Distributed Approximation of Maximum Independent Set and Maximum Matching[*]

Reuven Bar-Yehuda
Technion, Department of Computer Science
reuven@cs.technion.ac.il

Keren Censor-Hillel
Technion, Department of Computer Science
ckeren@cs.technion.ac.il

Mohsen Ghaffari
ETH Zurich
ghaffari@mit.edu

Gregory Schwartzman
Technion, Department of Computer Science
gregory.schwartzman@gmail.com

ABSTRACT

We present a simple distributed Δ-approximation algorithm for maximum weight independent set (MaxIS) in the CONGEST model which completes in $O(\texttt{MIS}(G) \cdot \log W)$ rounds, where Δ is the maximum degree, $\texttt{MIS}(G)$ is the number of rounds needed to compute a maximal independent set (MIS) on G, and W is the maximum weight of a node. Plugging in the best known algorithm for MIS gives a randomized solution in $O(\log n \log W)$ rounds, where n is the number of nodes. We also present a deterministic $O(\Delta + \log^* n)$-round algorithm based on coloring.

We then show how to use our MaxIS approximation algorithms to compute a 2-approximation for maximum weight matching without incurring any additional round penalty in the CONGEST model. We use a known reduction for simulating algorithms on the line graph while incurring congestion, but we show our algorithm is part of a broad family of *local aggregation algorithms* for which we describe a mechanism that allows the simulation to run in the CONGEST model without an additional overhead.

Next, we show that for maximum weight matching, relaxing the approximation factor to $(2 + \varepsilon)$ allows us to devise a distributed algorithm requiring $O(\frac{\log \Delta}{\log \log \Delta})$ rounds for any constant $\varepsilon > 0$. For the unweighted case, we can even obtain a $(1 + \varepsilon)$-approximation in this number of rounds. These algorithms are the first to achieve the provably optimal round complexity with respect to dependency on Δ.

ACM Reference format:
Reuven Bar-Yehuda, Keren Censor-Hillel, Mohsen Ghaffari, and Gregory Schwartzman. 2017. Distributed Approximation of Maximum Independent Set and Maximum Matching. In *Proceedings of PODC '17, Washington, DC, USA, July 25-27, 2017,* 10 pages.
https://doi.org/http://dx.doi.org/10.1145/3087801.3087806

[*]Supported in part by the Israel Science Foundation (grant 1696/14).

PODC '17, July 25-27, 2017, Washington, DC, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4992-5/17/07...$15.00
https://doi.org/http://dx.doi.org/10.1145/3087801.3087806

1 INTRODUCTION

We address the fundamental problems of approximating the maximum independent set and the maximum matching of a graph in the classic distributed CONGEST model [40]. In this model, the n nodes of the graph communicate in synchronous rounds, by sending one $O(\log n)$-bit message per round along links of the graph. Table 1 summarizes our contributions. Below, we elaborate on our results, the challenges, and how we overcome them.

Problem	Approx.	Prev. Results	Our Results	notes
MaxIS | MWM	Δ | 2	—	$O(\texttt{MIS}(G) \log W)$	rand.
MaxIS | MWM	Δ | 2	—	$O(\Delta + \log^* n)$	det.
MWM	$2 + \varepsilon$	$O(\log n)$	$O(\log \Delta / \log \log \Delta)$	rand.
MCM	$1 + \varepsilon$	$O(\log n)$	$O(\log \Delta / \log \log \Delta)$	rand.

Table 1: Summary of results for the CONGEST model. Here n denotes the number of nodes, Δ is their maximum degree, and W is the maximum weight.

1.1 Our Results, Part I: Better Approximations

Δ-approximation algorithms for maximum weight independent set. We present a simple distributed Δ-approximation algorithm for maximum weight independent set (MaxIS), where Δ is the maximum degree, which completes in $O(\texttt{MIS}(G) \cdot \log W)$ rounds, where $\texttt{MIS}(G)$ is the number of rounds needed to compute a maximal independent set (MIS) on G, and W is the maximum weight of a node. As standard, we assume that W is at most polynomial in n, so that the weight of each edge can be described in one message. Our algorithm adapts the *local ratio* technique [7] for maximization problems [4] to the distributed setting in a novel, yet simple, manner. Roughly speaking, in the simplest form of this technique, one repeatedly picks a node v and reduces its weight from every $u \in N(v)$, where $N(v)$ is the set of neighbors of v. Every neighbor $u \in N(v)$ whose weight becomes less than or equal to zero is removed from the graph, while v is added to a stack. We repeat this process with the induced graph until no nodes remain. We then begin popping nodes from the stack, adding them to the independent set if they have no neighbors in the set. This yields a Δ-approximation.

The challenge in translating this framework to the distributed setting is that if we allow all nodes to perform weight reductions simultaneously, then the above does not hold. For example, consider a star graph where the weight of the center is larger than the weight of any of its neighbors but smaller than their sum. After a single iteration the weights of all the nodes become negative, and no node gets selected. However, we show that if we first compute an independent set and then go on to perform weight reductions we achieve a Δ-approximation factor, while allowing using the power of parallelism. At each iteration we find an MIS, and the nodes chosen to the MIS perform weight reductions. This process is repeated until no nodes with positive weight remain. Nodes are then added to the independent set in reverse order of removal while maintaining the independence constraints. To analyze the running time, our main technique is to group the nodes into $\log W$ *layers* based on their weight. At each iteration, all of the nodes from the topmost layer move to lower layers.

This results in a round complexity of $O(\texttt{MIS}(G) \cdot \log W)$ in the CONGEST model. Our algorithm is deterministic apart from using a black-box algorithm to find an MIS at each iteration. Whether our algorithm is randomized or deterministic depends on the MIS algorithm it uses as a black-box.

We also present a deterministic coloring-based algorithm running in $O(\Delta + \log^* n)$ rounds. Here we first color the graph using $\Delta + 1$ colors, and then use each color group as an independent set to perform weight reductions as in the previous algorithm.

2-approximation algorithms for maximum weighted matching. We use a known reduction to simulate algorithms on the line graph [30], our MaxIS Δ-approximation algorithm gives a 2-approximation for maximum weight matching. Simulating an execution on the line graph in a naive fashion results in a $O(\Delta)$ multiplicative overhead in the CONGEST model. We show our algorithm is part of a broad family of *local aggregation* algorithms for which we describe a mechanism which allows the simulation to run in the CONGEST model without added overhead.

Our deterministic coloring-based algorithm has a favorable running time compared to the algorithm presented in [17] with parameters that result in a 2-approximation. Our randomized algorithm improves upon the $(2 + \epsilon)$-approximation factor of [34]. Using the maximal matching algorithm of [10] on the original graph as an MIS algorithm on the line graph we get a running time of $O((\log \Delta + \log^4 \log n) \cdot \log W)$ [1], with high probability, for the LOCAL model, and using Luby's classical MIS algorithm[37], we get an $O(\log n \cdot \log W)$ algorithm[2] for the CONGEST model. For constant values of W, this is $O(\log n)$ rounds.

1.2 Our Results, Part II: Faster Approximations

Approximations with Optimal Time-Complexity: We provide two approximations algorithms for maximum matching that achieve the optimal round complexity of $O(\log \Delta / \log \log \Delta)$: The first achieves a $(2 + \epsilon)$-approximation of maximum weight matching, and the second a $(1 + \epsilon)$-approximation of maximum cardinality matching, for any constant $\varepsilon > 0$.

These two algorithms improve upon the $O(\log n)$-round algorithms of Lotker et al.[35] for the same problems and same approximation guarantees. Furthermore, these two algorithms are the first constant-approximation algorithms that achieve an optimal round complexity, matching the $\Omega(\log \Delta / \log \log \Delta)$ lower bound of Kuhn et al. [31]. We note that this lower bound holds for any constant approximation, and so long as $\log \Delta \leq \sqrt{\log n}$.

Method Outline. A key ingredient in both of the above fast algorithms is an improvement of the *nearly-maximal independent set* algorithm of Ghaffari [21]. A nearly-maximal independent set is an independent set for which each node in the graph is in the set or has a neighbor in the set with probability at least $1 - \delta$ for a small δ. The main result of [21] is a maximal independent set algorithm with round complexity of $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$. The central building block in that result was finding a nearly-maximal independent set in $O(\log \Delta)$ rounds. Here, we provide an improved nearly-maximal independent set algorithm with a round complexity of $O(\log \Delta / \log \log \Delta)$. This algorithm builds upon the techniques of [21], but with some crucial modifications. The modification is partially inspired by the ideas of the recent vertex-cover approximation algorithm of Bar-Yehuda et al. [6], of balancing two types of progresses. While this improvement does not allow us to improve upon Ghaffari's MIS algorithm, it helps us in obtaining our fast maximum matching approximation algorithms, as we discuss next.

For the $(2 + \epsilon)$-approximation, this improved nearly-maximal independent set algorithm is essentially enough. We run it on the line graph of the network graph, and argue that it gives an $(2 + \epsilon)$-approximation of the maximum unweighted matching. To argue that the algorithm works in the CONGEST model, even when run on the line graph of the network graph, we use the property that this nearly-maximal independent set algorithm is a local aggregation algorithm. Then, we extend this approximation algorithm to the *weighted* case, using techniques of [35, 36].

For the unweighted $(1 + \epsilon)$-approximation, our goal is to use the general framework of Hopcroft and Karp [27], in which we repeatedly search for short non-intersecting augmenting paths and augment the matching with them, hence improving its size. However, in our setting, this does not work as is and poses significant challenges. One key challenge is that, to have the desired approximation factor, we need a much stronger near-maximality guarantee. It does not suffice to have a low probability for each short augmenting path to remain; we need to show that each node has a low probability of having a remaining augmenting path. To overcome the obstacles, first we show how to find a nearly-maximal matching in low-rank hypergraphs and how to modify the algorithm for obtaining the $(1 + \epsilon)$-approximation guarantee in the LOCAL model.

Making the algorithm suitable for the CONGEST model is even more demanding, in part because here we cannot explicitly work with the structure of the intersections between short augmenting paths; instead, we need to have a new variant of the near-maximal independent set algorithm that works *on the fly*. At a high level, we first address bipartite graphs, and show how to find a nearly-maximal independent set of short augmenting paths in them. Since the augmenting paths are not known explicitly, an interesting aspect here will be a variant of the dynamic probability adjustments in the algorithm of[21]. Now, various nodes of a path might decide differently regarding whether to raise or lower its probability. However, we will prove that still the net effect provides a sufficient move in the right direction. We complete by generalizing this from bipartite graphs to all graphs, using an idea of Lotker et al. [35], which essentially

[1]Note that Δ and n are the parameters of the original graph and not the line graph.
[2]Here the MIS algorithm is executed on the line graph, so we get $O(\texttt{MIS}(G)) = O(\log n^2) = O(\log n)$.

transforms the problem into randomly chosen bipartite subgraphs of it.

1.3 Related Work

The maximum independent set problem is known to be NP-hard, as it is complementary to the maximum clique problem, which is one of Karp's 21 NP-hard problems [29]. In the sequential setting, an excellent summary of the known results is given by [3], which we overview in what follows. For general graphs, the best known algorithm achieves a $O(n \log^2 \log n / \log^3 n)$-approximation factor [18]. Assuming $NP \not\subseteq ZPP$, [26] shows that no $(n^{1-\epsilon})$-approximation exists for every constant $\epsilon > 0$.

When the degree is bounded by Δ, a simple $(\Delta+2)/3$-approximation is achieved by greedily adding the node with minimal degree to the independent set and removing its neighbors [24]. The best known approximation factor is $O(\Delta \log\log\Delta / \log\Delta)$ [1, 22, 23, 25, 28]. Conditioned on the Unique Games Conjecture, there exist a $\Omega(\Delta/\log^2\Delta)$-approximation bound [2], where Δ is constant or some mildly increasing function of n. Assuming $P \neq NP$, a bound of $\Omega(\Delta/\log^4\Delta)$ is given in [12].

As for the distributed case, [14, 32] give a lower bound of $\Omega(\log^* n)$ rounds for any deterministic algorithm approximating MaxIS, while [14] provide randomized and deterministic approximations for planar graphs. In [11], an $O(1/\epsilon)$-round LOCAL randomized algorithm for $O(n^\epsilon)$-approximation is presented for the unweighted case, along with a matching lower bound.

Maximum matching is a classical optimization problem, for which the first polynomial time algorithm was given by Edmonds [15, 16] for both the weighted and unweighted case. In the distributed setting, the first algorithm for computing an approximate maximum matching was given in [41], where a 5-approximation factor is achieved w.h.p for general graphs, in $O(\log^2 n)$ rounds. In [36] a randomized $(4+\epsilon)$-approximation for the weighted case is given, running in $O(\log n)$ rounds for constant $\epsilon > 0$. This was later improved in [34] to achieve a $(2+\epsilon)$-approximation in $O(\log \epsilon^{-1} \log n)$ rounds. In [17] a deterministic $(1+\epsilon)$-approximation is given, in $\Delta^{O(1/\epsilon)} + O(1/(\epsilon^2)) \cdot \log^* n$ rounds for the unweighted case, and $\log(\min\{1/w_{min}, n/\epsilon\})^{O(1/\epsilon)} \cdot (\Delta^{O(1/\epsilon)} + \log^* n)$ rounds for the weighted case, where the edge weights are in $[w_{min}, 1]$. In [13] a deterministic $(1+\epsilon)$-approximation is given, which finishes after $O(\log^{D(1/\varepsilon)} n)$ rounds, where $D(1/\varepsilon)$ is some function of $1/\varepsilon$. Due to [31], every algorithm achieving a constant approximation for the problem requires $\Omega(\min\{\log\Delta/\log\log\Delta, \sqrt{\log n/\log\log n}\})$ rounds.

The first distributed algorithm that uses the local ratio technique is due to [39]. The local ratio technique was also used in [6] to compute a distributed $(2+\epsilon)$-approximation for weighted vertex cover. In [38], a similar technique of weight grouping is used in the primal-dual framework for scheduling.

2 MAXIS APPROXIMATION

We begin, in Subsection 2.1, by showing the idea behind the use of local ratio for approximating MaxIS. This is done by presenting a sequential meta-algorithm and analyzing its correctness. Then, in Subsection 2.2, we show how to implement this algorithm in the CONGEST model, and prove the claimed round complexity.

2.1 Sequential MaxIS approximation via local ratio

Here we provide a sequential Δ-approximation meta-algorithm to be used as the base for our distributed algorithm. The correctness of the algorithm is proved using the local ratio technique for maximization problems [4]. We assume a given weighted graph $G = (V, w, E)$, where $w : V \to \mathbb{R}_+$ is an assignment of weights for the nodes and the degree of each node is bounded by Δ. A simple Δ-approximation local ratio algorithm exists for the problem [5]. We rely on the following local ratio theorem for maximization problems [5, Theorem 9] in our proof.

THEOREM 2.1. *Let C be a set of feasibility constraints on vectors in $\mathbb{R}^n$. Let $w, w_1, w_2 \in \mathbb{R}^n$ be vectors such that $w = w_1 + w_2$. Let $x \in \mathbb{R}^n$ be a feasible solution (with respect to C) that is r-approximate with respect to w_1 and with respect to w_2. Then x is r-approximate with respect to w as well.*

In our case the vector w is the weight vector representing the weight function of $G(V, w, E)$, x is a binary vector indicating which nodes are chosen to the solution and the set of constraints C, is the set of independence constraints. We call the graph with weight vector w_1 the *reduced graph* and the graph with weight vector w_2 the *residual graph*.

As standard practice with the local ratio technique, the splitting of the weight vector into w_1, w_2 is done such that any r-approximate solution to the reduced graph can be easily transformed into an r-approximate solution to the residual graph, while keeping it an r-approximate solution for the reduced graph. This allows us to apply weight reductions iteratively, solving each subproblem while maintaining the constraints. It is important to note that the theorem also holds if the weights in the reduced graph take negative values.

For the specific problem of MaxIS, we note that picking some node $v \in V$ and reducing the weight of v from every $u \in N(v)$ splits the weight vector w into two vectors, w_1 and w_2. Where $w_2(v) = w(v)$ for every $u \in N(v)$ and zero for every other node, and $w_1 = w - w_2$. Note that any Δ-approximate solution for the reduced graph can be easily turned into a Δ-approximate solution for the residual graph. This is done by making sure that at least some $u \in N(v)$ is in the solution: If this is not the case, we can always add one $u \in N(v)$ to the solution without violating the independence constraints. This only increases the value of our solution, making it Δ-approximate for both the residual and the reduced graphs.

The above solution is sequential by nature. Implementing it directly in the distributed setting will require $O(n)$ rounds. We notice that if two nodes are in different neighborhoods of the graph then this process can be performed by both of them simultaneously without affecting each other. This observation forms the base for our distributed implementation.

We expand this idea by taking any independent set $U \subseteq V$ and for every $v \in U$ reducing the weight of v from every $u \in N(v)$ in parallel. Next, solve the problem for the reduced graph. If for some $v \in U$, every $u \in N(v)$ is not in the solution for the reduced graph, we add v to the solution for the reduced graph. This yields a Δ-approximate solution for the problem. For the sake of simplicity let $V = [n]$. Let w_2 be the weight vector of the residual graph after performing weight reductions as described above for some independent set $U \subseteq V$. By definition $w_2[v] = \sum_{u \in U \cap N(v)} w[u]$. The weight of the reduced graph is given by $w_1 = w - w_2$. Let $x \in \{0,1\}^n$ be some Δ-approximate solution for the reduced graph. The cost of the solution x is $\sum_v w[v]x[v]$. Let $x' \in \{0,1\}^n$ be defined as follows:

$$x'[u] = \begin{cases} 1 & u \in U \wedge \forall v \in N(u), x[v] = 0 \\ x[u] & \text{otherwise} \end{cases} \quad (1)$$

We prove the following lemma.

LEMMA 2.2. *x' is a Δ-approximate solution for both the reduced graph and the residual graph.*

PROOF. We note that $w_2[u] = w[u]$ for every $u \in U$. Thus, $w_1[u] = 0$ for every $u \in U$. We do not incur any additional cost for the reduced graph because x' is created by adding nodes from U to x. Because x is Δ-approximate for the reduced graph, so is x'.

For the residual graph, only nodes in $\cup_{u \in U} N(u)$ have non zero weights. Let $x^* \in \{0,1\}^n$ be an optimal solution for the residual graph. We can bound from above the weight of x^* by summing over the weights of $N(u)$ for every $u \in U$ where $x^*[u] = 1$, taking into account that for any neighborhood $N(u)$, at most $|N(v)| - 1$ nodes can be selected to a solution due to the independence constraints. We get the following upper bound for the weight x^*:

$$\sum_{v \in V} w_2[v]x^*[v] = \sum_{v \in V} \sum_{u \in U \cap N(v)} w_2[u]x^*[u] = \sum_{u \in U} \sum_{v \in N(u)} w_2[u]x^*[u]$$
$$\leq \sum_{u \in U} w_2[u] \cdot (|N(u)| - 1) = \sum_{u \in U} w_2[u] \cdot deg(u) \leq \Delta \sum_{u \in U} w_2[u].$$

On the other hand, x' is selected such that for each $u \in U$ at least one $v \in N(u)$ is in x' for any $u \in U$. Thus, $x' \cdot w_2 = \sum_{u \in U} \sum_{v \in N(u)} w_2[u] \cdot x'[u] \geq \sum_{u \in U} w_2[u]$, which means that x' is at least a Δ-approximation for x^* on the residual graph, and the proof is complete. □

Overview of Algorithm 1: Using Lemma 2.2 we construct a meta-algorithm that at each iteration picks an independent set $U \subseteq V$, reduces the weights of the elements in U from their neighborhood and calls itself recursively with the reduced weights. This implicitly splits the graph into the reduced graph and the residual graph. A recursive call returns a Δ-approximate solution for the reduced graph which is turned into a Δ-approximate solution for both graphs by adding all nodes in the independent set that do not have neighbors in the returned solution. According to the local ratio theorem the final solution is a Δ-approximation. Currently we are only interested in the correctness of the algorithm, thus it does not matter how the set U is picked. The recursive step of Algorithm 1 returns a Δ-approximate solution for the reduced graph which is then turned into a Δ-approximate solution for the residual graph. Correctness follows from Lemma 2.2 combined with a simple inductive argument. In the next section we implement this algorithm in a distributed setting.

2.2 Distributed MaxIS approximation via local ratio

In this section we implement Algorithm 1 in the distributed setting. We present an algorithm which iteratively finds independent sets and finishes after $\log W$ iterations. This yields a Δ-approximation in $O(\mathtt{MIS}(G) \log W)$ rounds, where $\mathtt{MIS}(G)$ is the running time of a black-box MIS algorithm used. The algorithm that wraps the MIS procedure is deterministic, while the MIS procedure may be random. If the MIS procedure is random and finishes after T rounds w.h.p then our algorithm requires $O(T \log W)$ rounds w.h.p. This holds for the CONGEST model.

Algorithm 1: $\mathtt{SeqLR}(V, E, w)$ - Sequential LR algorithm for maximum independent set

```
1  if $V = \emptyset$ then
2      Return $\emptyset$
3  foreach $v \in V$ do
4      if $w(v) \leq 0$ then
5          $V = V \setminus \{v\}$
6          $E = E \setminus \{(v, u) \mid u \in V\}$
7  Let $U \subseteq V$ be an independent set
8  Let $w_1 = w$
9  foreach $u \in U$ do
10     foreach $v \in N(u)$ do
11         $w_1(v) = w(v) - w(u)$
12 $R = \mathtt{SeqLR}(V, E, w_1)$
13 $U = U \setminus \bigcup_{v \in R} N(v)$
14 Return $R \cup U$
```

From now on we assume that all node weights are integers in $[W]$. The sequential meta algorithm can be implemented distributedly, by having each node in the set perform weight reductions independently of other nodes. The key questions left open in the transition to the distributed setting is how to select our independent set at each iteration and how many rounds we need. Iteratively running the MIS procedure and performing weight reductions does not guarantee anything with regard to the number of nodes removed at each iteration or to the amount of weight reduced.

Overview of the distributed algorithm. The algorithm works by dividing the nodes into *layers* according to their weights. The i-th layer is given by $L_i = \{v \mid 2^{i-1} < w(v) \leq 2^i\}$. During the algorithm each node keeps track of the weights (and layers) of neighboring nodes and updates are sent regarding weight changes and node removals. We divide the algorithm into two stages: the *removal stage* and the *addition stage*.

In the removal stage we find an independent set in the graph and perform weight reductions exactly as in the sequential meta algorithm. When finding the MIS, nodes in higher layers are prioritized over nodes in lower layers. A node cannot start running the MIS algorithm as long as it has a neighbor in a higher level. The most important thing to note here is that nodes in the topmost level never need to wait. A node who is selected to the MIS during the removal stage is a *candidate node*. A node whose weight becomes zero or negative without being added to the MIS is said to be a *removed node*. Removed nodes output $\mathtt{NotInIS}$ and finish, while candidate nodes continue to the addition stage. Both candidate and removed nodes are deleted from the neighborhood of their neighbors.

In the addition stage, a candidate node v remains only with neighbors with higher weights. We say these nodes have *precedence* over the node v. A node v may add itself to the solution only if it has no neighboring nodes which have precedence over it. After a node is added to the solution, all of its neighbors become removed. This corresponds line 13 in the sequential meta algorithm.

Algorithm 2: A distributed Δ-approximation for weighted MaxIS, code for node v

```
//$w(v)$ is the initial weight of $v$, $w_v(v)$ changes during the run of the algorithm
$w_v(v) = w(v)$
$\ell_v(v) = \lceil \log w_v \rceil$
$status = waiting$
while $true$ do
    foreach $reduce(x)$ $received\ from$ $u \in N(v)$ do
        $w_v(v) = w_v(v) - x$
        $N(v) = N(v) \setminus \{u\}$
        if $w_v(v) \leq 0$ then
            Send $removed(v)$ to all neighbors
            return NotInIS
    foreach $removed(u)$ $received\ from$ $N(v)$ do
        $N(v) = N(v) \setminus \{u\}$
    $\ell_v(v) = \lceil \log w_v(v) \rceil$
    Send $weightUpdate(v, w_v(v))$ to all neighbors
    foreach $weightUpdate(u, w')$ $received\ from$ $N(v)$ do
        $w_v(u) = w'$
        $\ell_v(u) = \lceil \log w_v(u) \rceil$
    if $status = waiting$ then
        if $\forall u \in N(v), \ell_v(u) \leq \ell_v(v)$ then
            $status(v) = ready$
            while $\exists u \in N(v), \ell_v(u) = \ell_v(v)$ and
              $status(u) \neq ready$ do
                Continue
            $v$ starts running MIS algorithm
        if $v$ $in\ MIS$ then
            Send $reduce(w_v(v))$ to all neighbors
            $w_v(v) = 0$
            $status = candidate$
        else
            $status = waiting$
    else if $status = candidate$ then
        if $N(v) = \emptyset$ then
            Send $addedToIS(v)$ to all neighbors
            Return InIS
        if $addedToIS(u)$ $received\ from$ $N(v)$ then
            Send $removed(v)$ to all neighbors
            Return NotInIS
```

The correctness of the distributed algorithm follows directly from the correctness of the sequential meta algorithm. We are only left to bound the number of rounds. Let us consider the communication cost of the removal stage. We define the topmost layer to be $L_{top} = L_j$ where $j = arg\,max_i\, L_i \neq \emptyset$. Note that nodes in L_{top} never wait to run the MIS, and that after the MIS finishes for L_{top}, the weight of every $v \in L_{top}$ is reduced by at least a factor of two, emptying that layer. This can repeat at most $\log W$ times.

We assume a black-box MIS algorithm that finishes after $\texttt{MIS}(G)$ rounds with probability at least $1 - p$.

LEMMA 2.3. *With probability at least $1 - p$, $L_{top} = \emptyset$ after $\mathtt{MIS}(G)$ rounds.*

PROOF. Let G' be the graph induced by L_{top}. By the code, nodes in L_{top} need not wait to run an MIS algorithm and as long as a node is not in L_{top} it does not participate in an MIS algorithm. With probability at least $1 - p$ an MIS is selected for G' after $\texttt{MIS}(G)$ rounds. All nodes selected to the MIS have their weights reduced to zero. Every other node v has at least one neighbor in the MIS, whose weight, by our definition of layers, is at least half of the weight of v. Thus the weight of every node $v \in L_{top}$ is halved, emptying the layer. □

We now arrive at the main theorem for this section.

THEOREM 2.4. *The distributed MaxIS approximation algorithm (Algorithm 2) finishes within $O(\mathtt{MIS}(G) \cdot \log W)$ rounds with probability at least $1 - p \log W$ in the* CONGEST *model.* [3]

PROOF. Applying a union bound over all layers, gives that all layers are empty after at most
$\texttt{MIS}(G) \cdot \log W$ iterations with probability at least $1 - p \log W$, by Lemma 2.3. We require $p = o(1/\log W)$. This bounds the communication cost for the removal stage.

Denote by C_i the set of candidate nodes from level L_i. These nodes are at level L_i when they are set to be candidate nodes. Nodes in C_i wait for neighbors with higher precedence to decided whether they enter the solution. We note that nodes in C_0 do not have any neighbors with higher precedence. After nodes in C_0 have decided, the nodes in C_1 do not have to wait and so on. Thus, all candidate nodes make a decision after at most $\log W$ rounds. This bounds the communication cost for the addition stage. □

2.3 Deterministic coloring-based approximation algorithm

In this section we present a simple coloring-based Δ-approximation algorithm for MaxIS. The advantage of this approach is that we have no dependence on W, yielding a deterministic algorithm running in $O(\Delta + \log^* n)$ rounds in the CONGEST model.

In the algorithm (pseudocode in Algorithm 3), instead of partitioning the nodes based on weights, they are partitioned based on colors, where colors with larger index have priority. This will result in a runtime independent of the maximum weight. Nodes perform weight reductions if their color is a local maxima. As in the previous section we have two stages: *removal* and *addition*, and three types of node states: *removed*, *candidate* and *precedent*. After one iteration all nodes of the top color are either candidate or removed nodes. Thus after $\Delta + 1$ iterations all nodes are either candidate or removed nodes. Thus, the removal stage finishes in $O(\Delta)$ rounds.

As in Algorithm 2, after the removal stage all candidate nodes only have nodes who have precedence over them as their neighbors. A node adds itself to the independent set if it has no neighbors with precedence over it, in which case all of its neighbors become

[3]The MIS algorithm is always executed on the entire graph G. Thus, its success probability does not change as we move between levels.

removed. We again note that candidate nodes of the smallest color have no neighbors and are added to the solution. Thus, the removal stages finishes in $O(\Delta)$ rounds.

Algorithm 3: Coloring-based distributed Δ-approximation for weighted MaxIS, code for node v

```
1  Run a $\Delta + 1$ coloring algorithm
2  Let $c : v \longrightarrow [\Delta + 1]$ be a coloring for the nodes
3  $w(v) = w_v$
4  foreach $reduce(w')$ received from $u \in N(v)$ do
5      $w(v) = w(v) - w'$
6      $N(v) = N(v) \setminus \{u\}$
7      if $w(v) \leq 0$ then
8          Send $removed(v)$ to all neighbors
9          return NotInIS
10 foreach $removed(u)$ received from $N(v)$ do
11     $N(v) = N(v) \setminus \{u\}$
12 if $N(v) = \emptyset$ then
13     Send $addedToIS(v)$ to all neighbors
14     Return InIS
15 if $addedToIS(u)$ received from $N(v)$ then
16     Send $removed(v)$ to all neighbors
17     Return NotInIS
18 if $\forall u \in N(v) \setminus \{v\}$ it holds that $c(v) > c(u)$ then
19     foreach $u \in N(v)$ do
20         Send $reduce(w(v))$
21     $w(v) = 0$
```

Algorithm 3 is a distributed implementation of Algorithm 1, where the independent set is selected via its color at each iteration. The correctness of the algorithm follows from the correctness of Algorithm 1. The number of rounds of Algorithm 3 is $O(\Delta + \log^* n)$ by using a deterministic distributed coloring algorithm of $O(\Delta + \log^* n)$ rounds [8, 9].[4] The $\log^* n$ factor cannot be improved upon due to a lower bound by Linial [33].

2.4 Distributed 2-approximation for maximum weighted matching

From the results in the previous section we can now derive local 2-approximation algorithms for maximum matching. Let G be a graph with weighted nodes, and let $L(G)$ be the line graph of G. It is well known that a maximum independent set in $L(G)$ corresponds to a maximum matching in G. An algorithm is executed on the line graph by assigning each edge in G to have its computation simulated by one of its endpoints [30]. We show that running our local ratio based approximation algorithms on $L(G)$ yields a 2-approximate maximum matching in G. The main challenge is how to handle congestion, since nodes in G may need to simulate many edges, thus may have to send many messages in a naive simulation [5].

[4][20] gives a faster coloring but is in LOCAL and in any case we need to pay for the number of colors as well.

[5]What follows is equivalent to iteratively running a maximal matching on weight groups in G and performing local ratio steps on the edges of the matching. We go to $L(G)$ in order to demonstrate how a wide class of algorithms can be executed on the line graph while avoiding congestion.

Recall Algorithm 1, the sequential Δ-approximation meta-algorithm. The approximation factor was proved in Lemma 2.2 to be Δ. Specifically, the following equation provided an upper bound for the weight of an optimal solution x^*. Where U is some MIS in $L(G)$.

$$\sum_{u \in U} \sum_{v \in N(u)} w[u]x^*[u] \leq \sum_{u \in U} w[u] \cdot (|N(u)| - 1)$$
$$= \sum_{u \in U} w[u] \cdot deg(u) \leq \Delta \sum_{u \in U} w[u].$$

The above bound uses the fact that for any node $u \in U$, at most $|N(u)| - 1$ nodes in $N(v)$ can be selected for the solution due to independence constraints. But in $L(G)$ the largest independent set in the neighborhood of some node in $L(G)$ is at most 2, yielding the following upper bound: $\sum_{u \in U} \sum_{v \in N(u)} w[u]x^*[u] \leq \sum_{u \in U} 2w[u]$. We conclude that the algorithms presented in the previous sections provide 2-approximation for maximum matching when executed on $G(L)$.

As for the communication complexity, the line graph has at most $n\Delta$ nodes and degree bounded by $2\Delta - 2$. Thus, simulating our algorithms on $L(G)$ in the LOCAL model does not incur any additional asymptotic cost. However, in a naive implementation in the CONGEST model, we pay an $O(\Delta)$ multiplicative penalty due to congestion. This can be avoided with some modifications to our algorithms, as explained next.

For $e = (v, u)$ that is simulated by v, we call v its *primary* node and u its *secondary* node. We define a family of algorithms called *local aggregation algorithms* and show that these algorithms can be augmented to not incur any additional communication penalty when executed on the line graph relative to their performance on the original graph in the CONGEST model. We begin with some definitions.

Definition 2.5. We say that $f : \Sigma^n \longrightarrow \Sigma$ is *order invariant*, if for any set of inputs $\{x_i\}_{i=1}^n$, and any permutation π, it holds that $f(x_1, ..., x_n) = f(x_{\pi(1)}, ..., x_{\pi(n)})$.

For the sake of simplicity, if f is order invariant we write $f(x_1, ..., x_n)$ as $f(\{x_i\})$. We may also give a partial parameter set to our function, in which case we assume all remaining inputs to the function are the empty character $\epsilon \in \Sigma$. Formally, for $X' = \{x_i\}_{i=1}^k$, denote $f(X') = f(x_1, ..., x_k, \epsilon, ..., \epsilon)$.

Definition 2.6. We say that a function $f : \Sigma^n \longrightarrow \Sigma$ is an *aggregate function* if it is order invariant and there exists a function $\phi : \Sigma^2 \longrightarrow \Sigma$ such that for any set of inputs $X = \{x_i\}^k$, and any disjoint partition of the inputs into X_1, X_2 it holds that $f(X) = \phi(f(X_1), f(X_2))$. The function ϕ is called the *joining function*.

OBSERVATION 2.7. *It is easy to see that Boolean "and" and "or" functions are aggregate functions.*

Let Alg be some algorithm for the CONGEST model. Let $D_{v,i}$ be the local data stored by v during the round i of the algorithm at. Let $D_{N(v),i} = \{D_{u,i} \mid u \in N(v)\}$ be the data of v's immediate neighborhood.

Definition 2.8. We call *Alg* a *local aggregation algorithm* if it only accesses $D_{N(v),i}$ using aggregate functions where $|\Sigma| = O(\log n)$ and $|D_{v,i}| = O(\log n)$ for every $v \in V, i \in [t]$.

We prove the following theorems:

THEOREM 2.9. *If Alg is a local aggregation algorithm running in the* CONGEST *model in $O(t)$ rounds, it can be executed on the line graph in $O(t)$ rounds.*

PROOF. *Alg* is executed on the primary node, and we maintain the invariant that $D_{v,i}$ is always present in both the primary and secondary nodes. Every time *Alg* needs to execute a function f, both the primary and secondary nodes already have the data of all of their neighbors. Each node calculates f on the data of its neighbors, the secondary node sends this calculation to the primary which in turn executes the joining function yielding the desired result. Afterwards the new node data is sent to the secondary node.

No communication is needed to access the data of the neighbors, as a neighbor of e must share a node with it, which contains its data. There is no congestion when sending the value of f or the new data to the secondary node. Thus, the number of rounds is $O(t)$. □

THEOREM 2.10. *Algorithm 2 is a local aggregation algorithm.*

PROOF. Let us explicitly define $D_{v,i}$. Each node knows its weight, status and degree. Formally, $D_{v,i} = \{w_i(v), status_v, deg_i(v)\}$. The algorithm uses "and" and "or" Boolean functions, which by Observation 2.7, are aggregate functions. Each node also needs to update its weight at each iteration. The weight update function can be written as $f_w(w_v, \{w_u \mid u \in N(v)\}) = w_v - \sum w_u$, which is of course an aggregate function. □

This exact same technique can be applied to Algorithm 3, giving the main result for this section:

THEOREM 2.11. *There exist a randomized* 2*-approximation algorithm for maximum weighted matching in the* CONGEST *model running in $O(\mathcal{MIS}(G) \cdot \log W)$ rounds, and a deterministic* 2*-approximation algorithm for maximum weighted matching in the* CONGEST *model running in $O(\Delta + \log^* n)$ rounds.*

3 TIME-OPTIMAL APPROXIMATIONS OF MAXIMUM MATCHING

Here, we provide our $O(\frac{\log \Delta}{\log \log \Delta})$-round algorithms for $(2+\varepsilon)$-approximation of maximum weighted matching and a sketch of a $(1+\varepsilon)$-approximation of maximum unweighted matching. As stated before, these are the first to obtain the provably optimal round complexity, matching the $\Omega(\frac{\log \Delta}{\log \log \Delta})$ lower bound of [31], which holds for *any* constant approximation.

3.1 A fast $(2+\varepsilon)$-approximation of maximum weighted matching

We first present a simple $O(\frac{\log \Delta}{\log \log \Delta})$-round $(2+\varepsilon)$-approximation for maximum unweighted matching. We then explain how this approximation extends to the weighted setting via known methods.

To get a $(2+\varepsilon)$-approximation, we gradually find large matchings and remove them from along with the other edges that are incident on them. At the end, we show that the remaining graph has only a small matching left, hence allowing us to prove an approximation guarantee.

The key algorithmic component in our approach is an adaptation of the algorithm of Ghaffari [21]. Ghaffari presented an MIS algorithm, which if executed on a graph H with maximum degree Δ for $O(\log \Delta + \log 1/\delta)$ rounds, computes an independent set IS of nodes of H, with the following probabilistic near-maximality guarantee: each node of H is either in IS or has a neighbor in it, with probability at least $1-\delta$. We will be applying a similar method on the line graph of our original graph, hence choosing a nearly-maximal independent set of edges. However, this running time is not quite fast enough for our target complexity.

We first explain relatively simple changes in the algorithm and its analysis that improve the complexity to $O(\frac{\log \Delta}{\log \log \Delta})$, for any constant $\varepsilon > 0$. We then explain how that leads to an $O(\frac{\log \Delta}{\log \log \Delta})$-round $(2+\varepsilon)$-approximation for maximum unweighted matching.

The Modified Nearly-Maximal Independent Set Algorithm. In each iteration t, each node v has a probability $p_t(v)$ for trying to join the independent set IS. Initially $p_0(v) = 1/K$, for a parameter K to be fixed later. The total sum of the probabilities of neighbors of v is called its *effective-degree* $d_t(v)$, i.e., $d_t(v) = \sum_{u \in N(v)} p_t(u)$. The probabilities change over time as follows:

$$p_{t+1}(v) = \begin{cases} p_t(v)/K, & \text{if } d_t(v) \geq 2 \\ \min\{Kp_t(v), 1/K\}, & \text{if } d_t(v) < 2. \end{cases}$$

The probabilities are used as follows: In each iteration, node v gets *marked* with probability $p_t(v)$ and if no neighbor of v is marked, v joins IS and gets removed along with its neighbors.

THEOREM 3.1. *For each node v, the probability that by the end of round $\beta(\log \Delta / \log K + K^2 \log 1/\delta) = O(\frac{\log \Delta}{\log \log \Delta})$, for a large enough constant β, node v is not in* IS *and does not have a neighbor in* IS *is at most δ. Furthermore, this holds even if coin tosses outside $N_2^+(v)$ are determined adversarially.*

Let us say that a node u is *low-degree* if $d_t(u) < 2$, and *high-degree* otherwise. We define two types of *golden rounds* for a node v: (1) rounds in which $d_t(v) < 2$ and $p_t(v) = 1/K$, (2) rounds in which $d_v(t) \geq 1$ and at least $d_t(v)/(2K^2)$ of $d_t(v)$ is contributed by low-degree neighbors.

LEMMA 3.2. *By the end of round $\beta(\log \Delta / \log K + K^2 \log 1/\delta)$, either v has joined* IS*, or has a neighbor in* IS*, or at least one of its golden round counts reached $\frac{\beta}{13}(\log \Delta / \log K + K^2 \log 1/\delta)$.*

PROOF. Let $T = \beta(\log \Delta / \log K + K^2 \log 1/\delta)$ for a sufficiently large constant β. We focus only on the first T rounds. Let g_1 and g_2 respectively be the number of golden rounds of types 1 and 2 for v, during this period. We assume that by the end of round T, node v is not removed and $g_1 \leq T/13$, and we conclude that $g_2 \geq T/13$.

Let h be the number of rounds during which $d_t(v) \geq 2$. Notice that the changes in $p_t(v)$ are governed by the condition $d_t(v) \geq 2$ and the rounds with $d_t(v) \geq 2$ are exactly the ones in which $p_t(v)$ decreases by a K factor. Since the number of K-factor increases in

$p_t(v)$ can be at most equal to its number of K-factor decreases, there are at least $T-2h$ rounds in which $p_t(v)=1/K$. Out of these rounds, at most h rounds can have $d_t(v)\geq 2$. Hence, $g_1 \geq T-3h$. The assumption $g_1 \leq T/13$ gives that $h \geq 4T/13$. Let us now consider the changes in the effective-degree $d_t(v)$ of v over time. If $d_t(v)\geq 1$ and this is not a golden round of type-2, then we have

$$d_{t+1}(v) \leq K\frac{1}{2K^2}d_t(v) + \frac{1}{K}(1-\frac{1}{2K^2})d_t(v) < \frac{3}{2K}d_t(v).$$

There are g_2 golden rounds of type-2. Except for these, whenever $d_t(v)\geq 1$, the effective-degree $d_t(v)$ shrinks by at least a $\frac{3}{2K}$ factor. In these exception cases, it increases by at most a K factor. Each of these exception rounds cancels the effect of no more than 2 shrinkage rounds, as $(\frac{3}{2K})^2\cdot K \ll 1$. Thus, ignoring the total of at most $3g_2$ rounds lost due to type-2 golden rounds and their cancellation effects, every other round with $d_t(v)\geq 2$ pushes the effective-degree down by a $\frac{3}{2K}$ factor. This cannot happen more than $\log_{\frac{2K}{3}}\Delta$ times as that would lead the effective degree to exit the $d_t(v)\geq 2$ region. Hence, the number of rounds in which $d_t(v)\geq 2$ is at most $\frac{\log\Delta}{\log\frac{2K}{3}}+3g_2$. That is, $h \leq \frac{\log\Delta}{\log\frac{2K}{3}}+3g_2$. Since $h\geq 4T/13$, and because $T=\beta(\log\Delta/\log K + K^2\log 1/\delta)$ for a sufficiently large constant β, we get that $g_2 > T/13$. □

LEMMA 3.3. *In each type-1 golden round, with probability at least $\Theta(1/K)$, v joins the IS. Moreover, in each type-2 golden round, with probability at least $\Theta(1/K^2)$, a neighbor of v joins the IS. Hence, the probability that by the end of round $\beta(\log\Delta/\log K+K^2\log 1/\delta)$, node v has not joined the IS and does not have a neighbor in it is at most δ. These statements hold even if the coin tosses outside $N_2^+(v)$ are determined adversarially.*

PROOF. In each golden type-1 round, we have $d_t(v)<2$ and $p_t(v)=1/K$. The latter means that node v gets marked with probability $1/K$, and the former means that the probability that none of the neighbors of v is marked is at least $\prod_{u\in N_t(v)}(1-p_t(u)) \geq 4^{-\sum_{u\in N_t(v)}p_t(u)} = 4^{-d_t(v)} \geq 1/16$. Hence, in each golden type-1 round, node v joins the IS with probability at least $1/(16K)$.

In each golden type-2 rounds, we have $d_v(t)\geq 1$ and at least $d_t(v)/(2K^2)$ of $d_t(v)$ is contributed by low-degree neighbors. Suppose we examine the set $L_t(v)$ of low-degree neighbors of v one by one and check whether they are marked or not. We stop when we reach the first marked node. The probability that we find at least one marked node is at least $1-\prod_{u\in L_t(v)}(1-p_t(u)) \geq 1-e^{-\sum_{u\in L_t(v)}p_t(u)} \geq 1-e^{-d_t(v)/(2K^2)} \geq 1-e^{-1/2K^2} \geq 1/4K^2$, given that $(2K^2)\geq 1$. Now that we have found the first marked light neighbor u, the probability that no neighbor w of u is marked is at least $\prod_{w\in N_t(u)}(1-p_t(w)) \geq 4^{-\sum_{w\in N_t(u)}p_t(w)} = 4^{-d_t(u)} \geq 1/16$. Therefore, overall, the probability that node v gets removed in a type-2 golden round is at least $\frac{1}{64K^2}$.

Now notice that these events are independent in different rounds. Hence, the probability that node v does not get removed after $\Theta(K^2\log 1/\delta)$ golden rounds is at most $(1-\frac{1}{64K^2})^{\Theta(K^2\log 1/\delta)} \leq \delta$. Furthermore, in the above arguments, we only relied on the randomness in the nodes that are at most within 2 hops of v. Hence, the guarantee is independent of the randomness outside the 2-neighborhood of v. □

PROOF OF 3.1. By 3.2, within the first $\beta(\log\Delta/\log K+K^2\log 1/\delta)$ round, each node v is either already removed (by joining or having a neighbor in the IS) or one of its golden round counts reaches at least $\beta(\log\Delta/\log K+K^2\log 1/\delta)/13$. As 3.3 shows, in each golden round, node v gets removed with probability at least $\Theta(1/K^2)$. Hence, given a large enough constant β, the probability that node v remains through $\beta(K^2\log 1/\delta)/13$ golden rounds is at most δ. □

THEOREM 3.4. *There is a distributed algorithm in the* CONGEST *model that computes a $(2+\varepsilon)$-approximation of maximum unweighted matching in $O(\frac{\log\Delta}{\log\log\Delta})$ rounds, for any constant $\varepsilon>0$, whp.*

PROOF. The algorithm executes the nearly-maximal independent set algorithm explained above on the line-graph. This finds a nearly-maximal set of independent edges, i.e., edges which do not share an endpoint, or in other words, a nearly-maximal matching. The fact that the algorithm can be run on the line-graph in the CONGEST model follows from 2.4, since it is easy to see that this is a local aggregation algorithm. The round complexity of $O(\frac{\log\Delta}{\log\log\Delta})$ follows from the $O(\log\Delta/\log K+K^2\log 1/\delta)$ bound of 3.1, by setting $K=\Theta(\log^{0.1}\Delta)$ and $\delta=2^{-\log^{0.7}\Delta}$. Let us now examine the approximation factor. Each edge of the optimal matching has probability at most δ of becoming unlucky and not being in our found matching and not having any adjacent edge in it either. These are the edges that remain after all iterations of the nearly-maximal independent set algorithm. Thus, we expect at most $\delta \ll \varepsilon$ fraction of the edges of the optimal matching to become unlucky. The number also has an exponential concentration around this mean[6]. Ignoring these $\varepsilon|OPT|$ unlucky edges of the optimal matching, among the rest of the edges, each edge of the found matching can be blamed for removing at most 2 edges of the optimal matching. So the found matching is a $(2+\varepsilon)$-approximation. □

Extension to the Weighted Case via Methods of Lotker et al. Above, we explain an $O(\log\Delta/\log\log\Delta)$-round algorithm for $(2+\varepsilon)$-approximation of maximum unweighted matching. This can be extended to the weighted case via known methods, while preserving the asymptotic complexity, as follows: First, we sketch a method of Lotker et al.[36] which allows one to turn a $(2+\varepsilon)$-approximation for the unweighted case to an $O(1)$-approximation for the weighted case. Classify the weights of edges into powers of a large constant β, i.e., by defining weight buckets of the form $[\beta^i,\beta^{i+1}]$. In each of these big-buckets, partition the weight range further into $O(\log_{1+\varepsilon}\beta)$ *small-buckets* in powers of $1+\varepsilon$. Run the following procedure in all big-buckets in parallel: Starting from the edges of the highest weight small-bucket in this big-bucket, find a $(2+\varepsilon)$-approximation of the matching in that small-bucket using the unweighted matching algorithm, remove all their incident edges in that big-bucket, and move to the next biggest small-bucket. After $O(\log_{1+\varepsilon}\beta)$ iterations of going through all the small-buckets, for each big-bucket, we have found a matching that is a $2+O(\varepsilon)$ approximation of the maximum weight matching among all the edges with weight in this big-bucket. However, altogether, this is not a matching as a node might have

[6]This concentration is due to the fact that the dependencies are local and each edge's event of being unlucky depends on only at most Δ other edges. However, one can obtain a better success probability. See **??** for an algorithm which provides a stronger concentration, giving a $(2+\varepsilon)$-approximation with probability $1-e^{-\Omega(-|OPT|)}$.

a "matching"-edge incident on it in each of the big-buckets. Keep each of these chosen edges only if it has the highest weight among the chosen edges incident on it. Lotker et al.[36] showed that this produces an $O(1)$-approximation of the maximum weight matching.

This $O(1)$-approximation can be turned into a $(2+\varepsilon)$-approximation. Lotker et al. [35, Section 4] present a method that via $O(1/\varepsilon)$ black-box usages of an $O(1)$-approximation Maximum Weight Matching algorithm $\mathcal{A}$, produces a $(2+\varepsilon)$-approximation of the maximum weight matching. We here provide only a brief and intuitive sketch. The method is iterative, each iteration is as follows. Let M be the current matching. We look only at weighted augmenting paths of M with length at most 3. We define an auxiliary weight for each unmatched edge e, which is equal to the overall weight-gain that would be obtained by adding e to the matching and instead erasing the matching M edges incident on endpoints of e (if there are any). Note that this auxiliary weight can be computed easily in $O(1)$ rounds. Then, we use algorithm $\mathcal{A}$ to find a matching which has an auxiliary weight at most an $O(1)$ factor smaller than the maximum weight matching, according to the auxiliary weights. Then we augment M with all these found matching edges, erasing the previously matching edges incident on their endpoints. We are then ready for the next iteration. As Lotker et al. show, after $O(1/\varepsilon)$ iterations, the matching at hand is a $(2+\varepsilon)$-approximation of the maximum weight matching.

3.2 A fast $(1+\varepsilon)$-approximation of maximum cardinality matching

We now discuss our $O(\log\Delta/\log\log\Delta)$-round algorithm for $(1+\varepsilon)$-approximation of maximum unweighted matching. Since the algorithm and its analysis are somewhat lengthy and technical, we can present only a bird's-eye view of them. We first discuss the algorithm for the LOCAL model, and then briefly discuss some of the ideas we use for extending it to the CONGEST model.

We follow a classical approach of Hopcroft and Karp[27]. Given a matching M, an augmenting path P with respect to M is a path that starts with an unmatched vertex, and alternates between non-matching and matching edges, and ends in an unmatched vertex. Flipping an augmenting path P means removing $P \cap M$ edges from M and replacing them with edges of $P \setminus M$. The approximation algorithm based on the method of Hopcroft and Karp[27] works as follows: For each $\ell = 1$ to $O(1/\varepsilon)$, find a *maximal set of vertex-disjoint augmenting paths of length* ℓ, and flip all of them.
The analysis of [27] shows that this finds a $(1+\varepsilon)$-approximation of maximum matching. Hence, all that we need to do is to find a maximal set of vertex-disjoint augmenting paths of length ℓ. This problem can be formulated as a maximal independent set problem in a virtual graph, called the *conflict graph*: put one vertex for each augmenting path of length ℓ, and connect two vertices if their corresponding paths intersect. Each communication round on the conflict graph can be simulated in $O(\ell) = O(1)$ rounds in the network in the LOCAL model. Thus, if we could find an MIS in $O(\log\Delta/\log\log\Delta)$ rounds, we would be done. However, we only know how to compute a nearly-maximal independent set in this number of rounds.

When applied in this context, the guarantee that our nearly-maximal independent set algorithm provides is that each augmenting path of length ℓ has only a small δ probability of remaining (without any intersecting ℓ-length augmenting path in the computed nearly-maximal set). However, this notion of near-maximality is not strong enough for us to be able to say that we still get a good approximation. For instance, one natural idea would be to simply discard the remaining augmenting paths (and thus also their vertices). However, this is not possible because with the current notion of near-maximality, each node may have a high probability of having at least one of the augmenting paths going through it remain. Notice that there are up to Δ^{ℓ} such paths and we cannot afford to use a union bound over them.

In a nutshell, our approach is to provide a much tighter analysis of (a simple variation of) the algorithm, by leveraging the fact that the paths are short, having length $O(1/\varepsilon)$. This tighter analysis allows us to say that if we run the algorithm for slightly more time, larger by an $O(1/\varepsilon^2)$ factor, then the probability of each node having a remaining augmenting path will be small enough to allow us to discard such nodes. This tighter analysis is presented in a more general framework, which may be of independent interest. It concerns computing a nearly-maximal matching in a low-rank hypergraph, where each hyperedge contains a small number of vertices. The relation is that we can think of each augmenting path as one hyperedge, on the same set of vertices. These hyperedges would have rank $O(1/\varepsilon)$, and a matching of hyperedges — that is, a set of hyperedges that do not share a vertex — would be the a set of vertex-disjoint paths.

The above sketches our algorithm in the LOCAL model. Making this algorithm work in the CONGEST model brings in a range of new challenges. For instance, we cannot build the conflict graph explicitly and hence, the above algorithm does not work as is. We use a number of ideas, in order to extend the algorithm to the CONGEST model, which are described in the full version.

One of the key ideas, which we find particularly interesting and may prove useful beyond our work, is a decentralized manner of performing the increase or decreases of the marking probabilities in the nearly-maximal independent set algorithm (as discussed in the previous subsection). Notice that now each augmenting path has a probability, which we would like to increase or decrease. Roughly speaking, we define an attenuation parameter $\alpha_t(v)$ for each node v and we let the marking probability of each augmenting path be the multiplication of the attenuations of its vertices. Each node will decide on its own whether to increase or decrease its attentuation. Of course it is possible that some nodes of the path raise their attenuation and some lower it. However, we prove that in a long enough span of time and by choosing the increase or decrease parameters right, the net effect will still be in the correct direction, allowing us to mimic the analysis of the ideal LOCAL model algorithm, and thus prove the approximation guarantee.

4 DISCUSSION

This papers gives distributed approximation algorithms for maximum independent set and maximum matching in CONGEST. We obtain a Δ-approximation for the former using local-ratio techniques, and deduce a 2-approximation for the latter by defining local aggregation algorithms and showing that they allow simulation on the line graph in the restricted CONGEST model, despite the need to simulate many nodes.

We then provide fast approximations that relax either the approximation factor to $2 + \varepsilon$ or the setting to an unweighted case, but have the optimal round complexity of $O(\log \Delta / \log \log \Delta)$.

One intriguing open question is whether this fast running time can also be obtained for the problem of finding a maximal independent set. Notice that by the algorithm we describe in 3.1, we can compute an almost maximal independent set in $O(\log \Delta / \log \log \Delta)$ rounds. Particularly, this is an independent set where the probability of each node remaining (without being, or having a neighbor, in the independent set) is at most $2^{-\log^{1-\gamma} \Delta}$, for any desirably small constant $\gamma > 0$. However, to be able to extend this to a maximal independent set, we would need this failure probability to be at most $2^{-\Theta(\log \Delta)}$. Furthermore, given our algorithm and the lower bound of Kuhn et al. [31], we now know that this is the complexity of finding a constant approximation for maximum matching. Similarly, by Bar-Yehuda et al. [6], we also know that this is the complexity of finding a constant approximation for the vertex-cover problem. However, for the problem of finding a maximal independent set, there remains a $\log \log \Delta$ gap between the lower bound of [31] and the algorithm of Ghaffari [21].

REFERENCES

[1] Noga Alon and Nabil Kahale. 1998. Approximating the independence number via the ϑ-function. *Mathematical Programming* 80, 3 (1998), 253–264.
[2] Per Austrin, Subhash Khot, and Muli Safra. 2009. Inapproximability of vertex cover and independent set in bounded degree graphs. In *Computational Complexity, 2009. CCC'09. 24th Annual IEEE Conference on*. IEEE, 74–80.
[3] Nikhil Bansal. 2015. Approximating independent sets in sparse graphs. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 1–8.
[4] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. 2001. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)* 48, 5 (2001), 1069–1090.
[5] Reuven Bar-Yehuda, Keren Bendel, Ari Freund, and Dror Rawitz. 2004. Local ratio: A unified framework for approximation algorithms. in memoriam: Shimon even 1935-2004. *ACM Computing Surveys (CSUR)* 36, 4 (2004), 422–463.
[6] Reuven Bar-Yehuda, Keren Censor-Hillel, and Gregory Schwartzman. To Appear 2016. A Distributed $(2 + \epsilon)$-Approximation for Vertex Cover in $O(\log \Delta / \epsilon \log \log \Delta)$ Rounds. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*. https://doi.org/10.1145/2933057.2933086
[7] Reuven Bar-Yehuda and Shimon Even. 1985. A local-ratio theorem for approximating the weighted vertex cover problem. *North-Holland Mathematics Studies* 109 (1985), 27–45.
[8] Leonid Barenboim. 2015. Deterministic $(\Delta + 1)$-Coloring in Sublinear (in Δ) Time in Static, Dynamic and Faulty Networks. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, Chryssis Georgiou and Paul G. Spirakis (Eds.). ACM, 345–354. https://doi.org/10.1145/2767386.2767410
[9] Leonid Barenboim, Michael Elkin, and Fabian Kuhn. 2014. Distributed (Delta+1)-Coloring in Linear (in Delta) Time. *SIAM J. Comput.* 43, 1 (2014), 72–95. https://doi.org/10.1137/12088848X
[10] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. 2016. The Locality of Distributed Symmetry Breaking. *J. ACM* 63, 3 (2016), 20:1–20:45. https://doi.org/10.1145/2903137
[11] Marijke HL Bodlaender, Magnús M Halldórsson, Christian Konrad, and Fabian Kuhn. To Appear 2016. Brief Announcement: Local Independent Set Approximation. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*.
[12] Siu On Chan. 2013. Approximation resistance from pairwise independent subgroups. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 447–456.
[13] Andrzej Czygrinow and Michal Hanckowiak. 2003. Distributed Algorithm for Better Approximation of the Maximum Matching. In *Computing and Combinatorics, 9th Annual International Conference, COCOON 2003, Big Sky, MT, USA, July 25-28, 2003, Proceedings*. 242–251. https://doi.org/10.1007/3-540-45071-8_26
[14] Andrzej Czygrinow, Michal Hańćkowiak, and Wojciech Wawrzyniak. 2008. Fast distributed approximations in planar graphs. In *Distributed Computing*. Springer, 78–92.
[15] Jack Edmonds. 1965. Maximum matching and a polyhedron with 0, l-vertices. *J. Res. Nat. Bur. Standards B* 69, 1965 (1965), 125–130.
[16] Jack Edmonds. 1965. Paths, trees, and flowers. *Canadian Journal of mathematics* 17, 3 (1965), 449–467.
[17] Guy Even, Moti Medina, and Dana Ron. 2015. Distributed maximum matching in bounded degree graphs. In *Proceedings of the 2015 International Conference on Distributed Computing and Networking*. ACM, 18.
[18] Uriel Feige. 2004. Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics* 18, 2 (2004), 219–225.
[19] Ted Fischer, Andrew V Goldberg, David J Haglin, and Serge Plotkin. 1993. Approximating matchings in parallel. *Inform. Process. Lett.* 46, 3 (1993), 115–118.
[20] Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. 2015. Local Conflict Coloring. *arXiv preprint arXiv:1511.01287* (2015).
[21] Mohsen Ghaffari. 2016. An Improved Distributed Algorithm for Maximal Independent Set. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '16)*. SIAM, 270–277. http://dl.acm.org/citation.cfm?id=2884435.2884455
[22] Magnús M Halldórsson. 1998. Approximations of independent sets in graphs. In *Approximation algorithms for combinatiorial optimization*. Springer, 1–13.
[23] Magnús M Halldórsson. 2000. Approximations of weighted independent set and hereditary subset problems. *Journal of Graph Algorithms and Applications* 4, 1 (2000), 1–16.
[24] Magnús M Halldórsson and Jaikumar Radhakrishnan. 1997. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica* 18, 1 (1997), 145–163.
[25] Eran Halperin. 2002. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM J. Comput.* 31, 5 (2002), 1608–1623.
[26] Johan Håstad. 1996. Clique is hard to approximate within $n^{1-\epsilon}$. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*. IEEE, 627–636.
[27] John E Hopcroft and Richard M Karp. 1973. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing* 2, 4 (1973), 225–231.
[28] David Karger, Rajeev Motwani, and Madhu Sudan. 1998. Approximate graph coloring by semidefinite programming. *Journal of the ACM (JACM)* 45, 2 (1998), 246–265.
[29] Richard M. Karp. 1972. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York*. 85–103. http://www.cs.berkeley.edu/~luca/cs172/karp.pdf
[30] Fabian Kuhn. 2005. *The price of locality*. Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 16213, 2005.
[31] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 2006. The price of being near-sighted. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. Society for Industrial and Applied Mathematics, 980–989.
[32] Christoph Lenzen and Roger Wattenhofer. 2008. Leveraging Linial's locality limit. In *Distributed Computing*. Springer, 394–407.
[33] Nathan Linial. 1987. Distributive Graph Algorithms Global Solutions from Local Data. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (SFCS '87)*. IEEE Computer Society, Washington, DC, USA, 331–335. https://doi.org/10.1109/SFCS.1987.20
[34] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. 2008. Improved distributed approximate matching. In *Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*. ACM, 129–136.
[35] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. 2015. Improved distributed approximate matching. *Journal of the ACM (JACM)* 62, 5 (2015).
[36] Zvi Lotker, Boaz Patt-Shamir, and Adi Rosén. 2009. Distributed approximate matching. *SIAM J. Comput.* 39, 2 (2009), 445–460.
[37] Michael Luby. 1986. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing* 15, 4 (1986), 1036–1053.
[38] Alessandro Panconesi and Mauro Sozio. 2008. Fast distributed scheduling via primal-dual. In *Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*. ACM, 229–235.
[39] Boaz Patt-Shamir, Dror Rawitz, and Gabriel Scalosub. 2008. Distributed approximation of cellular coverage. In *Principles of Distributed Systems*. Springer, 331–345.
[40] David Peleg. 2000. *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.
[41] Mirjam Wattenhofer and Roger Wattenhofer. 2004. *Distributed weighted matching*. Springer.