Distributed Approximate Maximum Matching in the CONGEST Model

Mohamad Ahmadi[1]
University of Freiburg, Germany
mahmadi@cs.uni-freiburg.de

Fabian Kuhn[2]
University of Freiburg, Germany
kuhn@cs.uni-freiburg.de

Rotem Oshman
Tel Aviv University, Israel
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.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms, Mathematics of computing → Approximation algorithms

Keywords and phrases distributed graph algorithms, maximum matching, deterministic rounding, communication complexity

Digital Object Identifier 10.4230/LIPIcs.DISC.2018.6

Related Version A full version of the paper is available at [1], `http://tr.informatik.uni-freiburg.de/reports/report286/report00286.pdf`.

[1] Supported by ERC Grant No. 336495 (ACDC).
[2] Supported by ERC Grant No. 336495 (ACDC).

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., [24]) and sublinear-time approximation (e.g., [29]). 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. 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 [17]. 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.** *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., [19, 26]). 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 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 [19]. The algorithm of [19] is general: it approximates general covering and packing LPs. When applied to the weighted fractional matching problem, the algorithm of [19] 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 [19], 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., [18, 19]). 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 $O(\log^2 \Delta \log n)$.[3]

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 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 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.

[3] 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.

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 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.*

The lower bound is based on the framework of [27], 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 [27], here we are not interested in a *verification* problem. In [27], 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., [27] 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, PXA, that we introduce for this purpose. We prove, using information complexity [4], that the randomized communication complexity of PXA 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 PXA 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".

For lack of space, many technical details are omitted in this version of the paper. They appear in the full version of the paper [1].

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 [17], 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 [2, 16, 23]. (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. [5], 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 $\Omega\big(\min\big\{\frac{\log\Delta}{\log\log\Delta}, \sqrt{\frac{\log n}{\log\log n}}\big\}\big)$ [20]. 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 [17], of repeatedly computing a (nearly) maximal vertex-disjoint set of short augmenting paths. The first such algorithm is [21], 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 [3], which gives a randomized algorithm with time complexity $O\big(\text{poly}(1/\varepsilon)\cdot\frac{\log\Delta}{\log\log\Delta}\big)$. Note that the Δ-dependency of the running time matches the lower bound of [20]. 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, the first paper to explicitly study distributed approximation of the weighted maximum matching is [28]. 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 [22] and in [21], which give $O(\log n)$-round randomized CONGEST algorithms with approximation ratios $(1/4-\varepsilon)$ and $(1/2-\varepsilon)$, respectively. In [3], 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 $(1/2-\varepsilon)$-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 [25]. The network is modeled as 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 Integral Maximum Matching in Bipartite Graphs

Here we present an $O(n \log n)$-round deterministic algorithm to compute a maximum integral matching for a given n-node bipartite graph. The algorithm is based on finding *augmenting paths* and using them to increase the size of the matching we are constructing, as in the celebrated Hopcroft–Karp sequential algorithm for matching in bipartite graphs [17]. We give a somewhat informal description of the algorithm here; the full version appears in [1].

Let us review some basic notions. Given a matching M, we say that a node $v \in V$ is *matched* if one of its edges is in the matching, and otherwise we say that v is *free*. A path $u_0, u_1, \ldots, u_k$ is called *alternating* (with respect to M) if u_0 is free, every odd-numbered edge $\{u_{2i}, u_{2i+1}\}$ (where $2i + 1 \leq k$) is in the matching M, and every even-numbered edge $\{u_{2i+1}, u_{2i+2}\}$ (where $2i + 2 \leq k$) is not in the matching. If an alternating path ends in a free node, then it is called an *augmenting* path, and in this case we can increase the size of the matching by removing all the even-numbered edges along the path from the matching, and instead adding all the odd-numbered edges.

Our algorithm is based on the following observation, which forms the basis for the Hopcroft–Karp algorithm:

▶ **Lemma 5** ([17]). *Consider an unweighted 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 5 we get an upper bound on the length of the shortest augmenting path remaining for a matching of given size:

▶ **Corollary 6.** *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 have: $|M| = i = s^* - (s^* - i) = s^*\,(1 - 1/(s^*/(s^* - i))) \leq s^*\,(1 - 1/\lceil s^*/(s^* - i)\rceil)$. Therefore, by Lemma 5, 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. To get an *upper* bound on the length of the shortest augmenting path, we need a *lower* bound on s^*. Thus, we first deterministically compute a 2-approximation $\hat{s} \in [s^*/2, s^*]$, using, e.g., [11] or [19], in $O(\log n)$ rounds. We set $s = 2\hat{s}$. We are guaranteed that $s \geq s^*$, and hence $s/(s - i) \geq s^*/(s^* - i)$, 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 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 6 and the fact that $s \geq s^*$, if $|M| = i$, then indeed there is an augmenting path of length less than $2\lceil s/(s-i)\rceil$. (If $|M| > i$, then we might not find an augmenting path in the current iteration, but this is fine; we move on to the next value of i.)

The time spent constructing the matching is $\sum_{i=1}^{n-1} O\left(s/(s-i)\right) = O(s \log s) = O(s^* \log s^*)$. Now let us explain how we find each augmenting path.

4.1 Setting Up an Augmenting Path: Procedure `SetupPath`

In the `SetupPath` procedure, we are given an upper bound $2k+1$ on the length of the shortest remaining augmenting path, and we want to find and "set up" a collection of vertex-disjoint augmenting paths, in $O(k)$ rounds. Setting up a path means assigning the path a unique *path ID*, informing all path nodes that they are on the path, and having them confirm that they will participate in the path. Once this is done, we augment the matching along all augmenting paths that were successfully set up. (To augment along the path, we only need each path node to know its successor and predecessor on the path.) Note that unlike Hopcroft–Karp, here we do not insist on finding a maximal set of vertex-disjoint augmenting paths; we are satisfied with merely finding and setting up *one* augmenting path, provided there is one with length $\leq 2k+1$.

Finding the path

To find the augmenting path, we perform a $k+1$-round BFS along alternating paths, starting from all free nodes. Initially, each free node sends out a BFS token carrying its ID along all its edges, and tries to have the network propagate this token along alternating paths: in odd rounds the token is sent along edges that are not in the matching, and in even rounds it is sent along matching edges. However, every node in the network forwards *at most one BFS token*, the first one it receives. (If multiple BFS tokens are received in the same round, the one with the smallest ID will be forwarded.) BFS tokens received in subsequent rounds (or in the same round but with a larger ID) are discarded.

During the BFS traversal, each node v stores the BFS token that it propagated, if there is one, in a local variable src_v. Also, node v stores the neighbor from which src_v was received in the local variable $pred_v$. (If src_v was received from multiple neighbors in the same round, then v chooses an arbitrary neighbor.)

An augmenting path is *detected in round* r if in this round, two adjacent nodes u, v both send each other BFS tokens of distinct free nodes $src_u \neq src_v$. This means that the alternating-path BFS started by src_u has "met" the one started by src_v, yielding an augmenting path (i.e., an alternating path that starts and ends at a free node).

We show that if $2\ell+1$ is the length of the shortest augmenting path in the graph, u is the free node with the minimum ID among the nodes that have augmenting paths of length $2\ell+1$, and v is the node with the minimum ID to which u has an augmenting path of length $2\ell+1$, then some augmenting path between u and v is detected in round $\ell+1$. This is because only free nodes start a BFS traversal, and no free node can "block" the BFS started by u unless it has a shorter alternating path to the same node. But this would then imply a shorter augmenting path than the one u has, and we assumed that u has the shortest augmenting path present. Since we know that there exists an alternating path with length at most $2k+1$, we only need to develop the BFS to depth $k+1$ before some augmenting path is detected. Therefore this phase requires $k+1$ rounds.

Setting up the path

When u and v detect an augmenting path in round r, they assign it the path identifier $(r, \{src_u, src_v\}, \{u, v\})$. Path identifiers are sorted in lexicographic order, and we assume some fixed ordering on unordered pairs of IDs. If a node detects more than one augmenting path at the same time, it keeps the one with the smallest path identifier. (This can happen if the augmenting path has length 1, that is, if there are two adjacent free nodes.)

Next, u and v inform all the path nodes of the detected path's ID, by sending messages backwards along the *pred* pointers of the nodes on the path. As we traverse backwards, each node stores its successor on the path in a local variable *succ*. (Note that the *succ* fields point "inwards", towards the edge that detected the path.)

Eventually, each free node receives a (possibly empty) list of augmenting path IDs for which it is an endpoint. (Note that each inner path node can only receive one path ID; only endpoints of the path may receive more than one path ID.) At this point, each free node selects the smallest path ID (in lexicographic order), and discards the others. We know that of the augmenting paths detected, the one with the smallest path ID will be selected by both its endpoints, so at least one augmenting path survives.

We now sweep forward along each selected path, to confirm that it is properly set up: the two endpoints send each other confirmation messages carrying the path ID, by having the path nodes forward the messages along the path. If an inner path node does not receive confirmation from both endpoints, it discards the path, and similarly, if an endpoint of the path does not receive confirmation from the other endpoint, it discards the path.

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 [19]. Further, the distributed algorithm in [19] 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 7.** *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). ◀

Distributed Algorithm for 2-Colored Bipartite Graphs

In light of Lemma 7, we can w.l.o.g. 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.

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 v as $\gamma(v) := \sum_{e \in E : v \in e} \frac{r_e}{w_e}$ and we define $\hat{\gamma}(v) := \max_{u \in \{v\} \cup N(v)} \gamma(u)$.

Our algorithm consists of phases: a node v participates in a phase as long as $\gamma(v) > 0$ and v terminates as soon as $\gamma(v) = 0$. Alg. 1 gives the details of a single phase.

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 incident edge e. 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 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,

Algorithm 1: A single phase of the fractional matching algorithm.

```
for $i \in \{0,1\}$ do
    for all $v \in V_i$ in parallel do
        if $\gamma(v) > 0$ then
            $\theta_v := \hat{\gamma}(v)/\alpha$;
            while $\gamma(v) \geq \theta_v$ do
                $x_v := x_v + 1$;
                for all $e \in E : v \in e$ do
                    $y_e := y_e + \frac{r_e/w_e}{\gamma(v)}$; $r_e := r_e/\alpha^{1/w_e}$;
                    if $r_e \leq \alpha^{-f}$ then $r_e := 0$
```

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. The proof is similar to the analyses in [9, 19] and it appears in the full version of this paper [1].

▶ **Lemma 8.** *At the end of running the above fractional weighted matching algorithm, for all nodes $v \in V$, we have*

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

▶ **Corollary 9.** *Let $\varepsilon \in (0, 1/2]$ be a parameter. By choosing $\alpha = 1 + \varepsilon/c$ and choosing $f = 2c \cdot \ln(\Delta W)/\varepsilon^2$ 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 $G = (V, E)$.*

It remains to bound the time complexity of the algorithm in the distributed setting. The proof appears in the full version of this paper [1]. It shows that a single phase of the algorithm can be implemented in $O(1)$ CONGEST model rounds and that the total number of phases is $O(\log(W\Delta)/\varepsilon^2)$.

▶ **Lemma 10.** *The described fractional weighted matching algorithm can be implemented in $O(f + \log_\alpha(W\Delta))$ rounds in the* CONGEST *model.*

Together with Corollay 9, Lemma 10 directly proves Theorem 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 $i \geq 0$. The relatively straightforward proof of the following lemma is omitted in this short version of the paper.

▶ **Lemma 11.** *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 $i = O\big(\log(\Delta/\varepsilon)\big)$.*

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 $v_1, \dots, v_{s_v}$ and we arbitrarily divide the $d_k(v)$ edges in E_k that are incident to v among the nodes $v_1, \dots, v_{s_v}$ 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 $2^{-(k-1)}$. The proof of the lemma appears in the full version [1].

▶ **Lemma 12.** *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 $\boldsymbol{y}$.

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. This simple change makes the rounding more efficient and it also makes it easier to argue that the rounded matching is not much smaller than the original matching, in particular in the case of weighted matchings. Lemma 12 implies that rounding fractional matchings to integral matchings essentially boils down to computing almost maximum (weighted) matchings in graphs of maximum degree 2.

Approximating Maximum Matching in Paths and Cycles. As discussed above, Lemma 12 essentially reduces the problem of rounding (weighted) fractional matchings to solving the weighted maximum matching problem in paths and cycles.

▶ **Lemma 13.** *Let $G = (V, E)$ be a weighted n-node graph with maximum degree 2 and assume that $\mathcal{W}$ 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.*[4] *In the* CONGEST *model, a matching of weight at least $\frac{g-1}{g} \cdot (1-\delta) \cdot \mathcal{W}/2$ 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 an $O(1/\delta)$-time algorithm that computes a matching of total weight at least $(1-\delta)\mathcal{W}/2$.

Proof Sketch. The full proof of the lemma appears in the full version [1]. The main idea of the proof is as follows. For short paths and cycles, it is straightforward to compute the required matchings. For long paths, we define an edge to be ℓ-light if its weight is at most the average weight in some subpath of length ℓ. We then choose a set L of ℓ-light edges such that after removing those edges, the matchings of the remaining paths can be computed efficiently (either because the paths are sufficiently short or in the case of 2-colored bipartite graphs because we have a 2-edge coloring of the paths that allows to find the matching). If the ℓ-light edges in L are sufficiently separated, one can show that we only lose a $(1 - O(1/\ell))$-factor in the matching size. ◀

Putting the Pieces Together. We now have all the tools that are needed for the rounding and we can therefore prove Theorem 3.

Proof of Theorem 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 7, we now first move to the bipartite double cover of G and by using Lemma 11, 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 12, we therefore have to set $\delta = O(\varepsilon/k) = O(\varepsilon/\log(\Delta/\varepsilon))$. Because B'_k is a 2-colored bipartite graph, Lemma 13 implies that the matching of B'_k which is necessary by Lemma 12 can be computed in time $O(1/\delta) = O(\log(\Delta/\varepsilon)/\varepsilon)$. 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 7 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 13. 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 $(1 - O(\varepsilon)) \cdot \frac{g-1}{g}$. ◀

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

7 Lower Bound

In this section we show that computing a $(1 - O(1/\sqrt{n}))$-approximation to the maximum fractional matching requires $\Omega(\sqrt{n})$ rounds, even in bipartite graphs of diameter $O(\log n)$, and even for randomized algorithms which may err with constant probability.

Our graph construction is based on [27]: it is a collection of $\Theta(\sqrt{n})$ long paths, of length $\Theta(\sqrt{n})$, connected to each other by a tree, which reduces the diameter to $O(\log n)$. The lower bound is by reduction from a 2-party communication complexity problem that we introduce for this purpose. Our lower bound is, very informally speaking, "all about" distinguishing even paths from odd paths; the communication complexity problem reflects this, and it asks the players to distinguish "odd inputs" from "even inputs".

One might wonder why we do not simply reduce from Set Disjointness, as is usually done (e.g., in [27]). The reason is that Set Disjointness is a *decision problem*: given sets $X, Y \subseteq [n]$, the players must decide whether $X \cap Y = \emptyset$. This suffices for [27], because the typical setup there is that the input graph has some *marked edges* in it, and the goal is to decide whether the subgraph induced by the marked edges satisfies some property. In contrast, here we are interested in a *search problem*: unlike [27], we do not have a set of marked edges as part of the input; there is only the network graph, on which the algorithm must approximate the maximum fractional matching.

Another difficulty that we must overcome is that our reduction does not allow both players to compute the same output. Instead, the players may output different bits, and we view their answer is the Boolean AND of their output bits.

The 2-Player Communication 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 $\mathsf{PXA}_{n,\delta}$ ("promise XOR-to-And") 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 $n/3$ copies i we have $x_i \oplus y_i = 1$, and for at least $(2/5)n$ copies i we have $x_i \oplus y_i = 0$. The goal is to solve at least δn of the copies correctly; that is, the players should produce outputs $a_1, b_1, \ldots, a_n, b_n$ such that for at least δn coordinates i we have $a_i \wedge b_i = x_i \oplus y_i$. The players are not charged for writing their outputs, only for the communication between them.

▶ **Theorem 14.** *The randomized communication complexity of $\mathsf{PXA}_{n,\delta}$ is $\Omega((1-\delta)n)$.*

We omit the proof of the communication lower bound here, as it uses standard techniques in information complexity [4]; see the full version [1] for this proof.

The Reduction

Given a CONGEST algorithm $\mathcal{A}$ that computes a $(1 - \Theta(1/\sqrt{n}))$-multiplicative approximation to the maximum fractional matching, we construct a 2-party protocol for $\mathsf{PXA}_{\Theta(\sqrt{n},\delta)}$ (for a small constant δ).

Fix a parameter $k = \Theta(\sqrt{n})$, and assume for simplicity that n/k is an integer. Assume that the algorithm $\mathcal{A}$ runs in time at most $n/k - 1 = O(\sqrt{n})$.

On 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 $i \in [k]$, if $x_i = 1$, Alice prepends an edge $e_i^A = \{(i, A), (i, 0)\}$ at the beginning of the path π_i.
- For each $i \in [k]$, if $y_i = 1$, 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 $x_i \oplus y_i = 0$. We argue that a good approximate matching algorithm must *distinguish* between these two cases on a noticeable fraction of the paths, allowing the players to solve PXA.

The players simulate the execution of $\mathcal{A}$ for $n/k-1$ rounds, until it terminates. The simulation is very similar to the one in [27], with each player initially simulating almost all nodes of the graph, but simulating fewer and fewer nodes as the execution of the algorithm progresses. Specifically, at each time $t \le n/k-1$, let $V_A^t = [k] \times \{A, 0, 1, \ldots, 2n/k - t\}$ and $V_B^t = [k] \times \{B, t, t+1, \ldots, 2n/k\}$ be the path nodes simulated by Alice and Bob, respectively, at time t. At each time t, each player can compute the messages that the nodes in V_A^{t+1} (resp. V_B^{t+1}) will receive in the current round from their neighbors on the path, because it knows the neighbors' local states at time t. In addition to the path nodes, the players also simulate the tree nodes and the spine nodes, fewer and fewer in each round. This part of the simulation is again very similar to [27], and we omit it here.

When $\mathcal{A}$ terminates, *both* players know the states of nodes $(i, n/k-1), (i, n/k), (i, n/k+1)$ for each $i \in [k]$. This overlap is important for our reduction.

Let E_A be the set of edges such that at the end of the simulation, Alice has the local states of both endpoints of the edge, and therefore knows the value the algorithm assigned to this edge. Similarly define E_B for Bob. Let M be the fractional matching computed by the algorithm.

Producing Outputs

After the simulation ends, the players examine the fractional matching produced by the algorithm, and use it to produce outputs, as follows.

For a path $\pi = u_0, \ldots, u_{k-1}$, let $\pi^{-1} = u_{k-1}, \ldots, u_0$ be the inverse path, and let $\mathsf{odd\text{-}edges}(\pi) = \{\{u_{2i}, u_{2i+1}\} \mid 2i+1 < k\}$ be the set of odd-numbered edges along the path (the first edge, the third edge, and so on). Note that if π has odd length (an odd number of edges), then $\mathsf{odd\text{-}edges}(\pi) = \mathsf{odd\text{-}edges}(\pi^{-1})$, but if π has even length, then $\mathsf{odd\text{-}edges}(\pi)$ and $\mathsf{odd\text{-}edges}(\pi^{-1})$ are a *partition* of the edges of π.

For each $i \in [k]$, Alice outputs $a_i = 1$ iff from her perspective, every odd-numbered edge "that she can still see" has value greater than $1/2$. That is, $a_i = 1$ iff for every $e \in \mathsf{odd\text{-}edges}(\pi_i^{x,y}) \cap E_A$ we have $M(e) > 1/2$. Bob does the same, but he views the path in reverse, because he is at the other end of it: he outputs $b_i = 1$ iff for every $e \in \mathsf{odd\text{-}edges}((\pi_i^{x,y})^{-1}) \cap E_B$ we have $M(e) > 1/2$. We argue that this odd-looking decision rule indeed captures the fact that M "looks different" on odd-length and even-length paths.

Correctness of the Deduction

First, note that we can effectively ignore the bridge edges, and assume that they have weight zero: If M assigns non-zero weight to some bridge edge $\{\ell_i, (j, 2i)\}$, we can "shift" this weight onto the corresponding spine edge $\{\ell_i, x_i\}$ and zero out the weight of the bridge edge. The resulting matching M' agrees with M on all the path edges, but since it now induces disconnected components consisting of the k paths and the tree (the bridges have value 0), it must "solve each path individuallly". We can also ignore the tree and spines, because they do not contribute too much to the total value. For simplicity, let us assume here that M is a $(1 - \Theta(1/\sqrt{n}))$-multiplicative approximation to the maximum matching on the paths $\pi_1^{x,y}, \ldots, \pi_k^{x,y}$.

Next, observe that if M is a $(1-\Theta(1/\sqrt{n}))$-multiplicative approximation to the maximum fractional matching, then for a constant fraction of our $k = \Theta(\sqrt{n})$ paths, M must be an $O(1)$-*additive* approximation to the maximum fractional matching on the path: the optimal fractional matching has total value at most $2n$ on all the paths (this is a coarse upper bound), so missing even a constant value $\alpha \in (0, 1)$ on $\beta\sqrt{n}$ paths leads to a multiplicative approximation of only $1 - \alpha\beta/(2\sqrt{n})$.

We set the constants so that for a δ-fraction of the k paths, M is at least a 1/3-additive approximation. Then we show that for any path $\pi_i^{x,y}$, if M achieves a 1/3-additive approximation to the maximum matching on $\pi_i^{x,y}$, then the players correctly solve coordinate i, that is, $a_i \wedge b_i = x_i \oplus y_i$.

The heart of the lower bound is the following simple observation:

▶ **Lemma 15.** *Let π be a path with $2r + 1$ edges, $r \geq 0$, and let M be a 1/3-additive approximation to the maximum fractional matching on π. Then for all $e \in \text{odd-edges}(\pi)$ we have $M(e) > 1/2$.*

Proof. Suppose not, and let $e \in \text{odd-edges}(\pi)$ be an edge with $M(e) \leq 1/2$. Removing edge e from π splits the path into two even-length paths (the suffix and the prefix), with combined length $2r$. On an even path of $2i$ edges, the maximum fractional matching has total value i, and therefore the total value of M on the two even-length paths is at most r. Because $M(e) \leq 1/2$, the total value of M on π is at most $r + 1/2$. But the maximum fractional matching on π has total value $r+1$, so M is not a 1/3-additive approximation. ◀

▶ **Corollary 16.** *If M is a 1/3-additive approximation on $\pi_i^{x,y}$, and $x_i \oplus y_i = 1$ (so $\pi_i^{x,y}$ has odd length), then for every odd-numbered edge $e \in \text{odd-edges}(\pi_i^{x,y})$ we have $M(e) > 1/2$.*

This shows that for odd-length paths that are well-approximated by M, the players do indeed solve XA correctly. What about even-length paths? Here the situation is even simpler:

▶ **Lemma 17.** *If $x_i \oplus y_i = 0$, then there exists an edge*

$$e \in \left(\text{odd-edges}(\pi_i^{x,y}) \cap E_A\right) \cup \left(\text{odd-edges}((\pi_i^{x,y})^{-1}) \cap E_B\right)$$

with $M(e) \leq 1/2$.

Proof. Recall that when $x_i \oplus y_i = 0$, the length of $\pi_i^{x,y}$ is even, and therefore odd-edges$(\pi_i^{x,y})$ and odd-edges$((\pi_i^{x,y})^{-1})$ form a partition of the path edges. Recall also that the overlap of the edges simulated by the players at the end, $E_A \cap E_B$, contains at least two adjacent edges on each path. Because M is feasible, it assigns value $\leq 1/2$ to at least one of these edges, and since the edge is either in odd-edges$(\pi_i^{x,y})$ or in odd-edges$((\pi_i^{x,y})^{-1})$, at least one of the players will output 0. ◀

References

1 M. Ahmadi, F. Kuhn, and R. Oshman. Distributed approximate maximum matching in the congest model. Technical Report 286, U. Freiburg, Dept. of Computer Science, 2018. URL: `http://tr.informatik.uni-freiburg.de/reports/report286/report00286.pdf`.

2 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.

3 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.

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

5 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.

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. `arXiv:1711.02194`.

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

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

18 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.

19 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.

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

21 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.

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

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

24 Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 170–181, 2005.

25 D. Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.

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

27 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

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

29 Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In *STOC '09*, pages 225–234, 2009.