(12\Delta)$, we set $y_e := 0$. This reduces the value of the weighted fractional matching $\boldsymbol{y}$ at most by a factor $(1-\varepsilon/4)$.

Using Lemma 5.1, we now first move to the bipartite double cover of G and by using Lemma 6.1, we create a multi-graph in which all matching values are negative powers of 2. Assume that the smallest matching value is 2^{-k}. Because all matching values of $\boldsymbol{y}$ are at least $\varepsilon/(12\Delta)$, we have $k = O(\log(\Delta/\varepsilon))$. We apply k iterations of the basic rounding, each time, we round the edges of the currently smallest values. In order to lose at most another $(1-\varepsilon/4)$-factor throughout the k phases of rounding, we make sure that in each of the k iterations, we only lose a $(1-O(\varepsilon/k))$-factor. In Lemma 6.2, we therefore have to set $\delta = O(\varepsilon/k) = O(\varepsilon/\log(\Delta/\varepsilon))$. Because B'_k is a 2-colored bipartite graph, Lemma 6.3 implies that the matching of B'_k which is necessary by Lemma 6.2 can be computed in time $O(1/k) = O(\log(\Delta/\varepsilon)/\varepsilon)$. After the k steps of rounding, we therefore obtain a matching of the bipartite double cover H of G of size at least $(1-\varepsilon/2)$ times the value of the given fractional matching of H. When using Lemma 5.1 to transform this matching back to G, we only obtain a fractional matching of G. However, this fractional matching is half-integral and rounding it to an integer matching can therefore be achieved by another application of Lemma 6.3. However, this time, we do not have a 2-coloring of the graph and G might also not be bipartite. The time for this last rounding step is therefore $O(\log^* n/\varepsilon)$ and we lose a factor $(1-O(\varepsilon)) \cdot \frac{g-1}{g}$. □

7 Lower Bound

7.1 The 2-Player Problem

Let XOR-to-And, or XA for short, be the following problem: the players receive input bits $x, y \in \{0,1\}$, respectively, and their goal is to output bits $a, b \in \{0,1\}$, respectively, such that $a \wedge b = x \oplus y$. That is, if $x \oplus y = 1$, then both players should output 1, but if $x \oplus y = 0$, then at least one player should output 0.

For $n \geq 1$, let $\mathsf{PXA}_{n,\delta}$ be the following problem: the players are given n copies of XA, $x_1, y_1, \ldots, x_n, y_n$, with the *promise* that for at least $n/4$ copies i we have $x_i \oplus y_i = 1$, and for at least $0.49n$ copies i we have $x_i \oplus y_i = 0$. The goal is to solve each copy with *marginal success probability* δ: the players should produce outputs $\mathbf{a}_1, \mathbf{b}_1, \ldots, \mathbf{a}_n, \mathbf{b}_n$, which we think of as random variables depending on the protocol's internal randomness, such that for each $i \in n$ we have $\Pr[\mathbf{a}_i \wedge \mathbf{b}_i = x_i \oplus y_i]$.

In Section 7.6 below, we show that the randomized communication complexity of $\mathsf{PXA}_{n,\delta}$ is $\Omega((1-\delta)n)$. However, first we give a reduction from $(1-\epsilon)$-approximate maximum fractional matching in graphs of $O(n)$ vertices to $\mathsf{PXA}_{\Theta(n/\epsilon),\delta}$, with $\delta \in (1/2, 3/5)$ a constant that will be fixed later.

7.2 The Lower Bound Graph

Fix a parameter $k \geq 40\epsilon n$, and assume for simplicity that n/k is an integer.

Given inputs $x, y \in \{0,1\}^k$, Alice and Bob construct a graph $G_{x,y} = (V, E_{x,y})$, consisting of

- k paths, each of length $2n/k$, denoted $\pi_0, \ldots, \pi_{k-1}$, where $\pi_i = (i,0), (i,1), \ldots, (i, 2n/k)$ for each $i \in [k]$.
- A complete binary tree, with $n/k+1$ leaves denoted $\ell_0, \ldots, \ell_{n/k}$. Each leaf ℓ_i is connected to each path node $(j, 2i)$ for $j = 0, \ldots, n/k$. The edges $\{\{\ell_i, (j,2i)\}\}_{j \in [k], i \in [n/k+1]}$ are called *bridges*.
- An additional $n/k+1$ nodes denoted $x_0, \ldots, x_{n/k}$, with an edge $\{\ell_i, x_i\}$ connecting x_i to the tree leaf ℓ_i for each $i \in [n/k+1]$. Nodes $x_0, \ldots, x_{n/k}$ are called *spines*.
- For each $i \in x$, Alice appends an edge $e_i^A = \{(i, A), (i, 0)\}$ at the beginning of the path π_i.
- For each $i \in y$, Bob appends an edge $e_i^B = \{(i, B), (i, 2n/k)\}$ at the end of the path π_i.

Let $\pi_i^{x,y}$ be the extended path π_i, after Alice or Bob either add or do not add their edge to their respective endpoints of π_i. The length of each extended path $\pi_i^{x,y}$ is $2n/k+1$ if $x_i \oplus y_i = 1$, and either $2n/k$ or $2n/k+2$ if $x_i \oplus y_i = 0$.

Let us see what Alice and Bob can deduce about $G_{x,y}$ when they see the result of the algorithm, that is, a $(1-\epsilon)$-approximation to the maximum fractional matching.

First we show that we can ignore the bridge edges, by assuming that they receive weight 0.

Lemma 7.1. *Let M be any fractional matching in $G_{x,y}$. There is a matching M', with $|M'| = M$, such that for any path edge $e \in \pi_i^{x,y}$ we have $M'(e) = M(e)$, but for any bridge edge e we have $M'(e) = 0$.*

Proof. We obtain M' from M as follows: let ℓ_i be a tree leaf such that for at least one bridge edge $\{\ell_i, (j,2i)\}$ we have $M(\{\ell_i, (j,2i)\}) > 0$. Since M is feasible, we have $\sum_j M(\{\ell_i, (j,2i)\}) + M(\{\ell_i, x_i\}) \leq 1$. In M', set let $M'(\{\ell_i, (j,2i)\}) = 0$ for all j, and we set $M'(\{\ell_i, x_i\}) = \sum_j M(\{\ell_i, (j,2i)\}) + M(\{\ell_i, x_i\})$. This preserves the total weight of the matching, as well as the total weight edges adjacent to ℓ_i.

We go through all the bridge edges in this manner until we have zeroed out all their weights. The weights of internal tree edges and of path edges in M' is the same as in M. □

Next, we show that any good fractional matching must assign total weight greater than n/k on at least 1/10-th of all odd-length paths, assuming that at least 1/4-th of the paths in $G_{x,y}$ have odd length:

Lemma 7.2. *Let M be a $(1-\epsilon)$-approximation to the maximum matching in $G_{x,y}$, and assume that M assigns no weight to any bridge edges: that is, for all i, j we have $M(\{\ell_i, (j, 2i)\}) = 0$. Assume further that $G_{x,y}$ contains at least $k/4$ odd-length paths $\pi_i^{x,y}$. Then M achieves weight greater than n/k on at least $k/10$ odd-length paths $\pi_i^{x,y}$.*

Proof. Suppose not, and I, $|I| > (1/4 - 1/10)k = (3/20)k$, be the set of odd-length paths on which M achieves weight at most k. We abuse notation by thinking of I as both the set of indices i such that $\pi_i^{x,y}$ has odd length, and also the set of all such path nodes. Similarly, let O be the set of all odd-length paths, $|O| \geq (1/4)k$.

Let M^* be a maximum fractional matching in $G_{x,y}$. By Lemma 7.1, we may assume that M, M^* assign no weight to the bridge edges; therefore, they are both collections of disjoint fractional matchings on the tree and the individual paths.

On any even-length path (which has length either $2n/k$ or $2n/k + 2$), M and M^* can achieve weight at most $n/k + 1$, and on odd-length paths (which have length $2n/k + 1$) the weight can be at most $n/k + 1/2$. Since M^* is optimal, it indeed achieves these weights. Let t be the weight of the optimal fractional matching on the tree and spines. Since they contain a total of fewer than $3n/k$ nodes, we have $t < 3n/k$. Thus, the total weight achieved by both M and M^* on $G_{x,y} \setminus I$ is at most $t + (k - |O|) \cdot (n/k + 1) + (|O| - |I|) \cdot (n/k + 1/2)$. And since M^* is optimal, it actually achieves this weight.

On the paths in I, we assumed that M achieves a total weight of at most $|I| \cdot (n/k)$, while M^* achieve weight $|I| \cdot (k+1/2)$ (these are paths of length $2n/k+1$). Therefore, $|M^*| - |M| \geq |I| \cdot (1/2) \geq (3/40)k$, and $|M^*| \leq t + (k - |O|) \cdot (n/k+1) + |O| \cdot (n/k+1/2) \leq 3k - 1 + (3/4)k \cdot (n/k+1) + (1/4)k \cdot (n/k + 1/2) < 10k + n < 2n$. We see that $(|M^*| - |M|)/|M^*| \geq (3/40)k/(2n) > (1/40)(k/n) > \epsilon$, contradicting our assumption that M is a $(1 - \epsilon)$-approximation to the optimum matching.

□

7.3 The Simulation

Fix a distributed algorithm $\mathcal{A}$ that computes an exact maximum matching in the graph family $\{G_{x,y}\}_{x,y\{0,1\}^n}$ in T rounds, where $T = n/k - 1$. Alice and Bob simulate the execution of $\mathcal{A}$ as follows: at each time t, Alice locally simulates a set S_A^t of nodes, and Bob locally simulates a set S_B^t, with $S_A^0 \supset S_A^1 \supset \ldots \supset S_A^T$ and $S_B^0 \supset S_B^1 \supset \ldots \supset S_B^T$.

Path nodes. For each $0 \leq t \leq T$, let

$$V_A^t = [k] \times \{A, 0, 1, \ldots, 2n/k - t\}, \quad V_B^t = [k] \times \{B, t, t+1, \ldots, 2n/k\}.$$

Alice only simulates path nodes in V_A^t, and Bob simulates path nodes in V_B^t.

Tree nodes. Since the players stop simulating more and more path nodes as the execution progresses, they are able to locally simulate fewer and fewer tree nodes without assistance from the other player.

Let us denote each tree node by the path from the root, denoted ε, to the node, with 0 denoting a left turn and 1 denoting a right turn. Let $H = \lceil \log(n/k+1) \rceil$ be the depth of the tree. In particular, a leaf ℓ_i is denoted by the binary representation of i, with leading zeroes padding the representation to H bits.

At time t, Alice only maintains local states for the tree leafs

$$L_A^{t,0} = \{\ell_i \mid 2i \leq 2n/k - t\}.$$

We define the set of nodes for which Alice stores a local state by induction, with $L_A^{t,h+1}$ containing any inner tree node that has at least one child in $L_A^{t,h}$. Note that $L_A^{t,h}$ is a set of consecutive nodes $\{0^{H-h}, \ldots, u^{t,h}\}$, where $u^{t,h}$ is the largest such that $u^{t,h}0 \leq u^{t,h-1}$. Also, since we remove at most one leaf from $L_A^{t,0}$ to obtain $L_A^{t+1,0}$, at each higher level $h > 0$ we also remove at most one node from $L_A^{t,h}$ to obtain $L_A^{t+1,h}$.

Consider a node $u \in L_A^{t,h}$ for $t > 0, h \geq 0$. If u is not the root, then u's parent is in $L_A^{t,h+1}$ by definition, and therefore the parent is also in $L_A^{t-1,h+1}$. Therefore Alice knows what message the parent sent to u in round $t-1$. As for the children of u (if $h > 0$), if $u \neq u^{t,h}$ (i.e., u is not the last node on level h), then both children are in $L_A^{t,h+1}$, and therefore they are also in $L_A^{t-1,h+1}$ and Alice knows the messages they send; if $u = u^{t,h}$, then still, at least one child of u is in $L_A^{t,h+1}$ and hence in $L_A^{t-1,h+1}$. The other child may not be, and in this case we ask Bob to tell Alice the message this child sends to node u. There is at most one such child on each level, so the total communication cost on all levels together is $O(B \log(n/k))$ per round of the simulation.

The simulation for Bob is symmetric, starting with the leafs

$$L_B^{t,0} = \{\ell_i \mid 2i \geq t\}$$

and proceeding upwards from there in the same way.

Finally, each player simulates any "spine node" x_i where ℓ_i is to be simulated by the player in the next round. This is trivial, because ℓ_i is the only neighbor of x_i.

7.4 The Players' Output

Let $M : E \to [0,1]$ be a $(1-\epsilon)$-approximation to a maximum fractional matching in $G_{x,y}$. For each path $\pi_i^{x,y} = w_1, \ldots, w_s$ in $G_{x,y}$, let $\overline{\pi_i^{x,y}} = w_s, \ldots, w_1$ be the inverted path. Also, given a path $\pi = w_1, \ldots, w_s$, let $\text{odd}(\pi) = \{\{w_{2j+1}, w_{2j+2}\} \mid 2j+2 \leq s\}$ be the set of odd-numbered edges on π.

Now,let $E_A^i = \text{odd}(\pi_i^{x,y})$ be the set of odd-numbered edges on $\pi_i^{x,y}$, from Alice's perspective, and let $E_B^i = \text{odd}(\overline{\pi_i^{x,y}})$ be the set of odd-numbered edges on $\pi_i^{x,y}$, from Bob's perspective (starting from the end of the path and going backwards). Note that if $\pi_i^{x,y}$ has odd length, then $E_A^i = E_B^i$, but if $\pi_i^{x,y}$ has even length, then E_A^i, E_B^i form a partition of the edges of $\pi_i^{x,y}$.

Alice produces the following output a_i in coordinate i: if there is some edge $e \in E_A^i \cap (V_A^T)^2$ with weight $M(e) \leq 1/2$, then Alice outputs 0; otherwise Alice outputs 1. Note that Alice can check whether there is such an edge, because she knows the final states of all the nodes in V_A^T. Bob does the same, substituting E_B^i, V_B^T for E_A^i, V_A^T, respectively.

7.5 Correctness of the Reduction

Lemma 7.3. *If $\pi_i^{x,y}$ is of odd length on which $|M|$ achieves weight greater than n/k, then for any edge $e \in E_A^i = E_B^i$ we have $M(e) > 1/2$. On the other hand, if $\pi_i^{x,y}$ has even length, then there is some edge $e \in \left(E_A^i \cap (V_A^T)^2\right) \cup \left(E_B^i \cap (V_B^T)^2\right)$ with $M(e) \leq 1/2$.*

Proof. Consider the matching M' from Lemma 7.1. It agrees with M on all path edges, so it suffices to show the claim for M'.

First suppose $\pi_i^{x,y}$ has even length. Recall that $V_A^T = [k] \times \{A, 0, 1, \ldots, n/k+1\}$ and $V_B^T = [k] \times \{B, n/k-1, \ldots, 2n/k\}$. Thus, nodes $(i, n/k-1), (i, n/k), (i, n/k+1)$ are simulated by both players until the end. Since M' is feasible, either $M'(\{(i, n/k-1), (i, n/k)\}) \le 1/2$, or $M'(\{(i, n/k), (i, n/k+1)\}) \le 1/2$, or both. Let $e \in \{\{(i, n/k-1), (i, n/k)\}, \{(i, n/k), (i, n/k+1)\}\}$ have $M'(e) \le 1/2$. Note that $e \in (V_A^T)^2 \cap (V_B^T)^2$. Now recall that since $\pi_i^{x,y}$ has even length, E_A^i and E_B^i form a partition of its edges, so either $e \in E_A^i$ or $e \in E_B^i$.

Now suppose $\pi_i^{x,y}$ has odd length, $2n/k+1$, and M' achieves weight greater than n/k on $\pi_i^{x,y}$.

Suppose for the sake of contradiction that there is some odd-numbered edge $e = \{w, w'\} \in E_A^i = E_B^i$ with $M'(e) \le 1/2$. Consider the prefix of $\pi_i^{x,y}$ up to w, inclusive, and the suffix starting from w', inclusive. Let L_1 be the length of the prefix, and L_2 be the length of the suffix. Both are even (as $\pi_i^{x,y}$ has odd length), so M' achieves weight at most $L_1/2$ on the prefix and at most $L_2/2$ on the suffix. Therefore the total weight M' has on $\pi_i^{x,y}$ is $(L_1 + L_2)/2 + M'(e) \le (2nk)/2 + 1/2 = n/k$, a contradiction. □

Lemma 7.4. *If M is a $(1-\epsilon)$-approximation to the maximum fractional matching, then for at least $0.55k$ indices i we have $a_i \wedge b_i = x_i \oplus y_i$.*

Proof. By Lemma 7.3, whenever $\pi_i^{x,y}$ has even length, at least one of the players outputs 0. Thus, whenever $x_i \oplus y_i = 0$, we have $a_i \wedge b_i = 0$ as well. Because we are promised that there are at least $0.49k$ even-length paths, this gives us $0.49k$ indices on which the output is correct.

As for odd-length paths ($x_i \oplus y_i = 1$), Lemmas 7.2, 7.3 together show that on at least $k/10$ paths, at least one of the two players "sees", from their perspective, an odd-numbered edge with weight $\le 1/2$, because there is such an edge in $\left(E_A^i \cap (V_A^T)^2\right) \cup \left(E_B^i \cap (V_B^T)^2\right)$. Therefore at least one player outputs 0, that is, $a_i \wedge b_i = 0$.

Together, we have at least $0.49k + k/10 > 0.55k$ indices solved correctly by the protocol. □

Corollary 7.5. *Given a algorithm for $(1-\epsilon)$-approximate maximum fractional matching in graphs of size $\Theta(n)$, if the running time of the algorithm is $T < 1/(40\epsilon)$, then there is a protocol with communication complexity $O(T \log n)$ for $\mathsf{PXA}_{\Theta(\epsilon n), 0.55}$.*

Proof. Given input x, y to PXA, we scramble the coordinates by applying a random permutation sampled publicly, then construct the graph $G_{x,y}$ and simulate the algorithm. As we showed, if we compute outputs as indicated in Section 7.4, at least a 0.55-fraction of paths are solved correctly. Because we scrambled the coordinates, each coordinate has marginal probability at least 0.55 of being solved correctly. □

7.6 A Lower Bound on PXA

We now show that there is an input distribution μ on $\{0,1\}^2$, such that

$$\underset{\mu^n}{\mathrm{IC}}(\mathsf{PXA}_{n,\delta}) = n \cdot (1 - 5\delta)/40,$$

and in addition, with probability $1 - o(1)$, at least 0.49 of coordinates have $x_i \oplus y_i = 0$, and at least $1/4$ of coordinates have $x_i \oplus y_i = 1$. It suffices to have $\mu(00) + \mu(11) \ge 1/2$ and $\mu(01) + \mu(10) > 1/4$, as Chernoff then shows that the probability of having fewer than 0.49 zero coordinates or fewer than $1/4$ one coordinates is exponentially small.

By the usual direct sum argument, it suffices to show that

$$\underset{\mu,\delta}{\mathrm{IC}}(\mathsf{XA}) = (1-5\delta)/40.$$

Let μ be the uniform distribution on $\{0,1\}^2$.

By definition of the information cost,

$$\begin{aligned}\underset{\mu}{\mathrm{IC}}(\Pi) &= \underset{\mu}{\mathrm{I}}(\mathbf{\Pi};\mathbf{X} \mid \mathbf{Y}) + \underset{\mu}{\mathrm{I}}(\mathbf{\Pi};\mathbf{Y} \mid \mathbf{X}) \\ &\geq \underset{\mu|\mathbf{Y}=1}{\mathrm{I}}(\mathbf{\Pi};\mathbf{X}) \cdot \Pr_{\mu}[\mathbf{Y}=1] + \underset{\mu|\mathbf{X}=1}{\mathrm{I}}(\mathbf{\Pi};\mathbf{Y}) \cdot \Pr_{\mu}[\mathbf{X}=1],\end{aligned}$$

where the last step follows because mutual information is non-negative. Thus,

$$\begin{aligned}\underset{\mu}{\mathrm{IC}}(\Pi) &\geq \frac{1}{2}\left(\underset{\mu|\mathbf{Y}=1}{\mathrm{I}}(\mathbf{\Pi};\mathbf{X}) + \underset{\mu|\mathbf{X}=1}{\mathrm{I}}(\mathbf{\Pi};\mathbf{Y})\right) \\ &= \frac{1}{2}\left(\mathbb{E}_{t\sim\pi|\mathbf{Y}=1}\left[\mathrm{D}\left(\frac{\mu(\mathbf{X}|\mathbf{\Pi}=t,\mathbf{Y}=1)}{\mu(\mathbf{X}|\mathbf{Y}=1)}\right)\right] + \mathbb{E}_{t\sim\pi|\mathbf{X}=1}\left[\mathrm{D}\left(\frac{\mu(\mathbf{Y}|\mathbf{\Pi}=t,\mathbf{X}=1)}{\mu(\mathbf{Y}|\mathbf{X}=1)}\right)\right]\right) \\ &\geq \frac{1}{2}\cdot\frac{1}{2}\left(\mathbb{E}_{t\sim\pi_{11}}\left[\mathrm{D}\left(\frac{\mu(\mathbf{X}|\mathbf{\Pi}=t,\mathbf{Y}=1)}{\mu(\mathbf{X}|\mathbf{Y}=1)}\right)\right] + \mathbb{E}_{t\sim\pi_{11}}\left[\mathrm{D}\left(\frac{\mu(\mathbf{Y}|\mathbf{\Pi}=t,\mathbf{X}=1)}{\mu(\mathbf{Y}|\mathbf{X}=1)}\right)\right]\right), \quad (4)\end{aligned}$$

using the fact that divergence is non-negative. So, for the typical transcript drawn from π_{11}, either Bob "learns $\mathbf{X}$" or Alice "learns $\mathbf{Y}$" from observing the transcript.

Let us figure out what this means. For any transcript t, we can write

$$\pi_{xy}(t) = a^t(x)\cdot b^t(x),$$

where $a^t, b^t : \{0,1\} \to [0,1]$ are functions measuring the player's contributions to the probability that t is generated.

For $\mathbf{X}$, the prior given $\mathbf{Y}=1$ is $\mu(\mathbf{X}|\mathbf{Y}=1) = \mathsf{B}_{1/2}$. The posterior after seeing $\mathbf{\Pi}=t$ is of course also Bernoulli, with

$$\begin{aligned}\mu(\mathbf{X}=1|\mathbf{\Pi}=t,\mathbf{Y}=1) &= \frac{\mu(\mathbf{X}=1|\mathbf{Y}=1)\pi_{11}(t)}{\mu(\mathbf{X}=1|\mathbf{Y}=1)\pi_{11}(t)+\mu(\mathbf{X}=0|\mathbf{Y}=1)\pi_{01}(t)} \\ &= \frac{\frac{1}{2}\pi_{11}(t)}{\frac{1}{2}\pi_{11}(t)+\frac{1}{2}\pi_{01}(t)} \\ &= \frac{a^t(1)b^t(1)}{a^t(1)b^t(1)+a^t(0)b^t(1)} = \frac{1}{1+\frac{a^t(0)}{a^t(1)}}.\end{aligned}$$

Say that t is *good* if

$$\frac{a^t(0)}{a^t(1)} \leq 1/2 \quad \text{or} \quad \frac{b^t(0)}{b^t(1)} \leq 1/2.$$

If $a^t(0)/a^t(1) \leq 1/2$, then

$$\mu(\mathbf{X}=1|\mathbf{\Pi}=t,\mathbf{Y}=1) = \frac{1}{1+\frac{a^t(0)}{a^t(1)}} \geq \frac{1}{1+1/2} = \frac{2}{3}.$$

Plugging this in to the divergence, we get that for any good transcript t,

$$\mathrm{D}\left(\frac{\mu(\mathbf{X}|\mathbf{\Pi}=t,\mathbf{Y}=1)}{\mu(\mathbf{X}|\mathbf{Y}=1)}\right) \geq \mathrm{D}\left(\frac{2/3}{1/2}\right) = \frac{2}{3}\log\left(\frac{2/3}{1/2}\right) + \frac{1}{3}\log\left(\frac{1/3}{1/2}\right) > 1/20.$$

Similarly, if $b^t(0)/b^t(1) \leq 1/2$, then

$$\mathrm{D}\left(\frac{\mu(\mathbf{Y}|\mathbf{\Pi}=t,\mathbf{X}=1)}{\mu(\mathbf{Y}|\mathbf{X}=1)}\right) > 1/20.$$

Therefore, if we can show that the probability of getting a good transcript is sufficiently high, we will get the bound we want from 4: let G be the set of good transcripts; then by (4),

$$\underset{\mu}{\mathrm{IC}}(\Pi) \geq \frac{1}{2}\pi_{11}(G)\cdot(1/20). \tag{5}$$

Let B the set of *bad* transcripts, i.e., those transcripts t that have

$$\frac{a^t(0)}{a^t(1)} > 1/2 \quad \text{and} \quad \frac{b^t(0)}{b^t(1)} > 1/2.$$

For each $z \in \{0,1\}$, let B_1 be the set of bad transcripts on which both players, if their input is 1, output 1. On input 11, the correct behavior is for at least one player to output 0, so

$$\pi_{11}(B_1) \leq \delta.$$

On the other hand, for a bad transcript in $B \setminus B_1$, at least one of the players outputs 0 when their input is 1. Let B_0^A, B_0^B be the set of transcripts on which Alice and Bob, respectively, output 0 when their input is 1. On inputs 01 and 10 the correct behavior is for both players to output 1, and therefore

$$\pi_{01}(B_0^B) \leq \delta \quad \text{and } \pi_{10}(B_0^A) \leq \delta. \tag{6}$$

Observe that if t is a bad transcript, then

$$\pi_{01}(t) = a^t(0)b^t(1) > a^t(1)b^t(1) = \pi_{11}(t)/2, \tag{7}$$

and similarly,

$$\pi_{10}(t) = a^t(1)b^t(0) > a^t(1)b^t(1) = \pi_{11}(t)/2. \tag{8}$$

Combining (6) with (7), (8) yields

$$\pi_{11}(B_0^B) < 2\delta \quad \text{and } \pi_{11}(B_0^A) < 2\delta. \tag{9}$$

All together, we get

$$\pi_{11}(B) \leq \pi_{11}(B_1) + \pi_{11}(B_0^A) + \pi_{11}(B_0^B) < 5\delta. \tag{10}$$

By (5),

$$\underset{\mu}{\mathrm{IC}}(\Pi) \geq \frac{1}{40}\pi_{11}(G) > \frac{1}{40}(1-5\delta).$$

References

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

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

[3] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.

[4] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. The locality of distributed symmetry breaking. In *Proceedings of 53th Symposium on Foundations of Computer Science (FOCS)*, 2012.

[5] R. Cole and U. Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986.

[6] A. Czygrinow and M. Hańćkowiak. Distributed algorithm for better approximation of the maximum matching. In *9th Annual International Computing and Combinatorics Conference (COCOON)*, pages 242–251, 2003.

[7] J. Edmonds. Maximum matching and a polyhedron with $0,1$ vertices. *Canadian Journal of mathematics*, pages 449–467, 1965.

[8] J. Edmonds. Paths, trees, and flowers. *J. of Res. the Nat. Bureau of Standards*, 69 B:125–130, 1965.

[9] F. Eisenbrand, S. Funke, N. Garg, and J. Könemann. A combinatorial algorithm for computing a maximum independent set in a t-perfect graph. In *Proceedings of 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 517–522, 2003.

[10] G. Even, M. Medina, and D. Ron. Distributed maximum matching in bounded degree graphs. In *Proceedings of the 2015 International Conference on Distributed Computing and Networking (ICDCN)*, pages 18:1–18:10, 2015.

[11] M. Fischer. Improved deterministic distributed matching via rounding. In *Proceedings of 31st Symposium on Distributed Computing (DISC)*, pages 17:1–17:15, 2017.

[12] M. Fischer, M. Ghaffari, and F. Kuhn. Deterministic distributed edge-coloring via hypergraph maximal matching. In *Proceedings of 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 180–191, 2017.

[13] M. Ghaffari, D. G. Harris, and F. Kuhn. On derandomizing local distributed algorithms, 2017.

[14] M. Hańćkowiak, M. Karoński, and A. Panconesi. On the distributed complexity of computing maximal matchings. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 219–225, 1998.

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

[16] M. Hańćkowiak, M. Karoński, and A. Panconesi. On the distributed complexity of computing maximal matchings. *SIAM J. Discrete Math.*, 15(1):41–57, 2001.

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

[18] R. M. Karp and J. E. Hopcroft. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 1973.

[19] F. Kuhn and T. Moscibroda. Distributed approximation of capacitated dominating sets. In *Proceedings of 19th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 161–170, 2007.

[20] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. In *Proceedings of 17th Symposium on Discrete Algorithms (SODA)*, pages 980–989, 2006.

[21] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Local computation: Lower and upper bounds. *J. of the ACM*, 63(2), 2016.

[22] Z. Lotker, B. Patt-Shamir, and S. Pettie. Improved distributed approximate matching. In *Proceedings of the 20th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 129–136, 2008.

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

[24] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.

[25] A. McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 170–181, 2005.

[26] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.

[27] S. Plotkin, D. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[28] A. D. Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

[29] M. Wattenhofer and R. Wattenhofer. Distributed weighted matching. In *Proceedings of 18th International Distributed Computing Conference (DISC)*, pages 335–348, 2004.

[30] Y. Yoshida, M. Yamamoto, and H. Ito. An improved constant-time approximation algorithm for maximum matchings. STOC '09, pages 225–234, 2009.