IEICE TRANS. INF. & SYST., VOL.E105–D, NO.3 MARCH 2022

PAPER

A Subquadratic-Time Distributed Algorithm for Exact Maximum Matching

Naoki KITAMURA[†a)], ***Nonmember and*** **Taisuke IZUMI**[††], ***Member***

SUMMARY 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 have 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 $O(s_{\max}^{3/2})$-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.

key words: *distributed graph algorithm, maximum matching, congest model*

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. However, the limited bandwidth in the CONGEST model precludes a trivial universal solution for every graph problem, where the leader node collects all the topological information of G and solves the problem using a centralized algorithm. This approach takes $O(n^2)$ rounds in the worst case of $m = \Omega(n^2)$. The technical challenge in designing CONGEST algorithms concerns how each node computes a fragment of the solution without information on the whole input instance. The recent development of design techniques for CONGEST algorithms has yielded many efficient solutions for various graph problems such as the minimum spanning tree [1]–[6], distance problems including shortest-path computation [7]–[13], and flow and cut [14]–[18]. Owing to the existence of the $O(n^2)$-round universal algorithm, the weakest non-trivial challenge in the design of a CONGEST algorithms is to achieve a subquadratic $o(n^2)$-round upper bound. In contrast to the universal upper bound, all the problems listed above belong to the class of *global* problems exhibiting an $\Omega(D)$-round lower bound, where D is the diameter of the input graph G. Thus, the tight round complexities of global problems lie between $\Theta(n^2)$ and $\Theta(D)$. For many of global problems, near-tight complexity bounds, typically $\tilde{\Theta}(\sqrt{n} + D)$ rounds or $\tilde{\Theta}(n)$ rounds, have been proved [19]–[21].

Many studies in the context of approximation algorithms provide insight into the globality 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. [22], 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, which is like the CONGEST model with arbitrarily large messages. This lower bound 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 $\tilde{\Omega}(D + \sqrt{n})$ rounds for the exact computation of the maximum matching [23]. 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. [20] 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 give any improved lower bound. These observations demonstrate the difficulty of revealing the inherent 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 pa-

Manuscript received April 15, 2021.
Manuscript revised October 29, 2021.
Manuscript publicized December 17, 2021.

†The author is with the Nagoya Institute of Technology, Nagoya-shi, 466–8555 Japan.
††The author is with the Osaka University, Suita-shi, 565–0871 Japan.
a) E-mail: ktmr522@yahoo.co.jp

DOI: 10.1587/transinf.2021EDP7083

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

Algorithm	Time Complexity	Approximation Level	Remark
Ben-Basat et al. [22]	$\Omega(\lvert s_{\max}\rvert)$	exact	LOCAL
Kuhn et al. [26]	$\Omega\left(\frac{\log\Delta}{\log\log\Delta}\right)$	constant ϵ	$\log\Delta \leq \sqrt{\log n}$
Ben-Basat et al. [22]	$\Omega\left(\frac{1}{\epsilon}\right)$	$1-\epsilon$	LOCAL
Kuhn et al. [27]	$\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$	$1-\epsilon$	LOCAL
Ben-Basat et al. [22]	$\tilde{O}(s_{\max}^2)$	exact	
Ahmadi et al. [23]	$\tilde{O}(s_{\max})$	exact	bipartite
Bar-Yehuda et al. [28]	$O\left(\frac{\log\Delta}{\log\log\Delta}\right)$	constant ϵ	
Lotker et al. [29]	$O\left(\frac{2^{2\epsilon^{-2}}\log s_{\max}\log n}{\epsilon^4}\right)$	$1-\epsilon$	
Ahmadi et al. [23]	$O\left(\frac{\log^2\Delta+\log^* n}{\epsilon}\right)$	$1-\epsilon$	bipartite
Ben-Basat et al. [22]	$\tilde{O}\left(s_{\max}+\left(\frac{s_{\max}}{\epsilon}\right)^2\right)$	$\frac{1}{2}-\epsilon$	
Ahmadi et al. [23]	$O\left(\frac{\log\Delta}{\epsilon^2}+\frac{\log^2\Delta+\log^* n}{\epsilon}\right)$	$\frac{2}{3}-\epsilon$	
Our result	$\boldsymbol{O\left(s_{\max}^{3/2}\right)}$	**exact**	

per 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 $O\left(s_{\max}^{3/2}\right)$ 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. 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 [24]): 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 and Kuhn's verification algorithm of maximum matching [25], 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 [22]. Together with the $\Omega(\sqrt{\log n/\log\log n})$-round lower bound reported by Kuhn et al. [27], 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. [30] 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 of the maximum matching as $\Omega(s_{max})$ in the LOCAL model [22].

Many papers in the literatures have addressed the maximum matching problem in the CONGEST model (see Table 1). Lotker et al. [29] 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. [28] 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. Kuhn et al. [26] have shown a lower bound of $\Omega(\log\Delta/\log\log\Delta)$ rounds if $\log\Delta \leq \sqrt{\log n}$ holds. Ben-Basat et al. [22] 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. [23] 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 $\tilde{O}(s_{max})$-round algorithm and $O((\log^2\Delta + \log^* n)/\epsilon)$-round $(1-\epsilon)$ approximate algorithm 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 [31], [32] by following the seminal blossom argument. Hopcroft and Karp proposed a phase-based algorithm of finding multiple augmenting paths [24]. 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 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 [33]–[35].

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 is done in synchronous rounds. 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 $e_i = (v_{i-1}, v_i)$, $v_i \in V(G)$, and $e_i \in E(G)$ holds for any $1 \leq i \leq \ell$. The *length* of the walk W is a number of edges in W. 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 $1 \leq i \leq \ell - 1$[†]. If the length $|P|$ of P satisfies $|P| \bmod 2 = \theta$, 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 matching (non-matching) edge. Due 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. Note that the augmenting path must be 1-alternating path. 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 $O(s_{\max})$-round maximal matching algorithm as follows. First we suppose each edge has a unique ID and priority associated with the ID. We run a simple parallel greedy algorithm, where each node adds an edge to the matching if all neighboring higher priority edges are already known not be in the maximal matching. One iteration of the algorithm increases the matching size at least

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

by one, and thus $O(s_{\max})$ rounds suffices to obtain a maximal matching. Since $s_{max} = \Omega(D(G))$ always holds, the termination of maximal matching construction is detected in $O(s_{\max})$ rounds by checking whether all nodes terminated or not. Note that the termination detection is executed once in every $\Theta(D(G))$ rounds. 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 $2|M^*|$ in a preprocessing phase using only $O(D(G)) = O(s_{\max})$ 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 [25] for maximum-matching verification as a building block. Although the original algorithm is 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. [25]): 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) For $1 \leq i \leq N - 1$, 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^i and g^i of length at most ℓ and (b) the diameter of G^i is $O(\ell)$. If G contains an augmenting path, $\mathsf{PART}(M, \ell)$ always returns at least two sets of vertices, otherwise $\mathsf{PART}(M, \ell)$ returns the set of vertices, that is, $V^N = V(G)$. Note that, $\mathsf{PART}(M, \ell)$ can be applied in a subgraph.

While the original paper [25] 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 uses them only $O(\text{poly}(n))$ times 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. If the edge e is included in the augmenting path, then the vertices connecting to e know that e is included in the augmenting path. Each node adds the edge e to a matching M if the edge e is included in the augmenting path and is not included in the matching, and removes the edge e from a matching M if the edge e is included in the augmenting path and the matching. 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 $O\left(s_{\max}^{3/2}\right)$-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 aug-

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

```
1: for i = 1; i ≤ ŝ − √ŝ; i++ do
2:     run the algorithm A(M, ℓ) with ℓ = ⌈2ŝ/(ŝ − i)⌉ for O(ℓ) rounds.
3:     if A(M, ℓ) finds a nonempty set of vertex-disjoint augmenting paths within O(ℓ) rounds, then
4:         improve the current matching using the set of vertex-disjoint augmenting paths.
5: for i = 1; i ≤ √ŝ; i++ do
6:     run the algorithm B(M) for O(ŝ) rounds.
7:     if B(M) finds a nonempty set of vertex-disjoint augmenting paths within O(ŝ) rounds, then
8:         improve the current matching M using the set of vertex-disjoint augmenting paths.
```

menting 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. The correctness and running time of Algorithm 1 are analyzed below.

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

Proof : Let $s(i)$ be the matching size at the end of i iterations of the algorithm $\mathsf{A}(M,\ell)$. We show that $s(\hat{s}-s_{\max}+j) \ge j$ holds for any $0 \le j \le s_{\max} - \sqrt{\hat{s}}$. It implies that the matching size is at least $s_{\max} - \sqrt{\hat{s}}$ after the application of $\mathsf{A}(M,\cdot)$. Therefore, the maximum matching is constructed by $\sqrt{\hat{s}}$ iterations of the algorithm $\mathsf{B}(M)$. The proof of the statement above follows the induction on j. (Basis) If $j = 0$, the statement trivially holds. (Inductive step) As the induction hypothesis, suppose $s(\hat{s} - s_{\max} + j') \ge j'$ holds. If $s(\hat{s} - s_{\max} + j') \ge j' + 1$, then the statement holds. Therefore, we consider the case in which $s(\hat{s} - s + j') = j'$ holds. By Proposition 1, there exists an augmenting path of length at most $\lfloor 2s_{\max}/(s_{\max} - j')\rfloor \le 2\hat{s}/(s_{\max} - j') = 2\hat{s}/(\hat{s} - (\hat{s} - s_{\max} + j')) \le 2\hat{s}/(\hat{s} - (\hat{s} - s_{\max} + (j' + 1)))$ at the end of $\hat{s} - s_{\max} + j'$ iterations of the algorithm $\mathsf{A}(M,\ell)$. Hence, the size of the matching is increased by at least one in the $(\hat{s} - s_{\max} + j' + 1)$-th iteration.

Now, we show the running-time analysis of Algorithm 1. Recall that $\hat{s} = \Theta(s_{max})$ holds. As $\mathsf{A}(M,\ell)$ is repeated $\hat{s} - \sqrt{\hat{s}}$ times and $\mathsf{B}(M)$ is repeated $\sqrt{\hat{s}}$ times, the running time of Algorithm 1 is as follows.

$$
\begin{aligned}
&O\left(s_{\max}\right) + O\left(\sum_{i=1}^{\hat{s}-\sqrt{\hat{s}}}\left(\left\lceil\frac{2\hat{s}}{\hat{s}-i}\right\rceil\right)^2\right) + O\left(\hat{s}\sqrt{\hat{s}}\right)\\
&= O\left(\sum_{i=1}^{\hat{s}-\sqrt{\hat{s}}}\left(\frac{\hat{s}}{\hat{s}-i}\right)^2 + \hat{s} - \sqrt{\hat{s}} + \hat{s}\sqrt{\hat{s}}\right)\\
&= O\left(\sum_{i=\sqrt{\hat{s}}}^{\hat{s}-1}\left(\frac{\hat{s}}{i}\right)^2 + \hat{s}\sqrt{\hat{s}}\right)\\
&= O\left(\hat{s}^2\sum_{i=\sqrt{\hat{s}}}^{\hat{s}-1}\left(\frac{1}{i}\right)^2 + \hat{s}\sqrt{\hat{s}}\right)\\
&= O\left(\hat{s}^2\frac{1}{\sqrt{\hat{s}}} + \hat{s}\sqrt{\hat{s}}\right)\\
&= O\left(\hat{s}^{3/2}\right)\\
&= O\left(s_{\max}^{3/2}\right).
\end{aligned}
$$

□

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

4.1 Outline

Let $P = v_0, e_1, v_1, \ldots, v_\ell$ 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 $\theta = (i-1) \bmod 2$ 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 $\theta = (i-1) \bmod 2$, 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

Algorithm 2 Construction of the augmenting path $\mathrm{CAP}((G, M), f, g, \ell)$.

Require: The path P_0 is an augmenting path with length ℓ from f to g.

$P_0, P_1, \ldots, P_\ell$: initially $\emptyset$.
target $= g$
for $i = 1; i \leq \ell; i++$ **do**
  if i is even **then**
    target chooses the node $v_{\ell-i}$ that satisfies $\mathbf{I}_M((\text{target}, v_{\ell-i})) = 1$.
    $P_{\ell-i} \leftarrow P_{\ell-i+1} \cup \{(\text{target}, v_{\ell-i})\}$.
    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 a node $v_{\ell-i}$ that satisfies $\mathbf{I}_M((\text{target}, v_{\ell-i})) = 0$ and $r^0_{H_{\ell-i+1}}(f, v_{\ell-i}) = \ell - i$.
    $P_{\ell-i} \leftarrow P_{\ell-i+1} \cup \{(\text{target}, v_{\ell-i})\}$.
    target $\leftarrow v_{\ell-i}$.

concatenated walk $Q \circ (v_i, v) \circ P_i$ is guaranteed to be simple.

4.2 Algorithm Details

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 $v_{\ell-i+1}$. Assume that the algorithm has already found $P_{\ell-i+1}$ at the beginning of the i-th step. Any node in $V(P_{\ell-i+1}) \setminus \{v_{\ell-i+1}\}$ 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 $(v_{\ell-i}, v_{\ell-i+1})$ must be a matching edge, and thus, the algorithm picks as predecessor of v_{l-i+1} the node at the other end of the matched edge that v_{l-i+1} is a part of. 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 picks an arbitrary neighbor v of $v_{\ell-i+1}$ satisfying $r^0_{G-P_{\ell-i+1}}(f, v) = \ell - i$ and $\mathbf{I}_M(v, v_{\ell-i+1}) = 0$ as the predecessor of $v_{\ell-i+1}$.

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

Proof : Let $z_0 = g$ and z_i be the node that satisfies target $= z_i$ at the end of the i-th iteration for $1 \leq i \leq \ell$. Let H_i be a subgraph induced by $V(G - P_i)$. We prove the statement that $P_{\ell-h}$ is a (h mod 2)-alternating path between z_0 and z_h. As $r^0_G(f, z_\ell) \leq r^0_{H_1}(f, z_\ell) = 0$, $z_\ell = v$ holds, and thus, we obtain P_0 as an augmenting path of length ℓ from f to g by setting $h = \ell$. The proof follows the induction on h. (Basis) Since z_0 chooses the node z_1 that satisfies $\mathbf{I}_M((\text{target}, v_{\ell-1})) = 0$ and $r^0_{H_\ell}(f, v_{\ell-1}) = \ell - 1$ in the first iteration of Algorithm 2, $P_{\ell-1} = \{(z_0, z_1)\}$ is a 1-alternating path between z_0 and z_1. (Inductive Step) As the induction hypothesis, suppose there exists a (h' mod 2)-alternating path between z_0 and $z_{h'}$ at the end of the h'-th iteration. Because $r^{h' \bmod 2}_{H_{\ell-h'+1}}(f, z_{h'}) = \ell - h'$ holds by the definition of $z_{h'}$, there exists an edge $(z_{h'}, v)$ that satisfies $\mathbf{I}_M((z_{h'}, v)) = h'$ mod 2, and $r^{(h'+1) \bmod 2}_{H_{\ell-h'}}(f, v) = \ell - h' - 1$ holds. Therefore, $z_{h'}$ can choose the node $z_{h'+1}$ that satisfies $\mathbf{I}_M((\text{target}, z_{h'+1})) = h'$ mod 2 and $r^0_{H_{\ell-h'}}(f, z_{h'+1}) = \ell - h' - 1$ in the $(h' + 1)$-th iteration of Algorithm 2. Hence, $P^{\ell-h'} \circ \{(z_{h'}, z_{h'+1})\}$ is a $((h + 1)$ mod 2)-alternating path between z_0 and $z_{h'+1}$ at the end of the $(h' + 1)$-th iteration.

We show the running-time analysis of Algorithm 2. The algorithm consists of ℓ iterations. As each iteration is obviously implemented in $O(\ell)$ rounds, the running time of Algorithm 2 is $O(\ell^2)$ rounds. □

Theorem 1 trivially follows from Lemma 4.

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

5.1 Outline

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 $r^\theta_H(f, v)$ 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 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

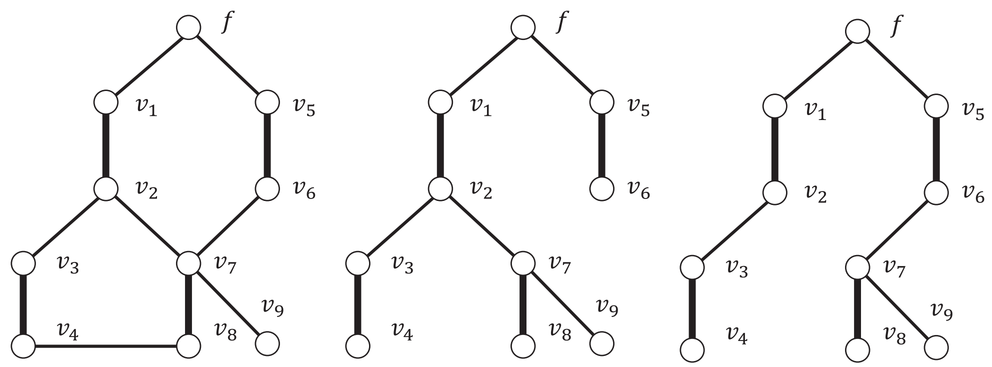

Fig. 1 Examples of the alternating base tree. Bold lines are matching edges, and thin lines are unmatched edges.

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 a 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, $r_G^{\gamma_G(v)}(f, v) = r_G^{1-\gamma_G(v)}(f, \mathsf{par}_T(v)) + 1$ and $\mathbf{I}_M((v, \mathsf{par}_{T_I}(v))) = 1 - \gamma_G(v)$ hold for any $v \in V(G) \setminus \{f\}$.

It is not difficult to check that such a spanning tree always exists. As a node might have two or more shortest alternating paths, T is not uniquely determined (see Fig. 1 (1) and (2) for examples). In the following argument, however, we fix an arbitrarily chosen alternating base tree T. It should be emphasized that the alternating base tree does not necessarily contain an alternating path from f to each node v. For example, both alternating base trees in Fig. 1 have no alternating path from f to v_9.

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. We define the *outgoing edges* of T_v as the set of edges whose one of endpoint belongs to $T(v)$ and the other endpoint does not belong to T_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 $F_k = \{(u, v) \mid (u, v) \in E(G) \setminus M \wedge \max(r_G^0(f, u), r_G^0(f, v)) = k\} \cup \{(u, v) \mid (u, v) \in M \wedge \max(r_G^1(f, u), r_G^1(f, v)) = k\}$. 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., all the edges forming a cut of size one) in G_k. Note that B_k 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 $F_k^c \subseteq F_k \setminus E(T)$ be an arbitrary subset of non-tree edges in F_k satisfying $B_{k-1} \setminus B_k \subseteq \cup_{e \in F_k^c} E(\mathsf{cyc}(e))$. Then, $(T + \cup_{1 \leq i \leq k} F_i^c) > G_k$ holds.

Lemma 5 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). Because $G_k \subseteq G_{k+1}$ holds for any $0 \leq k \leq n-1$, we have $B_{k+1} \subset B_k$, which implies that $B_k \setminus B_{k+1}$ for all k are mutually disjoint. Then, $\sum_{0 \leq i \leq n-1} |B_k \setminus B_{k+1}| = |B_0| = n - 1$ holds. Since at most one edge is augmented for each edge in $B_k \setminus B_{k+1}$, the size $|\cup_{0 \leq i \leq n-1} F_i^c|$ is bounded by $n - 1$. Since $\mathsf{cyc}(e)$ obviously covers $\mathsf{ep}(v)$, the constructed edge set F_k^c satisfies the lemma. Consequently, $H = T + \cup_{1 \leq i \leq n} F_i^c > G_n$ is satisfied, and thus, H is a sparse certificate.

The idea behind our algorithm is the seminal blossom argument by Edmonds [31]. A walk $W = v_0, e_1, v_1, e_2, \ldots, v_\ell$ is called an *odd (even) alternating cycle* if it satisfies the following condition:

- $\mathbf{I}_M(e_i) + \mathbf{I}_M(e_{i+1}) = 1$ holds for any $1 \leq i \leq \ell - 1$.
- $v_0 = v_\ell$ holds.
- The length of the walk W is odd (even).

If an odd alternating cycle has no consecutive proper subsequence forming an odd alternating cycle, it is called *minimal*[†]. Note that a minimal odd alternating cycle can still have a consecutive subsequence forming an even alternating cycle. The node that is first and last node of odd alternating cycle is called *stem node*. An odd alternating cycle C is said to be *reachable* from an unmatched node x if either the stem node is x or there exists a node v in C admitting an even alternating path from x to v not intersecting C. A node u is called *x-covered* if there exists a minimal odd alternating

[†]Due to some technical reason, we allow W to be non-simple, but this modification does not affect the correctness of the original argument by Edmonds.

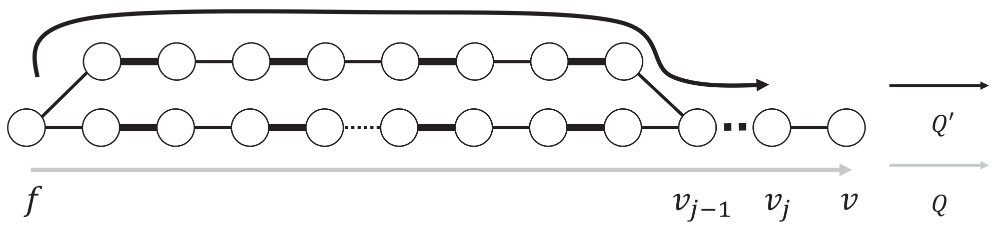

Fig. 2 Proof of Lemma 6 for (Case 2b). Bold lines are matching edges, and thin lines are unmatched edges. The dotted line is the edge included in G but not in G_k. Note that the edge (v_{j-1}, v_j) is actually included in G_k, but it is drawn with a dotted line for explaining the contradiction.

cycle C reachable from x such that C contains u as a non-stem node. It is known that the vertex v is bireachable in G if and only if v is f-covered in G [31]. Our algorithm adds the edges not in T incrementally with guaranteeing the invariant that v is f-covered in $(T + \cup_{1\leq i\leq k} F^c_i)$ if and only if v is included in a 2-edge connected component of size at least two in $T + \cup_{1\leq i\leq k} F^c_i$. The addition of edges from $B_{k-1} \setminus B_k$ in our algorithm can be seen as the process of f-covering the vertex v which is bireachable in G_{k+1} but not in any 2-edge connected component of $(T + \cup_{1\leq i\leq k} F^c_i)$, by creating new minimal odd alternating cycles reachable from f.

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^c_k to start the computation of F^c_{k+1}. As the information on $r^\theta_G(f, v)$ for $\theta \in \{0, 1\}$ is available to v, each node can identify the level of each incident edge. Thus, the construction of F^c_k for all k can be executed in parallel. The details of the distributed construction is explained in Sect. 5.3.

5.2 Proof Details

Before proving Lemma 5, we prove an auxiliary lemma.

Lemma 6: For any $\theta \in \{0, 1\}$ and $v \in V(G) \setminus \{f\}$ such that $r^\theta_G(f, v) \leq k+1$ holds, $r^\theta_{G_{k'}}(f, v) = r^\theta_G(f, v)$ holds for all $k' \geq k$.

Proof : The proof is based on induction on k. (Basis) $k = 0$: Let v be any node satisfying $r^\theta_G(f, v) \leq k + 1$ for some $\theta \in \{0, 1\}$, and let Q be the θ-shortest path from f to v in G. This path is contained in T because v chooses f as its parent in T. (Inductive Step): As the induction hypothesis, suppose $r^\theta_{G_{k-1}}(f, u) = r^\theta_G(f, u)$ holds (and also $r^\theta_{G_k}(f, u) = r^\theta_G(f, u)$ holds because of $G_{k-1} \subseteq G_k$) for any u and θ satisfying $r^\theta_G(f, u) \leq k$. Consider any node v such that $r^\theta_G(f, v) \leq k + 1$ holds. As the case of $r^\theta_G(f, v) < k + 1$ is evidently proved by the induction hypothesis, we assume $r^\theta_G(f, v) = k + 1$. The proof consists of the following two cases depending on whether the shortest alternating path from f to v is θ-alternating path or not.

(Case 1) $\gamma_G(v) = \theta$: By the definition of alternating base trees, we have $r^{1-\theta}_G(f, \mathsf{par}(v)) = r^\theta_G(f, v) - 1 = k$. In addition, for any $w \in T$, $r^{\gamma_G(w)}_G(f, w) = r^{1-\gamma_G(w)}_G(f, \mathsf{par}(w)) + 1 > r^{\gamma_G(\mathsf{par}(w))}_G(f, \mathsf{par}(w))$ holds. Therefore any node $w \in T_v$ satisfies $r^{\gamma_G(w)}_G(f, w) \geq k + 1$. Then, any outgoing non-tree edge of T_v has a level of at least $k + 1$. That is, $\mathsf{ep}(v)$ is a bridge in G_k. Since $r^{1-\theta}_G(f, \mathsf{par}(v)) = k$ holds, the induction hypothesis yields $r^{1-\theta}_{G_k}(f, \mathsf{par}(v)) = k$ and thus there exists a $(1 - \theta)$-alternating path P from f to $\mathsf{par}(v)$ in G_k. Due to the fact that $\mathsf{ep}(e)$ is a bridge, P does not contain v. Hence the concatenated path $P \circ \mathsf{ep}(e)$ is a θ-alternating path from f to v in G_k of length $k + 1$. That is, $r^\theta_{G_k}(f, v) = r^\theta_G(f, v)$ holds.

(Case 2) $\gamma_G(v) = 1 - \theta$: Let $Q = v_0, e_1, v_1, e_2, \ldots, e_{k+1}, v_{k+1}$ be the shortest θ-alternating path from f to v in G ($f = v_0$ and $v = v_{k+1}$). To prove the lemma, it suffices to show that any edge in Q has a level of at most k or is an edge in $E(T)$. Suppose for contradiction that a non-tree edge e_j has the level $k' > k$. Without loss of generality, we assume that j is the highest value for which this condition is satisfied. That is, any edge $e_{j'}$ for $j' > j$ has a level at most k or is an edge in T. We define ρ as $\mathbf{I}_M(e_j)$. We further divide Case 2 into the following three subcases depending on whether e_j is the last edge, and otherwise whether it's a matching edge or not.

(Case 2a) $j = k + 1$: Since Q is the shortest θ-alternating path of length $k + 1$, $\rho = 1 - \theta$ holds, and $Q^p_{v_k}$ is a $(1 - \theta)$-alternating path from f to v_k of length k. From the condition $\gamma_G(v) = \gamma_G(v_{k+1}) = 1 - \theta$ for Case 2, $r^{1-\theta}_G(f, v_k) \leq k$ and $r^{1-\theta}_G(f, v_{k+1}) < r^\theta_G(f, v_{k+1}) = k + 1$ hold. That is, the level of $e_j = e_{k+1}$ is at most k, which is a contradiction.

(Case 2b) $j < k + 1$ and $\rho = 1$: Since the length of $Q^p_{v_j}$ is j, we have $r^0_G(f, v_j) \leq j \leq k$. From the induction hypothesis, G_{k-1} contains a 0-alternating path Q' from f to v_j. In other words, v_j has a 0-alternating path Q' such that any non-tree edge in $E(Q')$ has a level of at most k. The assumption of $\rho = 1$ implies that Q' must terminate with a matching edge incident to v_j, i.e., the edge e_j. This is a contradiction because we assume that e_j is not contained in G_{k-1}.

(Case 2c) $j < k + 1$ and $\rho = 0$: We denote $R = Q^s_{v_j}$ as shorthand. As the length of Q^p_j is $j \leq k$, from the induction hypothesis, we have $r^1_{G_k}(f, v_j) = r^1_G(f, v_j) \leq j$, and thus, G_k contains a 1-alternating path P from f to v_j of shortest length. Let $v_h \in V(R) \cap V(P)$ be the first node in P, which also belongs to R. If e_{h+1} is a matched edge, $P^p_{v_h} \circ Q^s_{v_h}$ is a θ-alternating path in G_k (see Fig. 3 (a)), the length of which is bounded by $|P^p_{v_h} \circ Q^s_{v_h}| \leq |P| + (k + 1 - h) \leq j + (k + 1 - j) \leq k + 1$. Hence, we obtain $r^\theta_{G_k}(f, v_{k+1}) \leq k + 1 = r^\theta_G(f, v_{k+1})$, which is a contradiction. If e_{h+1} is an unmatched edge, e_h is a matched edge. Therefore, $P^p_{v_h} \circ \overline{R^p_{v_h}}$ is a 0-alternating path from f to v_j in G_k (see Fig. 3 (b)). Since we consider the case of $\rho = 0$, the edge e_j is an unmatched edge. Therefore, $v_h \neq v_j$ holds, and thus v_h is not the last node of P. This

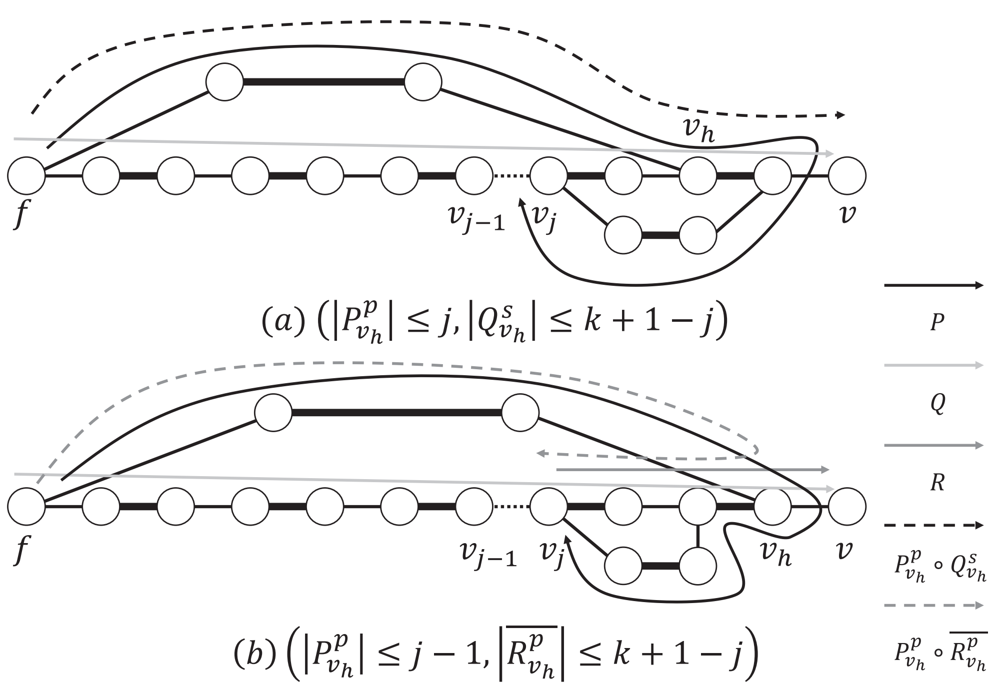

Fig. 3 Proof of Lemma 6 of (Case 2c). Bold lines are matching edges, and thin lines are unmatched edges. The dotted line is the edge included in G but not in G_k.

implies $|P^p_{v_h}| \le j-1$. We obtain $|P^p_{v_h} \circ \overline{R^p_{v_h}}| \le j-1+(k+1-h) \le j-1+(k+1-j) \le k$, and thus, $r^0_G(f, v_j) \le r^0_{G_k}(f, v_j) \le k$. Since $Q^s_{v_{j-1}}$ is a 0-alternating path from f to v_{j-1} of length $j-1$, we have $r^0_G(f, v_{j-1}) \le j-1 \le k$. This implies that the level of e_j is at most k, which is a contradiction. □

Now, we present the proof of Lemma 5.

Proof : Let $F^c_{\le k} = \cup_{1 \le i \le k} F^c_i$ and $H_k = T + F^c_{\le k}$. We prove the lemma inductively. For $k = 0$, $H_0 = T \succ G_0 = T$ evidently holds. Thus, it suffices to show $H_k \succ G_k$, assuming $H_{k'} \succ G_{k'}$ for all $0 \le k' < k$. For any $0 \le h \le n$, we define $U_h = \{(v, \theta) \mid v \in V(G) \wedge r^\theta_{G_k}(f, v) = h\}$. If v is θ-reachable in H_k for all $0 \le h \le n$ and $(v, \theta) \in U_h$, we can conclude that $H_k \succ G_k$. The proof of this statement follows the (nested) induction on h. (Basis) As U_0 contains only $(f, 1)$, the statement evidently holds. (Inductive Step) As the induction hypothesis, suppose v is θ-reachable for any $(v, \theta) \in \cup_{0 \le i \le h} U_i$, and consider any pair (v, θ) in U_{h+1}. Then, we consider the following two cases.

(Case 1) $\mathsf{ep}(v)$ is a bridge in G_k: We have $r^{1-\theta}_G(f, \mathsf{par}(v)) = h$ from the definition of alternating base trees. Since the induction hypothesis guarantees that $\mathsf{par}(v)$ preserves the reachability of G_k in H_k, there exists a $(1-\theta)$-alternating path P from f to $\mathsf{par}(v)$ in H_k. In addition, P does not contain $\mathsf{ep}(e)$, because $\mathsf{ep}(e)$ is a bridge in $H_k \subseteq G_k$. From $\mathbf{I}_M(\mathsf{ep}(v)) = 1-\theta$, which directly follows from the definition of alternating base trees, the concatenated path $P \circ \mathsf{ep}(v)$ becomes a θ-alternating path from f to v in H_k (see Fig. 4 (1)). Then, v is θ-reachable in H_k.

(Case 2) $\mathsf{ep}(v)$ is not a bridge in G_k: As $G_0 \subseteq G_1 \subseteq \ldots, \subseteq G_k$ holds, there exists $1 \le j \le k$ such that $\mathsf{ep}(v) \in B_{j-1} \setminus B_j$ holds. Then, F^c_j contains an outgoing edge e of T_v belonging to F_j. Let $e = (u, w)$ and u be the side contained in T_v. We assume that e is not a matching edge. By symmetry, the case of $e \in M$ is proved similarly. From the definition of F_j, we have $\max\{r^1_G(f, u), r^1_G(f, w)\} = j \le k$. Lemma 6 implies that both u and w have 1-alternating paths from f in G_{j-1}; from the induction hypothesis $H_{j-1} \succ G_{j-1}$, they have 1-alternating paths from f also in H_{j-1}, which we refer to as P and Q, respectively. Since $\mathsf{ep}(v)$ is a bridge of $G_{j-1} \supseteq H_{j-1}$, the suffix P^s_v is a subgraph of T_v. In addition, Q does not intersect $V(T_v)$, because both f and w are outside T_v. Thus, P^s_v and Q are mutually disjoint, and the concatenated path $Q' = Q \circ (w, u) \circ \overline{P^s_v}$ is simple. It is easy to check that Q' is an alternating path from f to v. As Q, e, and $\overline{P^s_v}$ are all contained in $H_{j-1} + F^c_j = H_j$, P^p_v and Q' are contained in H_j (see Fig. 4 (2)). The alternating paths P^s_v and Q' have different parities because their last edges are adjacent in P. Hence, we conclude that v is bireachable in H_j. □

5.3 Distributed Implementation

This section explains how to implement the centralized sparse certificate algorithm, presented in Sect. 5.1, 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 $r^1_G(f, v)$ and $r^0_G(f, v)$; 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

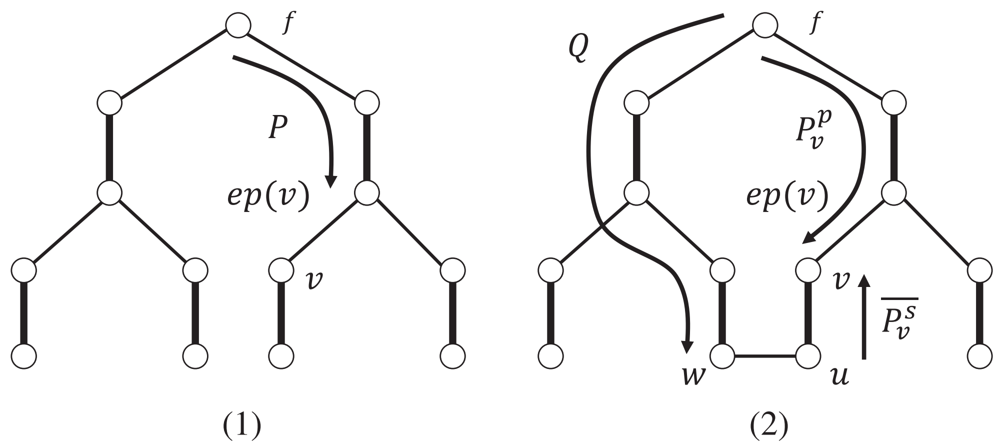

Fig. 4 Proof of Lemma 5. Bold lines are matching edges, and thin lines are unmatched edges.

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 $v_i \neq f$ **then**
  choose edge (u, v_i) that is incident on the vertex v_i and satisfies $r_G^{\gamma(v_i)}(f, v_i) = r_G^{1-\gamma(v_i)}(f, u) - 1$ and $\mathbf{I}((u, v_i)) = 1 - \gamma(v_i)$ (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 $\perp$.

algorithm, which is implemented in zero round.

The main idea of constructing the edge set $F^c = \cup_{1 \leq i \leq n} F_i^c$ 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 $\leq_{\mathsf{lca}}$ over all non-tree edges as $e_1 \leq_{\mathsf{lca}} e_2$ if and only if $\mathsf{lca}(e_1) \leq \mathsf{lca}(e_2)$. 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 1. The node v receiving message i decides $d_v = i$ and sends the value $i + 1$ to its children.
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 $\leq_{\mathsf{lca}}$. This task is implemented through a standard aggregation over T. Each leaf node v sends the minimum edge e in $F_k \cap E^*(T_v)$ together with $\mathsf{lca}(e)$. If $F_k \cap E^*(T_v) = \emptyset$ holds, the leaf sends a dummy edge e such that $\mathsf{lca}(e) = \infty$ holds. 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 $\leq_{\mathsf{lca}}$ and sends the chosen edge e_v and $\mathsf{lca}(e_v)$ to $\mathsf{par}(v)$. Finally, v outputs e_v if $\mathsf{lca}(e_v) < d_v$ holds or $\perp$ otherwise. The correctness of $\mathsf{ConstF}(k)$ follows the proposition below.

Proposition 3: Let e be the minimum edge in $E^*(T_v)$ with respect to $\leq_{\mathsf{lca}}$. Then, e is an outgoing edge of T_v if and only if $\mathsf{lca}(e) < d_v$ holds (thus, $\mathsf{ep}(v)$ is a bridge if $\mathsf{lca}(e) \geq d_v$ holds).

The edge set F^c is constructed by running $\mathsf{ConstF}(k)$ for all $1 \leq k \leq n$. 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 $1 \leq k \leq n$, 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. Following Lemma 6, each node v can decide the edge e that should be added to $F^c = \cup_{1 \leq i \leq n} F^c_i$.

6. Conclusion

We proposed a randomized $O(s_{\max}^{3/2})$-rounds (i.e. $O(n^{3/2})$-rounds) algorithm for computing a maximum matching in the CONGEST model, which is the first one 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.

Acknowledgements

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

References

[1] S. Kutten and D. Peleg, "Fast distributed construction of small k-dominating sets and applications," Journal of Algorithms, vol.28, no.1, pp.40–66, 1998.

[2] N. Kitamura, H. Kitagawa, Y. Otachi, and T. Izumi, "Low-congestion shortcut and graph parameters," Proc. 33rd International Symposium on Distributed Computing (DISC), pp.25:1–25:17, 2019.

[3] M. Ghaffari and B. Haeupler, "Distributed algorithms for planar networks II: Low-congestion shortcuts, MST, and min-cut," Proc. Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp.202–219, 2016.

[4] T. Jurdziński and K. Nowicki, "MST in $O(1)$ rounds of congested clique," Proc. Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp.2620–2632, 2018.

[5] M. Ghaffari and J. Li, "New distributed algorithms in almost mixing time via transformations from parallel algorithms," Proc. 32nd International Symposium on Distributed Computing (DISC), pp.31:1–31:16, 2018.

[6] B. Haeupler, T. Izumi, and G. Zuzic, "Near-optimal low-congestion shortcuts on bounded parameter graphs," Proc. 30th International Symposium on Distributed Computing (DISC), Lecture Notes in Computer Science, vol.9888, pp.158–172, Springer, Berlin, Heidelberg, 2016.

[7] B. Haeupler and J. Li, "Faster distributed shortest path approximations via shortcuts," 32nd International Symposium on Distributed Computing (DISC), pp.33:1–33:14, 2018.

[8] C. Lenzen and D. Peleg, "Efficient distributed source detection with limited bandwidth," Proc. 2013 ACM Symposium on Principles of Distributed Computing (PODC), pp.375–382, 2013.

[9] S. Holzer and R. Wattenhofer, "Optimal distributed all pairs shortest paths and applications," Proc. 2012 ACM Symposium on Principles of Distributed Computing (PODC), pp.355–364, 2012.

[10] D. Nanongkai, "Distributed approximation algorithms for weighted shortest paths," Proc. 46th Annual ACM Symposium on Theory of Computing (STOC), pp.565–573, 2014.

[11] A. Bernstein and D. Nanongkai, "Distributed exact weighted all-pairs shortest paths in near-linear time," Proc. 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC), p.334–342, 2019.

[12] M. Ghaffari and J. Li, "Improved distributed algorithms for exact shortest paths," Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC), pp.431–444, 2018.

[13] S. Forster and D. Nanongkai, "A faster distributed single-source shortest paths algorithm," 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS), pp.686–697, 2018.

[14] M. Ghaffari, A. Karrenbauer, F. Kuhn, C. Lenzen, and B. Patt-Shamir, "Near-optimal distributed maximum flow," 2015 ACM Symposium on Principles of Distributed Computing (PODC), pp.81–90, 2015.

[15] M. Dory, Y. Efron, S. Mukhopadhyay, and D. Nanongkai, "Distributed weighted min-cut in nearly-optimal time," arXiv preprint arXiv:2004.09129, 2020.

[16] M. Ghaffari and F. Kuhn, "Distributed minimum cut approximation," International Symposium on Distributed Computing (DISC), Lecture Notes in Computer Science, vol.8205, pp.1–15, Springer, Berlin, Heidelberg, 2013.

[17] D. Nanongkai and H.-H. Su, "Almost-tight distributed minimum cut algorithms," International Symposium on Distributed Computing (DISC), Lecture Notes in Computer Science, vol.8784, pp.439–453, Springer, Berlin, Heidelberg, 2014.

[18] M. Daga, M. Henzinger, D. Nanongkai, and T. Saranurak, "Distributed edge connectivity in sublinear time," arXiv preprint arXiv:1904.04341, 2019.

[19] S. Frischknecht, S. Holzer, and R. Wattenhofer, "Networks cannot compute their diameter in sublinear time," Proc. Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp.1150–1162, 2012.

[20] N. Bacrach, K. Censor-Hillel, M. Dory, Y. Efron, D. Leitersdorf, and A. Paz, "Hardness of distributed optimization," 2019 ACM Symposium on Principles of Distributed Computing (PODC), pp.238–247, 2019.

[21] A.D. Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer, "Distributed verification and hardness of distributed approximation," Proc. 43rd An-

nual ACM SIGACT Symposium on Theory of Computing (STOC), pp.363–372, 2011.
[22] R. Ben-Basat, K. Kawarabayashi, and G. Schwartzman, "Parameterized distributed algorithms," 33rd International Symposium on Distributed Computing (DISC), pp.6:1–6:16, 2018.
[23] M. Ahmadi, F. Kuhn, and R. Oshman, "Distributed approximate maximum matching in the CONGEST model," 32nd International Symposium on Distributed Computing (DISC), pp.6:1–6:17, 2018.
[24] J.E. Hopcroft and R.M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," SIAM Journal on Computing, vol.2, no.4, pp.225–231, 1973.
[25] M. Ahmadi and F. Kuhn, "Distributed maximum matching verification in CONGEST," 34th International Symposium on Distributed Computing (DISC), pp.37:1–37:18, 2020.
[26] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "The price of being near-sighted," 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp.980–989, 2006.
[27] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Local computation: Lower and upper bounds," Journal of the ACM (JACM), vol.63, no.2, Article No.17, pp.1–44, 2016.
[28] R. Bar-Yehuda, K. Censor-Hillel, M. Ghaffari, and G. Schwartzman, "Distributed approximation of maximum independent set and maximum matching," 36th Annual ACM Symposium on Principles of Distributed Computing (PODC), pp.165–174, 2017.
[29] Z. Lotker, B. Patt-Shamir, and S. Pettie, "Improved distributed approximate matching," Journal of the ACM (JACM), vol.62, no.5, Article No.38, pp.1–17, 2015.
[30] M. Ghaffari, F. Kuhn, and Y. Maus, "On the complexity of local distributed graph problems," 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC), pp.784–797, 2017.
[31] J. Edmonds, "Paths, trees, and flowers," Canadian Journal of Mathematics, vol.17, pp.449–467, 1965.
[32] J. Edmonds, "Maximum matching and a polyhedron with 0,1-vertices," Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics, pp.125–130, 1965.
[33] N. Blum, "A new approach to maximum matching in general graphs," International Colloquium on Automata, Languages, and Programming (ICALP), Lecture Notes in Computer Science, vol.443, pp.586–597, Springer, 1990.
[34] V.V. Vazirani, "A proof of the MV matching algorithm," arXiv preprint arXiv:2012.03582, 2020.
[35] H.N. Gabow and R.E. Tarjan, "Faster scaling algorithms for general graph matching problems," Journal of the ACM (JACM), vol.38, no.4, pp.815–853, 1991.

Naoki Kitamura received the B.S. and M.S. degrees in Computer Science from Nagoya Institute of Technology in 2017 and 2019, respectively. He is now Ph.D. student in Computer Science from Nagoya Institute of Technology.

Taisuke Izumi received the M.E. and D.I. degrees in computer science from Osaka University, Japan, in 2003 and 2006, respectively. He worked as an assistant professor at Nagoya Institute of Technology, Japan, during 2006–2009, and worked as an assistant professor during 2009–2020. He is currently an associate professor at the Graduate School of Information Science and Technology, Osaka University, Japan. His research interests include algorithms and distributed systems. He is a member of IEICE, IPSJ, and ACM.