Vol.2021-AL-184 No.1
2021/8/25

A Subquadratic-Time Distributed Algorithm for Exact Maximum Matching

Naoki Kitamura[1,a)] Taisuke Izumi[2,b)]

Abstract: For a graph $G = (V, E)$, finding a set of disjoint edges that do not share any vertices is called a matching problem, and finding the maximum matching is a fundamental problem in the theory of distributed graph algorithms. Although local algorithms for the approximate maximum matching problem have been widely studied, exact algorithms has not been much studied. In fact, no exact maximum matching algorithm that is faster than the trivial upper bound of $O(n^2)$ rounds is known for general instances. In this paper, we propose a randomized $O(s_{\max}^{3/2} + \log n)$-round algorithm in the CONGEST model, where $s_{\max}$ is the size of maximum matching. This is the first exact maximum matching algorithm in $o(n^2)$ rounds for general instances in the CONGEST model. The key technical ingredient of our result is a distributed algorithms of finding an augmenting path in $O(s_{\max})$ rounds, which is based on a novel technique of constructing a *sparse certificate* of augmenting paths, which is a subgraph of the input graph preserving at least one augmenting path. To establish a highly parallel construction of sparse certificates, we also propose a new characterization of sparse certificates, which might also be of independent interest.

1. Introduction

1.1 Background and Our Result

A fundamental graph problem is the *maximum (unweighted) matching* problem of finding the maximum cardinality subset of edges not sharing endpoints. In this study, we address the problem of computing exact maximum matchings in a distributed setting, namely, the *CONGEST model*. The CONGEST model is a standard computational model for distributed graph algorithms, where the network is modeled as an undirected graph $G = (V, E)$ of n nodes and m edges. Each node executes the deployed algorithm following round-based synchrony, and each link can transfer a small message of $O(\log n)$ bits per round.

Many studies in the context of approximation algorithms provide insight into the complexity of the maximum matching problem. Table 1 lists the known algorithms, where $s_{\max}$ is defined the cardinality of the maximum matching. While $O(1)$ approximation admits local solutions (i.e., $o(D)$-round algorithms), the complexity of the exact maximum matching problem makes it expensive. Precisely, following the lower bound of Ben-Basat et al. [5], there exists an instance of diameter $\Omega(n)$ and maximum matching size $\Omega(n)$ that exhibits an $\Omega(n)$-round lower bound. This lower bound was originally proved in the *LOCAL* model, but it trivially holds in the CONGEST model as well. Therefore, the exact maximum matching problem is placed in the class of global problems. Parametrizing the complexity by both n and D, it is possible to obtain the non-trivial lower bound of $\Omega(D+\sqrt{n})$ rounds for the exact computation of the maximum matching[*1]. However, the corresponding upper bound is yet to be found. For the exact maximum matching problem in general graphs, no known algorithm achieves non-trivial $O(n^2)$ rounds. In addition, Bacrach et al. [3] pointed out that the bound of $\Omega(\sqrt{n}+D)$ rounds is a strong barrier because the standard framework of two-party communication complexity is unlikely to deduce any improved lower bound. These observations demonstrate the difficulty of revealing the in-

[1] Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya, Aichi, Japan
[2] Osaka University, Yamada-Oka, Suita, Osaka, Japan.
[a)] ktmr522@yahoo.co.jp.
[b)] t-izumi@ist.osaka-u.ac.jp
The full version of this paper is available in Arxiv [13].

[*1] This lower bound was not explicitly shown in previous literatures, but it is derived using the lower-bound graph almost same as that used in the lower-bound proof for the fractional maximum matching by Ahmadi et al. [2]

herent complexity of the exact maximum matching in the CONGEST model.

The objective of this paper is to shed light on the complexity gap of the exact maximum matching problem in the CONGEST model. We present the main theorem of this paper in the CONGEST model below.

Theorem 1. *For any input graph G, there exists a randomized CONGEST algorithm to compute the maximum matching that terminates within $O\left(s_{\max}^{3/2} + \log n\right)$ rounds with probability $1-1/n^{\Theta(1)}$.*

To the best of our knowledge, the proposed algorithm is the first to compute the exact maximum matching algorithm in $o(n^2)$ rounds for general input instances in the CONGEST model.

1.2 Technical Outline

Our algorithm follows the standard technique of finding *augmenting paths*. If an augmenting path is found, the current matching is improved by flipping the labels of matching edges and non-matching edges along the path. It is well known that the current matching is the maximum if and only if there exists no augmenting path in G with respect to the current matching. Hence, the maximum matching problem is reduced to the task of finding augmenting paths $s_{\max}$ times. In the CONGEST model, this approach faces difficulty in the situation where any augmenting path with respect to the current matching is long (i.e., consisting of $\Theta(n)$ edges). It should be emphasized that BFS-like approaches do not work for finding augmenting paths in general graphs because the shortest alternating walk is not necessarily simple because of the existence of odd cycles. Thus, it is not trivial to even compute an augmenting path with a running time linearly dependent on its length. The key ingredient of our approach is two new algorithms for finding augmenting paths. They run in $O(\ell^2)$ rounds and $O(s_{\max})$ rounds respectively, where ℓ is the length of the shortest augmenting path for the current matching. Roughly, our algorithm switches between these two algorithms according to the current matching size. The running-time bound is obtained using the following seminal observation by Hopcroft and Karp:

Proposition 1 (Hopcroft and Karp [11])**.** *Given a matching $M \subseteq E$ of a graph G, there always exists an augmenting path of length less than $\lfloor 2s_{\max}/k \rfloor$ if the current matching size is at most the maximum matching size $s_{\max}$ minus k.*

Our augmenting path algorithms utilize Ahmadi's verification algorithm of maximum matching [1], in which each node returns the length of the shortest odd/even alternating paths from a given source (unmatched) node. The construction of the $O(\ell^2)$-round algorithm is relatively straightforward. It is obtained by iteratively finding the predecessor of each node in an augmenting path by sequential $O(\ell)$ invocations of the verification algorithm. The technical highlight of the proposed algorithm is the design of the $O(s_{\max})$-round algorithm. The $O(s_{\max})$-round algorithm constructs a *sparse certificate*, which is a sparse (i.e., containing $O(s_{\max})$ edges) subgraph of G preserving the reachability between two nodes by alternating paths. That is, a sparse certificate contains an augmenting path if and only if the original graph admits an augmenting path. By the sparseness property, a node can collect all the information on the sparse certificate within $O(s_{\max})$ rounds, trivially allowing the centralized solution of finding augmenting paths. To establish a highly parallel construction of sparse certificates, we also propose a new characterization of sparse certificates, which might also be of independent interest.

1.3 Related Works

In the LOCAL model, it is known that no $o(1/\epsilon)$ algorithm exists for the $(1-\epsilon)$-approximate maximum matching problem [5]. Together with the $\Omega(\sqrt{\log n}/\log\log n)$-round lower bound reported by Kuhn et al. [15], the lower bound in the LOCAL model is obtained as $\Omega(1/\epsilon + \sqrt{\log n/\log\log n}) = ((\log n)/\epsilon)^{\Omega(1)}$. Ghaffari et al. [10] showed a $((\log n)/\epsilon)^{O(1)}$ upper bound for the $(1-\epsilon)$ approximate maximum matching problem. By combining these results, we infer that the time complexity of solving the $(1-\epsilon)$ approximate maximum matching problem is $(\log n/\epsilon)^{\Theta(1)}$ in the LOCAL model. Ben-Basat et al. also proved the lower bound as $\Omega(|M|)$ in the LOCAL model [5].

Many literatures have addressed the maximum matching problem in the CONGEST model (see Table 1). Loker et al. [16] presented the first approximation algorithm in the CONGEST model, which is a randomized algorithm to compute $(1-\epsilon)$-approximate maximum matching in $O(\log n)$ rounds for any constant $\epsilon > 0$. The running time of the algorithm depends exponentially on $1/\epsilon$. Bar Yehuda et al. [4] improved the algorithm and proposed an $O(\log\Delta/\log\log\Delta)$-round algorithm of computing $(1-\epsilon)$-approximate matching for any constant $\epsilon > 0$, where Δ is maximum degree of the graph. Fabin et al. [14] has shown of $\Omega(\log\Delta/\log\log\Delta)$ rounds if $\log\Delta \leq \sqrt{\log n}$ holds. Ben-Basat et al. [5] proposed a deterministic $\tilde{O}(s_{\max}^2)$-round CONGEST algorithm. They also proposed a $(1/2-\epsilon)$ approximate algorithm in $\tilde{O}(s_{\max} + (s_{\max}/\epsilon)^2)$ rounds. Ahmadi et al. [2] proposed a deterministic $(2/3-\epsilon)$ approximate maximum matching algorithm in general graphs, which runs in $O(\log\Delta/\epsilon^2 + (\log^2\Delta + \log^* n)/\epsilon)$ rounds. They also presented an $\tilde{O}(M)$-round algorithm and $O((\log^2\Delta + \log^* n)/\epsilon)$-round $(1-\epsilon)$ approximate al-

gorithm in bipartite graphs. However, no $o(n^2)$-round algorithm for solving the exact maximum matching problem in the CONGEST model has been proposed so far.

In addition to distributed computing, many studies have considered centralized exact maximum matching algorithms. Edmonds presented the first centralized polynomial-time algorithm for the maximum matching problem [7,8] by following the seminal blossom argument. Hopcroft and Karp proposed a phase-based algorithm of finding multiple augmenting paths [11]. Their algorithm finds a maximal set of pairwise disjoint shortest augmenting paths in each phase. They showed that $O(\sqrt{n})$ phases suffice to compute the maximum matching and proposed an algorithm of implementing one phase in $O(m)$ time for bipartite graphs. Several studies have reported phase-based algorithms for general graphs that attain $O(\sqrt{n}m)$ time [6,9,17].

2. Preliminaries

2.1 CONGEST Model

The vertex set and edge set of a given graph G are, respectively, denoted by $V(G)$ and $E(G)$. A distributed system is represented by a simple undirected connected graph $G = (V(G), E(G))$. Let n and m be the numbers of nodes and edges, respectively. The diameter of a given subgraph $H \subseteq G$ is denoted by $D(H)$. Nodes and edges are uniquely identified by integer values, which are represented by $O(\log n)$ bits. The set of edges incident to $v \in V(G)$ is denoted by $I_G(v)$. In the CONGEST model, the computation follows round-based synchrony. In one round, each node v sends and receives $O(\log n)$-bit messages through the edges in $I_G(v)$ and executes local computation following its internal state, local random bits, and received messages. It is guaranteed that every message sent in a round is delivered to the destination within the same round. Each node has no prior knowledge of the network topology, except for its neighborhood IDs. We use the labeling of nodes and/or edges for specifying inputs and outputs of algorithms. Each node has information on the label(s) assigned to itself and those assigned to its incident edges. A *walk* W of G is an alternating sequence $W = v_0, e_1, v_1, e_2, \ldots, e_\ell, v_\ell$ of vertices and edges such that $e_i = (v_{i-1}, v_i)$ holds for any $1 \leq i \leq \ell$. A walk W is often treated as a subgraph of G. A walk $W = v_0, e_1, v_1, e_2, \ldots, e_\ell, v_\ell$ is called a (simple) *path* if every vertex in W is distinct. For any walk $W = v_0, e_1, v_1, \ldots, v_\ell$ of G, we define $W \circ u$ as the walk obtained by adding u, satisfying $(v_\ell, u) \in E(G)$, to the tail of W. For any edge $e = (v_\ell, u)$, we also define $W \circ e = W \circ u$. Given a walk W containing a node u, we denote by W_u^p and W_u^s the prefix of W up to u and the suffix of W from u, respectively. We also denote the inversion of the walk $W = v_0, e_1, v_1, \ldots, v_\ell$ (i.e., the walk $v_\ell, e_\ell, v_{\ell-1}, e_{\ell-1}, \ldots, v_0$) by $\overline{W}$. The length of a walk P is represented by $|P|$.

2.2 Matching and Augmenting Path

For a graph $G = (V, E)$, a matching $M \subseteq E$ is a set of edges that do not share endpoints. A node v is called a *matched node* if M intersects $I_G(v)$, or an *unmatched node* otherwise. A path $P = v_0, e_0, v_1, e_1, \ldots, v_\ell$ is called an *alternating path* if $\mathbf{I}_M(e_i) + \mathbf{I}_M(e_{i+1}) = 1$ holds for any $1 \leq i \leq \ell - 1$[*2]. If the length $|P|$ of P satisfies $|P| \mod 2 = \theta$, P is called *θ-alternating*. The value θ is called the *parity* of P. By definition, any 0-alternating (1-alternating) path from an unmatched node f finishes with a non-matching (matching) edge. Oue to a technical issue, we regard the path of length zero as a 0-alternating path. For any $\theta \in \{0, 1\}$ and $u, v \in V(G)$, we define $r^\theta(u, v)$ as the length of the shortest θ-alternating path between u and v. An *augmenting path* is an alternating path connecting two unmatched nodes. We say that (G, M) has an augmenting path if there exists an augmenting path in G with respect to M. The following proposition is a well-known fact in the maximum matching problem.

Proposition 2. *Given a matching $M \subseteq E(G)$ of graph G, M is the maximum matching if and only if (G, M) has no augmenting path.*

2.3 Approximate Maximum Matching

Our algorithm uses an $O(1)$-approximate upper bound for the maximum matching size of the input graph. To obtain the upper bound, we run the $O(\log n)$-round randomized maximal matching algorithm [12] as a preprocessing step. Let M^* be the computed maximal matching. Since any maximal matching is a (1/2)-approximate maximum matching, one can obtain the bound $2|M^*| \geq s_{\max}$. The size $s_{\max}$ is at least half of the diameter $D(G)$, and thus we can spend $O(D(G)) = O(s_{\max})$ rounds for counting and propagating the number of edges in M^*. That is, it is possible to provide each node with the value of $2|M^*|$ by the preprocessing of $O(D(G) + \log n) = O(s_{\max} + \log n)$ rounds. In the following argument, we denote $\hat{s} = 2s^*$, the value of which is available to each node.

2.4 Maximum-Matching Verification Algorithm

Our algorithm uses the algorithm by Ahmadi et al.'s [1] for maximum-matching verification as a building block. Although the original algorithm is

[*2] The indicator function $\mathbf{I}_X(x)$ returns one if $x \in X$ and zero otherwise.

Table 1 Lower and upper bounds of the maximum matching in the CONGEST model

Algorithm	Time Complexity	Approximation Level	Remark
Ben-Basat et al. [5]	$\Omega(\lvert s_{\max}\rvert)$	exact	LOCAL
Fabin et al. [14]	$\Omega\left(\frac{\log\Delta}{\log\log\Delta}\right)$	constant ϵ	$\log\Delta \leq \sqrt{\log n}$
Ben-Basat et al. [5]	$\Omega\left(\frac{1}{\epsilon}\right)$	$1-\epsilon$	LOCAL
Kuhn et al. [15]	$\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$	$1-\epsilon$	LOCAL
Ben-Basat et al. [5]	$\tilde{O}(s_{\max}^2)$	exact	
Ahmadi et al. [2]	$\tilde{O}\left(s_{\max}\right)$	exact	bipartite
Bar-Yehuda et al. [4]	$O\left(\frac{\log\Delta}{\log\log\Delta}\right)$	constant ϵ	
Lotker et al. [16]	$O\left(\frac{2^{\frac{2}{\epsilon^2}}\log s_{\max}\log n}{\epsilon^4}\right)$	$1-\epsilon$	
Ahmadi et al. [2]	$O\left(\frac{\log^2\Delta+\log^* n}{\epsilon}\right)$	$1-\epsilon$	bipartite
Ben-Basat et al. [5]	$\tilde{O}\left(s_{\max}+\left(\frac{s_{\max}}{\epsilon}\right)^2\right)$	$\frac{1}{2}-\epsilon$	
Ahmadi et al. [2]	$O\left(\frac{\log\Delta}{\epsilon^2}+\frac{\log^2\Delta+\log^* n}{\epsilon}\right)$	$\frac{2}{3}-\epsilon$	
Our result	$\boldsymbol{\tilde{O}\left(s_{\max}^{3/2}\right)}$	**exact**	

designed for the verification of maximum matching, it provides each node with information on the length of alternating paths to the closest unmatched nodes. Precisely, the following lemma holds.

Theorem 2 (Ahmadi et al. [1]). *Assume that a graph $G=(V,E)$ and a matching $M \subseteq E$ are given, and let W be the set of all unmatched nodes. There exist two $O(\ell)$-round randomized CONGEST algorithms $\mathsf{MV}(M,\ell,f)$ and $\mathsf{PART}(M,\ell)$ that output the following information at every node $v \in V(G)$ with a probability of at least $1-1/n^c$ for an arbitrarily large constant $c>1$.*

(1) Given M, a nonnegative integer ℓ, and a node $f \in W$, $\mathsf{MV}(M,\ell,f)$ outputs the pair $(\theta, r^\theta(f,v))$ at each node v if $r^\theta(f,v) \leq \ell$ holds (if the condition is satisfied for both $\theta=0$ and $\theta=1$, v outputs two pairs). The algorithm $\mathsf{MV}(M,\ell,f)$ is initiated only by the node f (with the value ℓ), and other nodes do not require information on the ID of f and value ℓ at the initial stage.

(2) The algorithm $\mathsf{PART}(M,\ell)$ outputs a partition $V^1, V^2, \ldots, V^N$ of $V(G)$ (as the label i for each node in V^i) such that (a) the subgraph G^i induced by V^i contains exactly two unmatched nodes f^i and g^i as well as an augmenting path between f_1^i and g_2^i of length at most ℓ and (b) the diameter of G^i is $O(\ell)$.

While the original paper [1] presents a single algorithm returning the outputs of both MV and PART, we intentionally separate it into two algorithms with different roles for clarity. Note that our matching-construction algorithm uses random bits only in the runs of these algorithms. As our algorithm activates them in $O(\mathrm{poly}(n))$ time as subroutines, we can guarantee that our algorithm has a high probability of success by taking a sufficiently large c. Hence, we do not pay much attention to the failure probability of our algorithm. Any stochastic statement in the following argument also holds with probability $1-n^c$ for an arbitrary constant $c>1$.

3. Computing the Maximum Matching in CONGEST

As explained in the introduction, the maximum matching problem is reducible to the problem of finding an augmenting path. We first present two key results below.

Lemma 1. *Let M be a matching of G. Provided that (G,M) has exactly two unmatched nodes $f,g \in V_G$ and contains an augmenting path of length at most ℓ between f and g, there exists an $O(\ell^2)$-round randomized algorithm that outputs an augmenting path connecting f and g.*

Lemma 2. *Let M be a matching of G. Provided that (G,M) has exactly two unmatched nodes $f,g \in V_G$ and contains an augmenting path between f and g, there exists an $O(n)$-round randomized algorithm that outputs an augmenting path that includes f.*

The outputs of both algorithms are the labels to the edges in the computed augmented path. To prove the lemmas, one can utilize the output of the algorithm PART. We first run the verification algorithm $\mathsf{PART}(M,\ell)$ (for Lemma 1) or $\mathsf{PART}(M,\hat{s})$ (for Lemma 2) as a preprocessing step and then execute the algorithms of Lemma 1 or 2 for each G^i output by PART independently. Note that each G^i contains only matched nodes and two unmatched nodes; thus, $|V(G^i)| \leq 2|M|+2$ holds for any G^i. Then, the following corollary is deduced:

Corollary 1. *There exist two randomized algorithms* $\mathsf{A}(M,\ell)$ *and* $\mathsf{B}(M)$ *satisfying the following conditions, respectively:*

- *For any graph* $G=(V,E)$ *and matching* $M\subseteq E$, $\mathsf{A}(M,\ell)$ *finds a nonempty set of vertex-disjoint augmenting paths within* $O(\ell^2)$ *rounds if* (G,M) *has an augmenting path of length at most* ℓ.
- *For any graph* $G=(V,E)$ *and matching* $M\subseteq E$, $\mathsf{B}(M)$ *finds a nonempty set of vertex-disjoint augmenting paths of* (G,M) *within* $O(|M|)$ *rounds if* (G,M) *has an augmenting path.*

We present an $O\left(s_{\max}^{3/2}+\log n\right)$-round algorithm for computing the maximum matching using the algorithms $\mathsf{A}(M,\ell)$ and $\mathsf{B}(M)$. The pseudocode of the whole algorithm is presented in Algorithm 1. It basically follows the standard idea of centralized maximum matching algorithms, i.e., finding an augmenting path and improving the current matching iteratively. The first $\hat{s}-\sqrt{\hat{s}}$ iterations use $\mathsf{A}(M,\ell)$ (lines 1–4), and the remaining $\sqrt{\hat{s}}$ iterations use $\mathsf{B}(M)$. In the i-th iteration, the algorithm $\mathsf{A}(M,\ell)$ runs with $\ell=\lceil 2\hat{s}/(2\hat{s}-i)\rceil$. This setting comes from Proposition 1. The improvement of the current matching by a given augmenting path is simply a local operation and is realized by flipping the labels of matching edges and non-matching edges on the path.

Lemma 3. *Algorithm 1 constructs a maximum matching with high probability in* $O\left(s_{\max}^{3/2}+\log n\right)$ *rounds.*

The following sections are devoted to proving Lemmas 1 and 2. Since the presented algorithms are intended to run in each G^i returned by the preprocessing run of $\mathsf{PART}(M,\cdot)$, without loss of generality, we assume that G has exactly two unmatched nodes f and g with an augmenting path between them. In addition, it is assumed that one of f and g is elected as a primary unmatched node (referred to as f hereafter). This election process is easily implemented in $O(\ell)$ rounds because the distance between f and g is at most ℓ. When we argue the existence of augmenting or alternating paths in a subgraph $H=(V(H),E(H))$ of G, the matching $M\cap E(H)$ of graph H is considered without explicit notice. Given a subgraph $H\subseteq G$, we denote the length of the shortest odd (even) alternating path from f to v in H by $r_H^1(f,v)$ ($r_H^0(f,v)$). If no odd or even alternating path exists from f to v in H, then we define $r_H^1(f,v)=\infty$ or $r_H^0(f,v)=\infty$. As sentinels, we also define $r_H^0(f,f)$ as ∞ and $r_H^1(f,f)$ as 0.

4. Construction of Augmenting Path in $O(\ell^2)$ Rounds

Let $P=v_0,v_1,\ldots,v_\ell$ be the shortest augmenting path from f to g (i.e., $f=v_0$ and $g=v_\ell$) and $P_i=P_{v_i}^s$ for short. The key idea of the algorithm is to find the predecessor of each node v_i along P sequentially. Note that it does not suffice to choose a neighbor v of v_i with $r_G^\theta(f,v)=i-1$ and $\mathbf{I}_M(v_i,v)=\theta$ for $\theta=(i-1) \mod 2$ as the predecessor. This strategy is problematic in the scenario in which there exists two neighbors v and u such that $r_G^\theta(f,v)=r_G^\theta(f,u)=i-1$ and $\mathbf{I}_M(v_i,u)=\mathbf{I}_M(v_i,v)=\theta$ for $\theta=(i-1) \mod 2$, where u is the correct successor. While v is guaranteed to have the alternating path Q from f to v of length $i-1$, it can intersect P_i. Then, the concatenation $Q\circ(v_i,v)\circ P_i$ is not simple. That is, it is not an augmenting path. To avoid this scenario, the algorithm finds the predecessor of v_i in the graph $G-P_i$, where $G-P_i$ is the induced graph by $V(G)\backslash V(P_i)$. If some neighbor v of v_i satisfies $r_{G-P_i}^\theta(f,v)=i-1$ and $\mathbf{I}_M(v_i,v)=1-\theta$, the concatenated walk $Q\circ(v_i,v)\circ P_i$ is guaranteed to be simple.

Algorithm 2 details the algorithm for constructing the augmenting path in $O(\ell^2)$ rounds. The algorithm consists of ℓ steps. In the i-th step, it finds the predecessor of $v_{\ell-i}$. Assume that the algorithm has already found $P_{\ell-i+1}$ at the beginning of the i-th step. Any node in $V(P_{\ell-i+1})\backslash\{v_{\ell-i+1}\}$ quits the algorithm (with the information of the predecessor in P_i), and thus, the nodes still running the algorithm are given by $V(G-P_{\ell-i+1})$. If i is even, the edge $(v_{\ell-i},v_{\ell-i+1})$ is the matching edge, and thus, the algorithm determines the neighbor of $v_{\ell-i+1}$ connected by the edge with M as the predecessor. Otherwise, the nodes still participating in the algorithm run $\mathsf{MV}(M,\ell-i+1,f)$ (that is, they run in the graph $G-P_{\ell-i+1}$) The algorithm decides an arbitrary neighbor v of v_i satisfying $r_{G-P_{\ell-i+1}}^0(f,v)=\ell-i-1$ and $\mathbf{I}_M(v,v_i)=0$ as the predecessor of $v_{\ell-i}$.

Lemma 4. *Algorithm 2 constructs an augmenting path between* f *and* g *with high probability in* $O(\ell^2)$ *rounds.*

Theorem 1 trivially follows from Lemma 4.

5. Construction of Augmenting Path in $O(n)$ Rounds

We first introduce several auxiliary notions and definitions. Given a subgraph $H\subseteq G$ and $\theta\in\{0,1\}$, a node $v\in V_H$ is called *θ-reachable* in H if $r_H^\theta(f,v)$ is finite. In addition, v is called *bireachable* in H if it is both 1-reachable and 0-reachable in H. A node that is neither 1-reachable nor 0-reachable in H is called *unreachable* in H. A node that is θ-reachable for some $\theta\in\{0,1\}$ in H but not bireachable in H is called *strictly θ-reachable* in H. Given two spanning subgraphs H_1 and H_2 of G, we say that a node $v\in V(H_1)$ *preserves the reachability* of H_2 in H_1 if for any $\theta\in\{0,1\}$, the θ-reachability of v in H_2 implies that in H_1. A graph H_1 is said to preserve the reachability of H_2 if any node $v\in V(H_1)$ preserves

Algorithm 1 Constructing a maximum matching in $O(n^{3/2})$ rounds.

for $i = 1; i \leq \hat{s} - \sqrt{\hat{s}}; i++$ **do**
  run the algorithm $\mathsf{A}(M, \ell)$ with $\ell = \lceil 2\hat{s}/(\hat{s} - i) \rceil$ for $O(\ell)$ rounds.
  if $\mathsf{A}(M, \ell)$ finds a nonempty set of vertex-disjoint augmenting paths within $O(\ell)$ rounds, **then**
    improve the current matching using the set of vertex-disjoint augmenting paths.
for $i = 1; i \leq \sqrt{\hat{s}}; i++$ **do**
  run the algorithm $\mathsf{B}(M)$ for $O(\hat{s})$ rounds.
  if $\mathsf{B}(M)$ finds a nonempty set of vertex-disjoint augmenting paths within $O(\hat{s})$ rounds, **then**
    improve the current matching M using the set of vertex-disjoint augmenting paths.

Algorithm 2 Construction of the augmenting path $CAP((G, M), f, g, \ell)$ for node v_i.

Require: The path P_0 is an augmenting path with length ℓ from f to g.
$P_0, P_1, \ldots, P_\ell$: initially $\emptyset$.
$\mathsf{target} = g$
for $i = 1; i \leq \ell; i++$ **do**
  if h is even **then**
    target chooses the node $v_{\ell-i}$ that satisfies $\mathbf{I}_M((\mathsf{target}, v_{\ell-i})) = 1$.
    $P_{\ell-i} \leftarrow P_{\ell-i+1} \cup \{(\mathsf{target}, v_{\ell-i})\}$, $\mathsf{target} \leftarrow v_{\ell-i}$.
  else
    run the algorithm $\mathsf{MV}(M, \ell - i, f)$ with the subgraph $H_{\ell-i+1}$ induced by $V(G - P_{\ell-i+1})$ as the input.
    for any $v \in V(G - P_{\ell-i+1})$, the node v sends $r^0_{H_{\ell-i+1}}(f, v)$ to its neighborhood.
    target chooses the node $v_{\ell-i}$ that satisfies $\mathbf{I}_M((\mathsf{target}, v_{\ell-i})) = 0$ and $r^0_{H_{\ell-i+1}}(f, v_{\ell-i}) = \ell - i$.
    $P_{\ell-i} \leftarrow P_{\ell-i+1} \cup \{(\mathsf{target}, v_{\ell-i})\}$, $\mathsf{target} \leftarrow v_{\ell-i}$.

the reachability of H_2 in H_1, which is denoted by $H_1 \succ H_2$. We define $r_H(f, v) = \min_{\theta \in \{0,1\}} r^\theta_H(f, v)$ and $\gamma_H(v) = \mathrm{argmin}_{\theta \in \{0,1\}} r^\theta_H(f, v)$. Note that $r^0_H(f, v) = r^1_H(f, v)$ does not hold, because $r^0_H(f, v)$ is even and $r^1_H(f, v)$ is odd. When $r^0_H(f, v) = \infty$ and $r^1_H(f, v) = \infty$ hold, $\gamma_H(v)$ is defined as zero. We assume that any node v unreachable from f in G does not join our algorithm. Therefore, without loss of generality, we assume that none of the nodes $v \in V_G$ are unreachable in G without loss of generality. In addition, we assume that any node $v \in V_G$ has information on the values of $r^0_G(f, v)$ and $r^1_G(f, v)$ at the beginning of the algorithm. This assumption is realized by activating $\mathsf{MV}(M, n, f)$ as a preprocessing step.

The key idea of our proof is to construct a *sparse certificate* H, which is a spanning subgraph $H \subseteq G$ of $O(n)$ edges satisfying $H \succ G$. If such a graph is obtained, the trivial centralized approach (i.e., the approach in which f collects the whole topological information of H) yields an $O(n)$-round algorithm for constructing the augmenting path. For constructing sparse certificates, we first introduce a novel tree structure associated with G, M, and f:

Definition 1 (Alternating base tree)**.** *An* alternating base tree *for G, M, and f is the rooted spanning tree T of G satisfying the following conditions:*

- *f is the root of T.*
- *For any $v \in V(G)$, the edge from v to its parent in T is the last edge of the shortest alternating path from f to v in G. Formally, letting $\mathsf{par}_T(v)$ be the parent of $v \in V(G) \setminus \{f\}$ in T, $r^{\gamma_G(v)}_G(f, v) = r^{1-\gamma_G(v)}_G(f, \mathsf{par}_T(v)) + 1$ and $\mathbf{I}_M((v, \mathsf{par}_{T_l}(v))) = 1 - \gamma_G(v)$ hold for any $v \in V(G) \setminus \{f\}$.*

It is not difficult to check that such a spanning tree always exists. Fixing T, the subscript T of the notation $\mathsf{par}_T(v)$ is omitted in the following argument. We define $\mathsf{ep}(v)$ as the edge from v to its parent and T_v as the subtree of T rooted by v. Any non-tree edge $e = (u, w) \in E(G) \setminus E(T)$ and the unique path from u to w in T form a simple cycle in G, which is denoted by $\mathsf{cyc}(e)$.

The sparse certificate is obtained by incrementally augmenting edges to T. For any $1 \leq k \leq n$, we define the *level-k edge set* F_k as $F_k = \{(u, v) \mid (u, v) \in E(G) \setminus M \wedge \max(r^0_G(f, u), r^0_G(f, v)) = k\} \cup \{(u, v) \mid (u, v) \in M \wedge \max(r^1_G(f, u), r^1_G(f, v)) = k\}$. We also define $F_{\leq k} = \cup_{0 \leq i \leq k} F_k$ and $G_k = T + F_{\leq k}$. Moreover, we define $F_0 = \emptyset$ as a sentinel. Let B_k be the set of all the bridges (i.e., the edge forming a cut of size one) in G_k. Note that B_h is a subset of $E(T)$ because T is a spanning tree of G. The following lemma is the key technical ingredient of our construction.

Lemma 5. *Let $F^c_k \subseteq F_k \setminus E(T)$ be an arbitrary subset of non-tree edges in F_k satisfying $B_{k-1} \setminus B_k \subseteq \cup_{e \in F^c_k} E(\mathsf{cyc}(e))$. Then, $(T + \cup_{1 \leq i \leq k} F^c_i) \succ G_k$ holds. In addition, the edge set $F^c = \cup_{0 \leq i \leq k} F^c_i$ contains at*

most $n-1$ edges.

This lemma naturally yields the following incremental construction of sparse certificates: each node v identifies k such that $\mathsf{ep}(v) \in B_{k-1} \setminus B_k$ holds, and if T_v has an outgoing edge e belonging to F_k, v adds e to F_k^c (if F_k contains two or more outgoing edges, one is chosen arbitrarily). Since $\mathsf{cyc}(e)$ obliviously covers $\mathsf{ep}(v)$, the constructed edge set F_k^c satisfies the lemma. Consequently, $H = T + \cup_{1 \le i \le n} F_i^c \succ G_n$ is satisfied, and thus, H is a sparse certificate.

Considering the distributed construction of H, a useful property of Lemma 5 is that one does not have to wait for the computation of F_k^c to start the computation of F_{k+1}^c. As the information on $r_G^\theta(f, v)$ for $\theta \in \{0, 1\}$ is available to v, each node can identify the level of each incident edge. Thus, the construction of F_k^c for all k can be executed in parallel.

We explain how to implement the centralized sparse certificate algorithm in the CONGEST model to obtain the algorithm of Theorem 2. It is relatively straightforward to construct the alternating base tree T. From the preprocessing run of $\mathsf{MV}(M, n, f)$, each node v has information on the values of $r_G^1(f, v)$ and $r_G^0(f, v)$; thus, it has information on $\gamma_G(v)$ as well. Then, v chooses an arbitrary neighbor u of v satisfying the second condition of the alternating base tree as its parent (i.e., it chooses (v, u) as an edge of T). Algorithm 3 presents the pseudocode of the alternative base tree construction. This algorithm is a local algorithm, which is implemented in zero round.

The main idea of constructing the edge set $F^c = \cup_{1 \le i \le n} F_i^c$ in the distributed manner is implemented by the CONGEST algorithm $\mathsf{ConstF}(k)$, where each node v outputs an outgoing edge of T_v of level k if it exists (or $\perp$ otherwise). Let d be the height of the constructed alternating base tree T. Given a non-tree edge $e = (u, w) \in E(G) \setminus E(T)$, the depth of the lowest common ancestor of u and w is denoted by $\mathsf{lca}(e)$. In addition, we introduce the ordering relation $\le_{\mathsf{lca}}$ over all non-tree edges as $e_1 \le_{\mathsf{lca}} e_2$ if and only if $\mathsf{lca}(e_1) \le \mathsf{lca}(e_2)$. The algorithm ConstF works under the assumption that for any non-tree edge $e = (u, v)$, u and v have information on the value of $\mathsf{lca}(e)$. This assumption is realized by the following $O(d)$-round preprocessing.

(1) Each node v computes its depth d_v in T through a downward message propagation from f along T. The root f first sends to its children the value one. The node v receiving message i decides $d_v = i$ and sends the value $i + 1$ to its chldren.

(2) Each node v broadcasts the pair of its ID and depth (v, d_v) to all the nodes in T_v. First, each node sends the pair to its children. In the following rounds, each node forwards the message from its parents to the children. This task finishes within $O(d)$ rounds.

(3) The broadcast information of the previous step allows each node v to identify the path $p_T(v)$ from v to f in T. For all non-tree edges $e = (u, v)$, u and v exchange $p_T(v)$ (taking $O(d)$ rounds) and compute the value of $\mathsf{lca}(e)$.

The pseudocode of Algorithm $\mathsf{ConstF}(k)$ is presented in Algorithm 4. Let $E^*(T_v)$ be the set of non-tree edges e such that at least one endpoint of e belongs to $V(T_v)$. Each node v computes the minimum edge $e_v \in E^*(T_v) \cap F_k$ with respect to $\le_{\mathsf{lca}}$. This task is implemented through a standard aggregation over T. Each leaf node v sends the minimum edge e in $F_k \cap E^*(T_v)$. If $F_k \cap E^*(T_v) = \emptyset$ holds, the leaf sends a dummy edge e such that $\mathsf{lca}(e) = \infty$ holds (the edge sent to the parent is implicitly associated with the value of $\mathsf{lca}(e)$ to admit the comparison based on $\le_{\mathsf{lca}}$). Let X be the set of edges a non-leaf node v received from its children. Then, v chooses e_v as the minimum edge in $X \cup (I(v) \cap F_k \cap E^*(T_v))$ with respect to $\le_{\mathsf{lca}}$ and sends the chosen edge to $\mathsf{par}(v)$. Finally, v outputs e_v if $\mathsf{lca}(e_v) < d_v$ holds or $\perp$ otherwise. The edge set F^c is constructed by running $\mathsf{ConstF}(k)$ for all $1 \le k \le n$. As this algorithm is implemented by one-shot aggregation over T, one can utilize the standard pipelining technique for completing $\mathsf{ConstF}(k)$ for all $1 \le k \le n$, which takes $O(n)$ rounds in total (including the preprocessing step of computing $\mathsf{lca}(e)$). The result of ConstF provides node v with the information of the minimum k, such that $\mathsf{ep}(v) \in B_{k-1} \setminus B_k$, as well as an outgoing edge of T_v in F_k. Each node v can decide the edge e that should be added to $F^c = \cup_{1 \le i \le n} F_i^c$.

6. Conclusion

We proposed the randomized $O(s_{\max}^{3/2} + \log n)$-rounds (i.e. $O(n^{3/2})$-rounds) algorithm for computing a maximum matching in the CONGEST model, which is the first one of attaining $o(n^2)$-round complexity for general graphs. Our algorithm follows the standard augmenting-path approach, and the technical core lies two fast algorithms of finding augmenting paths respectively running in $O(\ell^2)$ and $O(s_{\max})$ rounds.

While we believe that our result is a big step toward the goal of revealing the tight round complexity of the exact maximum matching problem, the gap between the upper and lower bounds are still large. It should be noted that we leave the possibility of much faster augmenting path algorithms. Once an $o(\ell^2)$-round or $o(s_{\max})$-round algorithm of finding an augmenting path is invented, the upper bound automatically improves. This direction is still promising.

Algorithm 3 Construction of the alternating base tree for v_i: $\mathsf{ABT}((G, M))$

Require: The graph induced by the edge set $\bigcup_{i:v_i \in V} E_i$ is an alternating base tree.

E_i: initially $\emptyset$.
if $v \neq f$ **then**
  choose edge (u, v) that is incident on the vertex v and satisfies $r_I^{\gamma(v)}(f, v) = r_I^{1-\gamma(v)}(f, u)$ and $\mathbf{I}((u, v)) = 1 - \gamma(v)$ (if multiple edges satisfy these conditions, the node arbitrarily chooses one).
  $E_i \leftarrow E_i \cup (u, v)$.

Algorithm 4 Construction of F_k^c for v_i: $\mathsf{ConstF}(k)$

Require: The edge e_i is an outgoing edge of T_{v_i} if node v_i outputs e_i; otherwise, T_{v_i} does not have an outgoing edge.

for $i = 1; i \leq d; i++$ **do**
  if v_i is a leaf node **then**
    if $I(v_i) \cap F_k \cap E^*(T_v) = \emptyset$ **then**
      $e_{v_i} \leftarrow$ dummy edge e such that $\mathsf{lca}(e) = \infty$.
    else
      $e_{v_i} \leftarrow \min_{e \in I(v_i) \cap F_k \cap E^*(T_{v_i})} e$ w.r.t. $\leq_{\mathsf{lca}}$.
    if $v_i \neq f$ **then**
      send e_{v_i} to its parent.
  else
    if v_i receives the set of edges X from all its children **then**
      $e_{v_i} \leftarrow \min_{e \in X \cup (I(v_i) \cap F_k \cap E^*(T_{v_i}))} e$ w.r.t. $\leq_{\mathsf{lca}}$.
if $\mathsf{lca}(e_{v_i}) \leq d(v_i)$ **then**
  output e_v.
else
  output $\perp$.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Numbers JP19J22696, 20H04140, 20H04139, and 19K11824.

References

[1] Ahmadi, M. and Kuhn, F.: Distributed maximum matching verification in CONGEST, *34th International Symposium on Distributed Computing (DISC)*, pp. 37:1–37:18 (2020).
[2] Ahmadi, M., Kuhn, F. and Oshman, R.: Distributed approximate maximum matching in the congest model, *32rd International Symposium on Distributed Computing (DISC)*, pp. 6:1–6:17 (2018).
[3] Bacrach, N., Censor-Hillel, K., Dory, M., Efron, Y., Leitersdorf, D. and Paz, A.: Hardness of distributed optimization, *2019 ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 238–247 (2019).
[4] Bar-Yehuda, R., Censor-Hillel, K., Ghaffari, M. and Schwartzman, G.: Distributed approximation of maximum independent set and maximum matching, *36th annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 165–174 (2017).
[5] Ben-Basat, R., Kawarabayashi, K.-i. and Schwartzman, G.: Parameterized distributed algorithms, *33rd International Symposium on Distributed Computing (DISC)*, pp. 6:1–6:16 (2018).
[6] Blum, N.: A new approach to maximum matching in general graphs, *International Colloquium on Automata, Languages, and Programming*, pp. 586–597 (1990).
[7] Edmonds, J.: Maximum matching and a polyhedron with 0,1-vertices, *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, p. 125 (1965).
[8] Edmonds, J.: Paths, trees, and flowers, *Canadian Journal of mathematics*, pp. 449–467 (1965).
[9] Gabow, H. N. and Tarjan, R. E.: Faster scaling algorithms for general graph matching problems, *Journal of the ACM (JACM)*, pp. 815–853 (1991).
[10] Ghaffari, M., Kuhn, F. and Maus, Y.: On the complexity of local distributed graph problems, *49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pp. 784–797 (2017).
[11] Hopcroft, J. E. and Karp, R. M.: An n^5/2 algorithm for maximum matchings in bipartite graphs, *SIAM Journal on computing*, pp. 225–231 (1973).
[12] Israeli, A. and Itai, A.: A fast and simple randomized parallel algorithm for maximal matching, *Information Processing Letters*, pp. 77–80 (1986).
[13] Kitamura, N. and Izumi, T.: A Subquadratic-Time Distributed Algorithm for Exact Maximum Matching, *arXiv preprint arXiv:2104.12057* (2021).
[14] Kuhn, F., Moscibroda, T. and Wattenhofer, R.: The price of being near-sighted, *17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1109557–1109666 (2006).
[15] Kuhn, F., Moscibroda, T. and Wattenhofer, R.: Local computation: Lower and upper bounds, *Journal of the ACM (JACM)*, pp. 1–44 (2016).
[16] Lotker, Z., Patt-Shamir, B. and Pettie, S.: Improved distributed approximate matching, pp. 1–17 (2015).
[17] Vazirani, V. V.: A proof of the MV matching algorithm, *arXiv*.