

Distributed Maximum Matching in Bounded Degree Graphs

Guy Even*

Moti Medina*[†]

Dana Ron*[‡]

August 27, 2018

Abstract

We present deterministic distributed algorithms for computing approximate maximum cardinality matchings and approximate maximum weight matchings. Our algorithm for the unweighted case computes a matching whose size is at least $(1 - \varepsilon)$ times the optimal in $\Delta^{O(1/\varepsilon)} + O\left(\frac{1}{\varepsilon^2}\right) \cdot \log^*(n)$ rounds where n is the number of vertices in the graph and Δ is the maximum degree. Our algorithm for the edge-weighted case computes a matching whose weight is at least $(1 - \varepsilon)$ times the optimal in $\log(\min\{1/w_{\min}, n/\varepsilon\})^{O(1/\varepsilon)} \cdot (\Delta^{O(1/\varepsilon)} + \log^*(n))$ rounds for edge-weights in $[w_{\min}, 1]$.

The best previous algorithms for both the unweighted case and the weighted case are by Lotker, Patt-Shamir, and Pettie (SPAA 2008). For the unweighted case they give a randomized $(1 - \varepsilon)$ -approximation algorithm that runs in $O((\log(n))/\varepsilon^3)$ rounds. For the weighted case they give a randomized $(1/2 - \varepsilon)$ -approximation algorithm that runs in $O(\log(\varepsilon^{-1}) \cdot \log(n))$ rounds. Hence, our results improve on the previous ones when the parameters Δ , ε and w_{\min} are constants (where we reduce the number of runs from $O(\log(n))$ to $O(\log^*(n))$), and more generally when Δ , $1/\varepsilon$ and $1/w_{\min}$ are sufficiently slowly increasing functions of n . Moreover, our algorithms are deterministic rather than randomized.

Keywords. Centralized Local Algorithms, Distributed Local Algorithms, Maximum Matching, Maximum Weighted Matching, Graph Algorithms.

1 Introduction

In this work we consider the problem of distributively computing an approximate maximum (weighted) matching in bounded degree graphs. Let $G = (V, E)$ denote an edge weighted graph with n vertices and maximum degree Δ . Assume that the maximum edge weight is 1 and let w_{\min} denote the minimum edge weight. Denoting by $\text{MCM}(G)$ the maximum cardinality of a matching in G and by $\text{MWM}(G)$ the maximum weight of a matching in G , we present the following results:

*School of Electrical Engineering, Tel-Aviv Univ., Tel-Aviv 69978, Israel.
{guy, medinamo, danar}@eng.tau.ac.il.

[†]M.M was partially funded by the Israeli Ministry of Science and Technology.

[‡]Research supported by the Israel Science Foundation grant number 671/13.

- A deterministic distributed algorithm that for any $\varepsilon \in (0, 1)$ computes a matching whose size is at least $(1 - \varepsilon) \cdot \text{MCM}(G)$ in

$$\Delta^{O(1/\varepsilon)} + O\left(\frac{1}{\varepsilon^2}\right) \cdot \log^*(n)$$

rounds.

- A deterministic distributed algorithm that and for any $\varepsilon \in (0, 1)$ computes a matching whose weight is at least $(1 - \varepsilon) \cdot \text{MWM}(G)$ in

$$\log(\min\{1/w_{\min}, n/\varepsilon\})^{O(1/\varepsilon)} \cdot (\Delta^{O(1/\varepsilon)} + \log^*(n))$$

rounds.

The best previous algorithms for both the unweighted and weighted cases are by Lotker, Patt-Shamir, and Pettie [LPSP08]. For the unweighted case they give a randomized $(1 - \varepsilon)$ -approximation algorithm that runs in $O((\log(n))/\varepsilon^3)$ rounds with high probability¹ (w.h.p). Hence we get an improved result when $\Delta^{O(1/\varepsilon)} = o(\log(n))$. In particular, for constant Δ and ε , the number of rounds is $O(\log^*(n))$. Note that an $O(1)$ -approximation of a maximum matching in an n -node ring cannot be computed by any deterministic distributed algorithm in $o(\log^*(n))$ rounds [CHW08, LW08]. For the weighted case, they give a randomized $(1/2 - \varepsilon)$ -approximation algorithm that runs in $O(\log(\varepsilon^{-1}) \cdot \log(n))$ rounds (w.h.p)². Our MWM approximation algorithm runs in significantly fewer rounds for various settings of the parameters Δ , $1/\varepsilon$, and $1/w_{\min}$. In particular, when they are constants, the number of rounds is $O(\log^*(n))$.

Previous work				Here (Deterministic)	
problem	# rounds	success prob.	apx. ratio.	# rounds	apx. ratio.
MCM	$O(\frac{\log(n)}{\varepsilon^3})$	$1 - \frac{1}{\text{poly}(n)}$	$1 - \varepsilon$ [LPSP08]	$\Delta^{O(\frac{1}{\varepsilon})} + O(\frac{1}{\varepsilon^2}) \cdot \log^*(n)$	$1 - \varepsilon$ [Thm. 8]
	$\Delta^{O(\frac{1}{\varepsilon})}$	$1 - \Theta(1)$	$1 - \varepsilon$ [NO08]		
MWM	$O(\log(\varepsilon^{-1}) \cdot \log(n))$	$1 - \frac{1}{\text{poly}(n)}$	$1/2 - \varepsilon$ [LPSP08]	$\log^{O(\frac{1}{\varepsilon})}(\Gamma) \cdot (\Delta^{O(\frac{1}{\varepsilon})} + \log^*(n))$	$1 - \varepsilon$ [Thm. 12]
	$O(\frac{\log^4(n)}{\varepsilon} \cdot \log(\Gamma))$	deterministic	$1/6 - \varepsilon$ [PS10]		

Table 1: A comparison between MCM and MWM DISTLOCAL algorithms. The ratio between the maximum to minimum edge weight is denoted by Γ (we may assume that $\Gamma \leq n/\varepsilon$).

1.1 Techniques

1.1.1 Centralized local computation algorithms

For both the unweighted and the weighted versions of the problem we design (deterministic) *Centralized Local Computation Algorithms*, which translate into distributed algorithms. Centralized local computation (CENTLOCAL) algorithms, as defined by Rubinfeld et al. [RTVX11],

¹We say that an event occurs with high probability if it occurs with probability at least $1 - \frac{1}{\text{poly}(n)}$.

²Lotker, Patt-Shamir and Pettie remark [LPSP08, Sec. 4] that a $(1 - \varepsilon)$ -MWM can be obtained in $O(\varepsilon^{-4} \log^2 n)$ rounds (using messages of linear size), by adapting the algorithm of Hougardy and Vinkemeyr [HV06] (where details are not provided in the paper).

are algorithms that answer queries regarding (global) solutions to computational problems by performing local (sublinear time) computations on the input. The answers to all queries must be consistent with a single solution regardless of the number of possible solutions. In particular, for the problems we study, each query is an edge e in the graph G , and the algorithm needs to answer whether e belongs to a matching M whose size (or weight) is at least $(1 - \varepsilon)$ times the optimal. Consistency means that all answers to the queries must be according to the same matching M . To this end the algorithm can probe the graph G by asking about the neighbors of vertices of its choice. In this manner the algorithm can obtain the local neighborhood of each queried edge e .

A CENTLOCAL algorithm may be randomized, so that the solution according to which it answers queries may depend on its internal coin flips. However, the solution should not depend on the sequence of the queries (this property is called query order obliviousness [RTVX11]). The main performance measure of CENTLOCAL algorithms is the maximum number of probes performed per query. In this work we will actually be interested in the probe-radius, that is, the maximum distance in the graph of a probe from the queried edge. This translates into the number of rounds performed by the corresponding distributed algorithm.

We believe that using the design methodology of first describing and analyzing a CENTLOCAL algorithm and then transforming it into a distributed local³ (DISTLOCAL) algorithm makes both the presentation and the analysis simpler and easier to follow. The benefit of designing CENTLOCAL algorithms is that it removes the need for coordination between vertices when performing computations on auxiliary graphs (for discussion on these graphs see the following subsection). The transformation from a CENTLOCAL algorithm to a DISTLOCAL one is especially straightforward when the CENTLOCAL algorithm is deterministic (and stateless – Section 2.4).

1.1.2 A Global Algorithm for Approximate Maximum Cardinality Matching

Previous CENTLOCAL and DISTLOCAL-algorithms for finding an approximate maximum cardinality matching are based on the following framework [NO08, LPSP08, MV13]. First consider a global/abstract algorithm whose correctness is based on a result of Hopcroft and Karp [HK73]. The algorithm works iteratively, where in each iteration it constructs a new matching (starting from the empty matching). Each new matching is constructed based on a maximal set of vertex disjoint paths that are augmenting paths with respect to the previous matching. Such a maximal set is a maximal independent set (MIS) in the intersection graph over the augmenting paths. (See Algorithm. 1 for precise details.) The question is how to simulate this global algorithm in a local/distributed fashion, and in particular, how to compute the maximal independent sets over the intersection graphs.

1.1.3 Local Simulation

Our CENTLOCAL algorithm for approximate MCM follows Nguyen and Onak’s [NO08] sub-linear algorithm for approximating the size of a maximum matching (see also [MV13]). The algorithm works recursively, where recursion is applied to determine membership in the previous matching (defined by the global algorithm) as well as membership in an augmenting path that belongs to the maximal set of augmenting paths.

³Strictly speaking, a distributed algorithm is considered *local* if it performs a number of rounds that does not depend on n . Here we allow a weak dependence on n (i.e., $\log^*(n)$ or even $\text{polylog}(n)$).

We differ from [NO08] and [MV13] in the CENTLOCAL MIS algorithm that we apply, which is the algorithm presented in [EMR14]. Recall that the MIS algorithm is applied to intersection graphs over augmenting paths. The MIS algorithm works by computing an acyclic orientation of the edges of the graph, where the radius of the orientation (the longest directed path in the oriented graph) is $\text{poly}(\Delta)$. This in turn is performed by coloring the vertices in $\text{poly}(\Delta)$ colors. An acyclic orientation induces a partial ordering over the vertices, which enables to (locally) simulate the simple greedy sequential algorithm for MIS. The main issue is analyzing the total probe-radius of the resulting combined CENTLOCAL algorithm.

1.1.4 Weighted Matchings

Our CENTLOCAL algorithm for approximate MWM is also based on the abovementioned MIS algorithm as an “inner” building block. As the “outer” building block we use a result described in [Ona10] (and mentioned in [NO08]) for approximating the maximum weight of a matching, which in turn builds on work of Pettie and Sanders [PS04].

1.2 Related Work

We compare our results to previous ones in Table 1. The first line refers to the aforementioned algorithm by Lotker, Patt-Shamir, and Pettie [LPSP08] for the unweighted case. The second line in Table 1 refers to an algorithm of Nguyen and Onak [NO08]. As they observe, their algorithm for approximating the size of a maximum matching in sublinear time can be transformed into a randomized distributed algorithm that succeeds with constant probability (say, $2/3$) and runs in $\Delta^{O(1/\varepsilon)}$ rounds. The third line refers to the aforementioned algorithm by Lotker, Patt-Shamir, and Pettie [LPSP08] for the weighted case. The fourth line refers to the algorithm by Panconesi and Sozio [PS10] for weighted matching. They devise a deterministic distributed $(1/6 - \varepsilon)$ -approximation algorithm that runs in $O\left(\frac{\log^4(n)}{\varepsilon} \cdot \log(\Gamma)\right)$ rounds, where Γ is the ratio between the maximum to minimum edge weight.

We remark that the randomized CENTLOCAL-algorithm by Mansour and Vardi [MV13] for $(1 - \varepsilon)$ -approximate maximum cardinality matching in bounded-degree graphs can be transformed into a randomized DISTLOCAL-algorithm for $(1 - \varepsilon)$ -approximate maximum cardinality matching (whose success probability is $1 - 1/\text{poly}(n)$). Their focus is on bounding the number of probes, which they show is polylogarithmic in n for constant Δ and ε . To the best of our understanding, an analysis of the probe-radius of their algorithm will not imply a DISTLOCAL-algorithm that runs in fewer rounds than the algorithm of Lotker, Patt-Shamir, and Pettie [LPSP08].

2 Preliminaries

2.1 Notations

Let $G = (V, E)$ denote an undirected graph. Let $n(G)$ denote the number of vertices. We denote the degree of v by $\text{deg}_G(v)$. Let $\Delta(G)$ denote the maximum degree, i.e., $\Delta(G) \triangleq \max_{v \in V} \{\text{deg}_G(v)\}$. The length of a path equals the number of edges along the path. We denote the length of a path p by $|p|$. For $u, v \in V$ let $\text{dist}_G(u, v)$ denote the length of the shortest path

between u and v in the graph G . The ball of radius r centered at v in the graph G is defined by

$$B_r^G(v) \triangleq \{u \in V \mid \text{dist}_G(v, u) \leq r\}.$$

If the graph G is clear from the context, we may drop it from the notation, e.g., we simply write n , m , $\text{deg}(v)$, or Δ .

For $k \in \mathbb{N}^+$ and $n > 0$, let $\log^{(k)}(n)$ denote the k th iterated logarithm of n . Note that $\log^{(0)}(n) \triangleq n$ and if $\log^{(i)}(n) = 0$, we define $\log^{(j)}(n) = 0$, for every $j > i$. For $n \geq 1$, define $\log^*(n) \triangleq \min\{i : \log^{(i)}(n) \leq 1\}$.

A subset $I \subseteq V$ is an *independent set* if no two vertices in I are an edge in E . An independent set I is *maximal* if $I \cup \{v\}$ is not an independent set for every $v \in V \setminus I$. We use MIS as an abbreviation of a maximal independent set.

A subset $M \subseteq E$ is a matching if no two edges in M share an endpoint. Let M^* denote a maximum cardinality matching of G . We say that a matching M is a $(1 - \varepsilon)$ -approximate maximum matching if

$$|M| \geq (1 - \varepsilon) \cdot |M^*|.$$

Let $w(e)$ denote the weight of an edge $e \in E$. The weight of a subset $F \subseteq E$ is $\sum_{e \in F} w(e)$ and is denoted by $w(F)$. Let M_w^* denote a maximum weight matching of G . A matching M is a $(1 - \varepsilon)$ -approximate maximum weight matching if $w(M) \geq (1 - \varepsilon) \cdot w(M_w^*)$. We abbreviate the terms maximum cardinality matching and maximum weight matching by MCM and MWM, respectively.

2.2 The DistLocal Model

The model of local distributed computation is a classical model (e.g., [Lin92, Pel00, Suo13]). A distributed computation takes place in an undirected labeled graph $G = (V, E)$. In a labeled graph vertices have unique identifies (IDs). The neighbors of each vertex v are numbered from 1 to $\text{deg}(v)$ in an arbitrary but fixed manner. Ports are used to point to the neighbors of v ; the i th port points to the i th neighbor. Each vertex in the labeled graph models a processor, and communication is possible only between neighboring processors. All processors execute the same algorithm. Initially, every $v \in V$ is input a local input. The computation is done in $r \in \mathbb{N}$ synchronous rounds as follows. In every round: (1) every processor receives a message from each neighbor, (2) every processor performs a computation based on its local input and the messages received from its neighbors, (3) every processor sends a message to each neighbor. We assume that a message sent in the end of round i is received in the beginning of round $i + 1$. After the r th round, every processor computes a local output.

The following assumptions are made in the DISTLOCAL model: (1) The local input to each vertex v includes the ID of v , the degree of the vertex v , the maximum degree Δ , the number of vertices n , and the ports of v to its neighbors. (2) The IDs are distinct and bounded by a polynomial in n . (3) The length of the messages sent in each round is not bounded.

We say that a distributed algorithm is a DISTLOCAL[r]-algorithm if the number of communication rounds is r . Strictly speaking, a distributed algorithm is considered *local* if r is bounded by a constant. We say that a DISTLOCAL[r]-algorithm is *almost local* if $r = O(\log^*(n))$. When it is obvious from the context we refer to an almost DISTLOCAL algorithm simply by a DISTLOCAL algorithm.

2.3 The CentLocal Model

In this section we present the model of centralized local computations that was defined in [RTVX11]. The presentation focuses on problems over labeled graphs (i.e., maximal independent set and maximum matching).

Probes In the CENTLOCAL model, access to the labeled graph is limited to probes. A *probe* is a pair (v, i) that asks “who is the i th neighbor of v ?”. The answer to a probe (v, i) is as follows. (1) If $\deg_G(v) < i$, then the answer is “null”. (2) If $\deg_G(v) \geq i$, then the answer is the (ID of) vertex u that is pointed to by the i th port of v . For simplicity, we assume that the answer also contains the port number j such that v is the j th neighbor of u . (This assumption reduces the number of probes by at most a factor of Δ .)

Online Property of CENTLOCAL-algorithms The set of solutions for problem Π over a labeled graph G is denoted by $\text{sol}(G, \Pi)$. A deterministic CENTLOCAL-algorithm ALG for problem Π over labeled graphs is defined as follows. The input for the algorithm consists of three parts: (1) access to a labeled graph G via probes, (2) the number of vertices n and the maximum degree Δ of the graph G , and (3) a sequence $\{q_i\}_{i=1}^N$ of queries. Each query q_i is a request for an evaluation of $f(q_i)$ where $f \in \text{sol}(G, \Pi)$. Let y_i denote the output of ALG to query q_i . We view algorithm ALG as an online algorithm because it must output y_i without any knowledge of subsequent queries.

A CENTLOCAL-algorithm ALG for Π must satisfy the following condition, called *consistency*,

$$\exists f \in \text{sol}(G, \Pi) \text{ s.t. } \forall N \in \mathbb{N} \forall \{q_i\}_{i=1}^N \forall i : y_i = f(q_i). \quad (1)$$

Resources and Performance Measures The main performance measure is the *maximum number of probes* that the CENTLOCAL-algorithm performs per query. We consider an additional measure called *probe radius*. The probe radius of a CENTLOCAL-algorithm C is r if, for every query q , all the probes that algorithm C performs in G are contained in the ball of radius r centered at q .

Stateless Algorithms A *state* of a CENTLOCAL-algorithm is the maximum number of bits stored between consecutive queries. A CENTLOCAL-algorithm is *stateless* if the algorithm does not store any information between queries. In particular, a stateless algorithm does not store previous queries, answers to previous probes, or answers given to previous queries.⁴ In this paper all our CENTLOCAL-algorithms are stateless.

Example Consider the problem of computing a maximal independent set. The CENTLOCAL-algorithm is input a sequence of queries, each of which is a vertex. The algorithm outputs whether q_i is in I , for some maximal independent set $I \subseteq V$. Consistency in this example means that the algorithm has to satisfy this specification even though it does not probe all of G , and obviously does not store the maximal independent set I . Moreover, a stateless algorithm

⁴We remark that in [RTVX11] no distinction was made between the space needed to answer a query and the space needed to store the state between queries. Our approach is different and follows the DISTLOCAL model in which one does not count the space and running time of the vertices during the execution of the distributed algorithm. Hence, we ignore the space and running time of the CENTLOCAL-algorithm during the processing of a query.

does not even remember the answers it gave to previous queries. Note that if a vertex is queried twice, then the algorithm must return the same answer. Similarly, if two queries are neighbors, then the algorithm may not answer that both are in the independent set. If all vertices are queried, then the answers constitute the maximal independent set I .

2.4 Simulation of CentLocal by DistLocal

Based on an observation made in [PR07] in a slightly different setting, CENTLOCAL-algorithms can simulate DISTLOCAL-algorithms. In this section we consider simulations in the converse direction.

The following definition considers CENTLOCAL-algorithms whose queries are vertices of a graph. The definition can be easily extended to edge queries.

Definition 1. A DISTLOCAL-algorithm D simulates a CENTLOCAL-algorithm C if, for every vertex v , the local output of D in vertex v equals the answer that algorithm C computes for the query v .

The following proposition states that CENTLOCAL-algorithm can be simulated by a DISTLOCAL[r] algorithm provided that the probe radius is r . Message lengths grow at a rate of $O(\Delta^{r+1} \cdot \log n)$ as information (e.g., IDs and existence of edges) is accumulated.

Proposition 1. Every stateless deterministic CENTLOCAL-algorithm C whose probe radius is at most r can be simulated by a deterministic DISTLOCAL[r]-algorithm D .

Proof. The distributed algorithm D collects, for every v , all the information in the ball of radius r centered at v . (This information includes the IDs of the vertices in the ball and the edges between them.)

After this information is collected, the vertex v locally runs the CENTLOCAL-algorithm C with the query v . Because algorithm C is stateless, the vertex has all the information required to answer every probe of C . \square

Proposition 1 suggests a design methodology for distributed algorithms. For example, suppose that we wish to design a distributed algorithm for maximum matching. We begin by designing a CENTLOCAL-algorithm C for computing a maximum matching. Let r denote the probe radius of the CENTLOCAL-algorithm C . The proposition tells us that we can compute the same matching (that is computed by C) by a distributed r -round algorithm.

3 Acyclic Orientation with Bounded Radius (O-RAD)

In this section we define the problem of *Acyclic Orientation with Bounded Radius* (O-RAD). We then design a CENTLOCAL algorithm for O-RAD based on vertex coloring.

Definitions An *orientation* of an undirected graph $G = (V, E)$ is a directed graph $H = (V, A)$, where $\{u, v\} \in E$ if and only if $(u, v) \in A$ or $(v, u) \in A$ but not both. An orientation H is *acyclic* if there are no directed closed paths in H . The *radius* of an acyclic orientation H is the length of the longest directed path in H . We denote the radius of an orientation by $rad(H)$. In the problem of acyclic orientation with bounded radius (O-RAD), the input is an undirected

graph. The output is an orientation H of G that is acyclic. The goal is to compute an acyclic orientation H of G that minimizes $\text{rad}(H)$.

As in [EMR14], an acyclic orientation is induced by a vertex coloring. Previous works obtain an acyclic orientation by random vertex ranking [NO08, YYI12, ARVX12, MRVX12, MV13].

Proposition 2 (Orientation via coloring). *Every coloring by c colors induces an acyclic orientation H with*

$$\text{rad}(H) \leq c - 1.$$

Proof. Direct each edge from a high color to a low color. By monotonicity the orientation is acyclic. Every directed path has at most c vertices, and hence the radius is bounded as required. \square

Many distributed coloring algorithms find a vertex coloring in $O(\log^*(n) + \text{poly}(\Delta))$ rounds (giving us the same upper bound on the probe-radius of the corresponding CENTLOCAL-algorithm) and use $\text{poly}(\Delta)$ colors (see, for example, [BE09, Lin92, CV86, PR01, Kuh09]). For concreteness, in this paper, we employ a CENTLOCAL simulation of a distributed vertex coloring algorithm with $O(\log^*(n) + \text{poly}(\Delta))$ rounds that uses $\text{poly}(\Delta)$ colors.

We remark that a CENTLOCAL-algorithm for O-RAD simply computes, for every vertex v and every port i , whether the edge incident to v at port i is an incoming edge or an outgoing edge in the orientation.

4 A CentLocal-Algorithm for MIS

In this section we briefly describe a CENTLOCAL-algorithm for the MIS problem. The algorithm is a special case of a more general technique of designing CENTLOCAL-algorithms from “greedy” (global) sequential algorithms (see [EMR14, MRVX12]).

Suppose we wish to compute a maximal independent set MIS of a graph $G = (V, E)$. The greedy algorithm proceeds by scanning the vertices in some ordering σ . A vertex v is added to the MIS if none of its neighbors that appear before v in σ have been added to the MIS. Let MIS_σ denote the MIS that is computed by the greedy algorithm if the vertices are scanned by the ordering σ .

Every acyclic orientation $H = (V, A)$ of G induces a partial order P_H simply by considering the transitive closure of H . The key observation is that $\text{MIS}_\sigma = \text{MIS}_\tau$ for every two linear orderings σ and τ that are linear extensions of P_H . Let MIS_{P_H} denote the MIS that corresponds to MIS_σ for linear extensions σ of P_H .

A CENTLOCAL-algorithm can compute MIS_{P_H} as follows. Given v , the algorithm performs a directed DFS from v according to the directed edges A . When the DFS backtracks from a node u , it adds u to MIS_{P_H} if none of the descendants of u in the DFS tree are in MIS_{P_H} .

We conclude with the following lemma that summarizes the above description.

Lemma 3. *Let AO denote a stateless CENTLOCAL-algorithm that computes an acyclic orientation $H = (V, A)$ of a graph $G = (V, E)$. Let r denote the probe radius of AO. Then, there exists a stateless CENTLOCAL-algorithm for MIS whose probe radius is at most $r + \text{rad}(H)$.*

Let L-MIS denote the CENTLOCAL MIS-algorithm in Lemma 3. Let $\text{L-MIS}(G, v)$ denote the Boolean predicate that indicates if v is in the MIS of G computed by Algorithm L-MIS.

5 A CentLocal Approximate MCM Algorithm

In this section we present a stateless deterministic CENTLOCAL algorithm that computes a $(1 - \varepsilon)$ -approximation of a maximum cardinality matching. The algorithm is based on a CENTLOCAL-algorithm for maximal independent set (see Lemma 3) and on the local improvement technique of Nguyen and Onak [NO08].

Terminology and Notation Let M be a matching in $G = (V, E)$. A vertex $v \in V$ is M -free if v is not an endpoint of an edge in M . A simple path is M -alternating if it consists of edges drawn alternately from M and from $E \setminus M$. A path is M -augmenting if it is M -alternating and if both of the path's endpoints are M -free vertices. Note that the length of an augmenting path must be odd. The set of edges in a path p is denoted by $E(p)$, and the set of edges in a collection P of paths is denoted by $E(P)$. Let $A \oplus B$ denote the symmetric difference of the sets A and B .

Description of The Global Algorithm Similarly to [LPSP08, NO08, MV13] our local algorithm simulates the global algorithm listed as Algorithm 1. This global algorithm builds on the next two lemmas of Hopcroft and Karp [HK73], and Nguyen and Onak [NO08], respectively.

Lemma 4 ([HK73]). *Let M be a matching in a graph G . Let k denote the length of a shortest M -augmenting path. Let P^* be a maximal set of vertex disjoint M -augmenting paths of length k . Then, $(M \oplus E(P^*))$ is a matching and the length of every $(M \oplus E(P^*))$ -augmenting path is at least $k + 2$.*

Lemma 5 ([NO08, Lemma 6]). *Let M^* be a maximum matching and M be a matching in a graph G . Let $2k + 1$ denote the length of a shortest M -augmenting path. Then*

$$|M| \geq \frac{k}{k+1} \cdot |M^*|.$$

Algorithm 1 Global-APX-MCM(G, ε).

Input: A graph $G = (V, E)$ and $0 < \varepsilon < 1$.

Output: A $(1 - \varepsilon)$ -approximate matching

- 1: $M_0 \leftarrow \emptyset$.
 - 2: $k \leftarrow \lceil \frac{1}{\varepsilon} \rceil$.
 - 3: **for** $i = 0$ to k **do**
 - 4: $P_{i+1} \leftarrow \{p \mid p \text{ is an } M_i\text{-augmenting path, } |p| = 2i + 1\}$.
 - 5: $P_{i+1}^* \subseteq P_{i+1}$ is a maximal vertex disjoint subset of paths.
 - 6: $M_{i+1} \triangleq M_i \oplus E(P_{i+1}^*)$.
 - 7: **end for**
 - 8: **Return** M_{k+1} .
-

Algorithm 2 Global-APX-MCM'(G, ε).

Input: A graph $G = (V, E)$ and $0 < \varepsilon < 1$.

Output: A $(1 - \varepsilon)$ -approximate matching

- 1: $M_0 \leftarrow \emptyset$.
 - 2: $k \leftarrow \lceil \frac{1}{\varepsilon} \rceil$.
 - 3: **for** $i = 0$ to k **do**
 - 4: Construct the intersection graph H_i over P_i .
 - 5: $P_{i+1}^* \leftarrow \text{MIS}(H_i)$.
 - 6: $M_{i+1} \triangleq M_i \oplus E(P_{i+1}^*)$.
 - 7: **end for**
 - 8: **Return** M_{k+1} .
-

Algorithm 1 is given as input a graph G and an approximation parameter $\varepsilon \in (0, 1)$. The algorithm works in k iterations, where $k = \lceil \frac{1}{\varepsilon} \rceil$. Initially, $M_0 = \emptyset$. The invariant of the algorithm is that M_i is a matching, every augmenting path of which has length at least $2i + 1$. Given M_i , a new matching M_{i+1} is computed as follows. Let P_{i+1} denote the set of shortest M_i -augmenting paths. Let $P_{i+1}^* \subseteq P_{i+1}$ denote a maximal subset of vertex disjoint paths. Define $M_{i+1} \triangleq M_i \oplus E(P_{i+1}^*)$. By Lemmas 4 and 5, we obtain the following result.

Theorem 6. *The matching M_{k+1} computed by Algorithm 1 is a $(1 - \varepsilon)$ -approximation of a maximum matching.*

The intersection graph Define the intersection graph $H_i = (P_i, C_i)$ as follows. The set of nodes P_i is the set of M_{i-1} -augmenting paths of length $2i - 1$. We connect two paths p and q in P_i by an edge $(p, q) \in C_i$ if p and q intersect (i.e., share a vertex in V). Note that H_1 is the line graph of G and that M_1 is simply a maximal matching in G . Observe that P_i^* as defined above is a maximal independent set in H_i . Thus, iteration i of the global algorithm can be conceptualized by the following steps (see Algorithm 2): construct the intersection graph H_i , compute a maximal independent set P_i^* in H_i , and augment the matching by $M_i \triangleq M_{i-1} \oplus (E(P_i^*))$.

Implementation by a stateless deterministic CENTLOCAL Algorithm The recursive local improvement technique in [NO08, Section 3.3] simulates the global algorithm. It is based on a recursive oracle \mathcal{O}_i . The input to oracle \mathcal{O}_i is an edge $e \in E$, and the output is a bit that indicates whether $e \in M_i$. Oracle \mathcal{O}_i proceeds by computing two bits τ and ρ (see Algorithm 3). The bit τ indicates whether $e \in M_{i-1}$, and is computed by invoking oracle \mathcal{O}_{i-1} . The bit ρ indicates whether $e \in E(P_i^*)$ (where P_i^* is an MIS in H_{i-1}). Oracle \mathcal{O}_i returns $\tau \oplus \rho$ because $M_i = M_{i-1} \oplus E(P_i^*)$.

We determine whether $e \in E(P_i^*)$ by running the CENTLOCAL-algorithm \mathcal{A}_i over H_i (see Algorithm 4). Note that \mathcal{A}_1 simply computes a maximal matching (i.e., a maximal independent set of the line graph H_1 of G). The main difficulty we need to address is how to simulate the construction of H_i and probes to vertices in H_i . We answer the question whether $e \in E(P_i^*)$ by executing the following steps: (1) Listing: construct the set $P_i(e) \triangleq \{p \in P_i \mid e \in E(p)\}$. Note that $e \in E(P_i^*)$ if and only if $P_i(e) \cap P_i^* \neq \emptyset$. (2) MIS-step: for each $p \in P_i(e)$, input the query p to an MIS-algorithm for H_i to test whether $p \in P_i^*$. If an affirmative answer is given to one of these queries, then we conclude that $e \in E(P_i^*)$. We now elaborate on how the listing step and the MIS-step are carried out by a CENTLOCAL-algorithm.

The listing of all the paths in $P_i(e)$ uses two preprocessing steps: (1) Find the balls of radius $2i - 1$ in G centered at the endpoints of e . (2) Check if $e' \in M_{i-1}$ for each edge e' incident to vertices in the balls. We can then exhaustively check for each path p of length $2i - 1$ that contains e whether p is an M_{i-1} -augmenting path.

The MIS-step answers a query $p \in P_i^*$ by simulating the MIS CENTLOCAL-algorithm over H_i . The MIS-algorithm probes H_i . A probe to H_i consists of an M_{i-1} -augmenting path q and a port number. We suggest to implement this probe by probing all the neighbors of q in H_i (so the port number does not influence the first part of implementing a probe). See Algorithm 5. As in the listing step, a probe q in H_i can be obtained by (1) finding the balls in G of radius $2i - 1$ centered at endpoints of edges in $E(q)$, and (2) finding out which edges within these balls are in M_{i-1} . The first two steps enable us to list all of the neighbors of q in H_i (i.e., the M_{i-1} -augmenting paths that intersect q). These neighbors are ordered (e.g., by lexicographic

order of the node IDs along the path). If the probe asks for the neighbor of q in port i , then the implementation of the probe returns the i th neighbor of q in the ordering.

By combining the recursive local improvement technique with our deterministic stateless CENTLOCAL MIS-algorithm, we obtain a deterministic stateless CENTLOCAL-algorithm that computes a $(1 - \varepsilon)$ -approximation for maximum matching. The algorithm is invoked by calling the oracle \mathcal{O}_{k+1} . The next lemma can be proved by induction.

Lemma 7. *The oracle $\mathcal{O}_i(e)$ computes whether $e \in M_i$.*

Algorithm 3 $\mathcal{O}_i(e)$ - a recursive oracle for membership in the approximate matching.

Input: A query $e \in E$.

Output: Is e an edge in the matching M_i ?

1: If $i = 0$ then return false.

2: $\tau \leftarrow \mathcal{O}_{i-1}(e)$.

3: $\rho \leftarrow \mathcal{A}_i(e)$.

4: **Return** $\tau \oplus \rho$.

Algorithm 4 $\mathcal{A}_i(e = (u, v))$ - a procedure for checking membership of an edge e in one of the paths in P_i^* .

Input: An edge $e \in E$.

Output: Does e belong to a path $p \in P_i^*$?

1: **Listing step:**

▷ Compute all shortest M_{i-1} -augmenting paths that contain e .

2: $B_u \leftarrow \text{BFS}_G(u)$ with depth $2i - 1$.

3: $B_v \leftarrow \text{BFS}_G(v)$ with depth $2i - 1$.

4: For every edge e' in the subgraph of G induced by $B_u \cup B_v$: $\chi_{e'} \leftarrow \mathcal{O}_{i-1}(e')$.

5: $P_i(e) \leftarrow$ all M_{i-1} -augmenting paths of length $2i - 1$ that contain e .

6: **MIS-step:**

▷ Check if one of the augmenting paths is in P_i^* .

7: For every $p \in P_i(e)$: If $\text{L-MIS}(H_i, p)$ **Return** true.

8: **Return** false.

Algorithm 5 $\text{probe}(i, p)$ - simulation of a probe to the intersection graph H_i via probes to G .

Input: A path $p \in P_i$ and the ability to probe G .

Output: The set of M_{i-1} -augmenting paths of length $2i - 1$ that intersect p .

1: For every $v \in p$ do

2: $B_v \leftarrow \text{BFS}_G(v)$ with depth $2i - 1$.

3: For every edge $e' \in B_v$: $\chi_{e'} \leftarrow \mathcal{O}_{i-1}(e')$.

▷ determine whether the path is alternating and whether the endpoints are M_{i-1} -free.

4: $P_i(v) \leftarrow$ all M_{i-1} -augmenting paths of length $2i - 1$ that contain v .

5: **Return** $\bigcup_{v \in p} P_i(v)$.

6 A DistLocal Approximate MCM Algorithm

In this section, we present a DISTLOCAL-algorithm that computes a $(1 - \varepsilon)$ -approximate maximum cardinality matching. The algorithm is based on collecting information from balls and then simulating the CENTLOCAL algorithm presented in Section 5.

Theorem 8. *There is a deterministic DISTLOCAL $[\Delta^{O(1/\varepsilon)} + O(\frac{1}{\varepsilon^2}) \cdot \log^*(n)]$ -algorithm for computing a $(1 - \varepsilon)$ -approximate MCM.*

Proof. The proof of the theorem is based on the simulation of a CENTLOCAL-algorithm by a DISTLOCAL-algorithm, as summarized in Section 2.4. In Lemma 9 we prove that the probes are restricted to a ball of radius $\Delta^{O(1/\varepsilon)} + O(\frac{1}{\varepsilon^2}) \cdot \log^*(n)$, and the theorem follows. \square

Lemma 9. *The probe radius of the CENTLOCAL-algorithm $\mathcal{O}_{1+\lceil 1/\varepsilon \rceil}$ is*

$$r = \Delta^{O(1/\varepsilon)} + O\left(\frac{1}{\varepsilon^2}\right) \cdot \log^*(n).$$

Proof. Consider a graph G' and a CENTLOCAL-algorithm A that probes G' . Let $r_{G'}(A)$ denote the probe radius of algorithm A with respect to the graph G' .

The description of the oracle \mathcal{O}_i implies that the probe radius $r_G(\mathcal{O}_i)$ satisfies the following recurrence:

$$r_G(\mathcal{O}_i) = \begin{cases} 0 & \text{if } i = 0, \\ r_G(\mathcal{A}_1) & \text{if } i = 1, \\ \max\{r_G(\mathcal{O}_{i-1}), r_G(\mathcal{A}_i)\} & \text{if } i \geq 2. \end{cases}$$

The description of the procedure \mathcal{A}_i implies that the probe radius $r_G(\mathcal{A}_i)$ satisfies the following recurrence:

$$r_G(\mathcal{A}_i) \leq \max\{2i + r_G(\mathcal{O}_{i-1}), 2i - 1 + r_G(\text{L-MIS}(H_i))\}$$

We bound the probe radius $r_G(\text{L-MIS}(H_i))$ by composing the radius $r_{H_i}(\text{L-MIS}(H_i))$ with the increase in radius incurred by the simulation of probes to H_i by probes to G . Recall that the L-MIS-algorithm is based on a deterministic coloring algorithm C . We denote the number of colors used by C to color a graph G' by $|C(G')|$.

The MIS-algorithm orients the edges by coloring the vertices. The radius of the orientation is at most the number of colors. It follows that

$$r_{H_i}(\text{L-MIS}(H_i)) \leq r_{H_i}(C(H_i)) + |C(H_i)|.$$

The simulation of probes to H_i requires an increase in the radius by a factor of $2i - 1$ in addition to the radius of the probes. Hence,

$$r_G(\text{L-MIS}(H_i)) \leq (2i - 1) \cdot r_{H_i}(\text{L-MIS}(H_i)) + r_G(\text{probe}(i, p)).$$

Many distributed coloring algorithms find a vertex coloring in $O(\log^*(n) + \text{poly}(\Delta))$ rounds (giving us the same upper bound on the probe-radius of the corresponding CENTLOCAL-algorithm) and use $\text{poly}(\Delta)$ colors (see, for example, [BE09, Lin92, CV86, PR01, Kuh09]). Plugging these parameters in the recurrences yields

$$\begin{aligned} r_G(\mathcal{O}_i) &\leq 2i + r_G(\text{L-MIS}(H_i)) \\ &\leq 2i \cdot (1 + r_{H_i}(\text{L-MIS}(H_i))) + r_G(\text{probe}(i, p)) \\ &\leq r_G(\mathcal{O}_{i-1}) + O\left(i \cdot \log^*(n(H_i)) + \text{poly}(\Delta(H_i))\right), \end{aligned}$$

Since $\Delta(H_i) \leq (2i)^2 \Delta^{2i-1}$ and $n(H_i) \leq n^{2i}$ we get that

$$\begin{aligned} r_G(\mathcal{O}_k) &\leq \sum_{i=1}^k O\left(i \cdot \log^*(n) + \text{poly}((2i)^2 \cdot \Delta^{2i})\right) \\ &= O(k^2 \cdot \log^*(n)) + \Delta^{O(k)}. \end{aligned}$$

Let $k = 1 + \lceil \frac{1}{\varepsilon} \rceil$, and the lemma follows. \square

7 A CentLocal Approximate MWM Algorithm

In this section we present a deterministic stateless CENTLOCAL-algorithm that computes a $(1 - \varepsilon)$ -approximation of a maximum weighted matching. The algorithm is based on the sublinear approximation algorithm for weighted matching [Ona10, NO08]; we replace the randomized MIS-algorithm by our deterministic MIS-algorithm. The pseudo-code is listed in the Appendix.

Terminology and Notation In addition to the terminology and notation used in the unweighted case, we define the following terms. For a matching M and an alternating path p , the *gain* of p is defined by

$$g_M(p) \triangleq w(p \setminus M) - w(p \cap M).$$

We say that an M -alternating path p is *M -augmenting* if: (1) p is a simple path or a simple cycle, (2) $M \oplus E(p)$ is a matching, and (3) $g_M(p) > 0$. We say that a path p is $(M, [1, k])$ -*augmenting* if p is M -augmenting and $|E(p) \setminus M| \leq k$. Note that an $(M, [1, k])$ -augmenting path may contain at most $2k + 1$ edges.

Preprocessing and Discretization of Weights We assume that the edge weights are positive as nonpositive weight edges do not contribute to the weight of the matching. We also assume that the maximum edge weight is known to all the vertices. By normalizing the weights, we obtain that the edge weights are in the interval $(0, 1]$. Note, that at least one edge has weight 1. As we are interested in a $(1 - \varepsilon)$ -approximation, we preprocess the edge weights by ignoring lightweight edges and rounding down weights as follows: (1) An edge e is *lightweight* if $w(e) < \varepsilon/n$. The contribution of the lightweight edges to any matching is at most $\varepsilon/2$. It follows that ignoring lightweight edges decreases the approximation ratio by at most a factor of $(1 - \varepsilon/2)$. (2) We round down the edge weights to the nearest integer power of $(1 - \varepsilon/2)$. Let $w(e)$ denote the original edge weights and let $w'(e)$ denote the rounded down weights. Therefore, $w(e) \cdot (1 - \varepsilon/2) < w'(e) \leq w(e)$. It follows that, for every matching M , we have $w'(M) \geq (1 - \varepsilon/2) \cdot w(M)$. The combined effect of ignoring lightweight edges and discretization of edge weights decreases the approximation factor by at most a factor of $(1 - \varepsilon)$. We therefore assume, without loss of generality, that the edge weights $w(e)$ are integer powers of $(1 - \varepsilon/2)$ in the interval $[\varepsilon/n, 1]$. Let

$$w_{\min}(\varepsilon) \triangleq \max\{\varepsilon/n, \min_e w(e)\}.$$

In particular, if $w_{\min}(\varepsilon) < 1$ (i.e., the weighted case), then there are at most

$$W \triangleq \Theta\left(\frac{1}{\varepsilon} \cdot \log\left(\frac{1}{w_{\min}(\varepsilon)}\right)\right)$$

distinct weights. This implies that the set of all possible gains achievable by $(M, [1, k])$ -augmenting paths is bounded by $T_k \triangleq \Theta(W^{2k+1})$. We denote the set of T_k possible gains by $\{g_1, \dots, g_{T_k}\}$, where $g_i > g_{i+1}$.

Description of The Global Algorithm The starting point is the global algorithm of Pettie and Sanders [PS04] for approximating an MWM. Onak [Ona10] suggested an implementation of this algorithm that is amenable to localization (See Algorithm 6 and 7 in the Appendix). The

main difference between the algorithms for the weighted case and the unweighted case is that the maximum length of the augmenting paths (and cycles) does not grow; instead, during every step, $(M, [1, k])$ -augmenting paths are used. In [PS04], a maximal set of augmenting paths is computed by greedily adding augmenting paths in decreasing gain order. Discretization of edge weights enables one to simulate this greedy procedure by listing the augmenting paths in decreasing gain order [Ona10].

Algorithm Notation The global algorithm uses the following notation. The algorithm computes a sequence of matchings $M_{i,j}$ that are doubly indexed (where $i \in [1, L]$ and $j \in [1, T_k]$). We denote the initial empty matching by $M_{1,0}$. These matchings are ordered in the lexicographic ordering of their indexes, and $M_{prev(i,j)}$ denotes the predecessor of $M_{i,j}$. Let $P_{i,j}$ denote the set of $(M_{prev(i,j)}, [1, k])$ -augmenting paths whose gain is g_j . Let $H_{i,j}$ denote the intersection graph over $P_{i,j}$ with edges between paths whenever the paths share a vertex. Let G_k denote the intersection graph over all paths of length at most $2k + 1$ in G . Note that each $H_{i,j}$ is the subgraph of G_k induced by $P_{i,j}$. Hence, a vertex coloring of G_k is also a vertex coloring of $H_{i,j}$. Let $P_{i,j}^*$ denote a maximal independent set in $H_{i,j}$.

Implementation by a Stateless Deterministic CENTLOCAL Algorithm The CENTLOCAL implementation of the global algorithm is an adaptation of the CENTLOCAL-algorithm from Section 5. The oracle $\mathcal{O}_{i,j}$ is doubly indexed and so is the procedure $\mathcal{A}_{i,j}$. The algorithm is invoked by calling the oracle \mathcal{O}_{L,T_k} . The next lemma can be proved by induction.

Lemma 10. *The oracle $\mathcal{O}_{i,j}(e)$ computes whether $e \in M_{i,j}$.*

The proof of the following theorem is based on the proof of Theorem 2.4.7 in [Ona10].

Theorem 11. *Algorithm 7 computes a $(1 - \varepsilon)$ -approximate maximum weighted matching.*

8 A DistLocal Approximate MWM Algorithm

In this section, we present a DISTLOCAL-algorithm that computes a $(1 - \varepsilon)$ -approximate weighted matching. The algorithm is based on the same design methodology as in Section 6. Namely, we bound the probe radius of the CENTLOCAL-algorithm for MWM (see Lemma 13) and apply the simulation technique (see Section 2.4).

Theorem 12. *There is a deterministic DISTLOCAL $[r]$ -algorithm for computing a $(1 - \varepsilon)$ -approximate MWM with*

$$r = (\log^*(n) + \Delta^{O(1/\varepsilon)}) \cdot \left(\log \left(\frac{1}{w_{\min}(\varepsilon)} \right) \right)^{O(1/\varepsilon)}.$$

The proof of Theorem 12 is based on the following lemma. Recall that ignoring lightweight edges implies that $\frac{1}{w_{\min}(\varepsilon)} \leq \frac{n}{\varepsilon}$.

Lemma 13. *The probe radius of the CENTLOCAL-algorithm \mathcal{O}_{L,T_k} is*

$$r_G(\mathcal{O}_{L,T_k}) \leq (\log^*(n) + \Delta^{O(1/\varepsilon)}) \cdot \left(\log \left(\frac{1}{w_{\min}(\varepsilon)} \right) \right)^{O(1/\varepsilon)}$$

Proof. The description of the oracle $\mathcal{O}_{i,j}$ implies that the probe radius $r_G(\mathcal{O}_{i,j})$ satisfies the following recurrence:

$$r_G(\mathcal{O}_{i,j}) = \begin{cases} 0 & \text{if } (i, j) = (1, 0), \\ \max\{r_G(\mathcal{O}_{prev(i,j)}), r_G(\mathcal{A}_{i,j})\} & \text{else.} \end{cases}$$

The description of the procedure $\mathcal{A}_{i,j}$ implies that the probe radius $r_G(\mathcal{A}_{i,j})$ satisfies the following recurrence:

$$r_G(\mathcal{A}_{i,j}) \leq O(k) + \max\{r_G(\mathcal{O}_{prev(i,j)}), r_G(\text{L-MIS}(H_{i,j}))\}.$$

The simulation of probes to $H_{i,j}$ implies an increase in the radius by a factor of $2k + 1$ in addition to the radius of the probes. Hence,

$$r_G(\text{L-MIS}(H_{i,j})) \leq (2k + 1) \cdot r_{H_{i,j}}(\text{L-MIS}(H_{i,j})) + r_G(\text{probe}(i, j, p)).$$

The orientation of $H_{i,j}$ can be based on a coloring of the intersection graph G_k . Hence, by Lemma 3

$$r_{H_{i,j}}(\text{L-MIS}(H_{i,j})) \leq r_{G_k}(C(G_k)) + |C(G_k)|.$$

By employing a distributed vertex coloring algorithm with $O(\log^*(n) + \text{poly}(\Delta))$ rounds that uses $\text{poly}(\Delta)$ colors, we obtain

$$\begin{aligned} r_G(\mathcal{O}_{i,j}) &\leq O(k) + r_G(\text{L-MIS}(H_{i,j})) \\ &\leq O(k) \cdot r_{H_{i,j}}(\text{L-MIS}(H_{i,j})) + r_G(\text{probe}(i, j, p)) \\ &\leq r_G(\mathcal{O}_{prev(i,j)}) \\ &\quad + O\left(k \cdot \log^*(n(G_k)) + \text{poly}(\Delta(G_k))\right), \end{aligned}$$

Since $\Delta(G_k) \leq (2k + 1)^2 \Delta^{2k+1}$ and $n(G_k) \leq n^{2k+1}$ we get that

$$\begin{aligned} r_G(\mathcal{O}_{L,T_k}) &\leq L \cdot T_k \cdot O\left(k \cdot \log^*(n) + \text{poly}(\Delta^k)\right) \\ &= O\left(\frac{1}{\varepsilon} \cdot \log\left(\frac{1}{\varepsilon}\right) \cdot W^{O(\frac{1}{\varepsilon})} \cdot \left(\frac{1}{\varepsilon} \cdot \log^*(n) + \text{poly}\left(\Delta^{\frac{1}{\varepsilon}}\right)\right)\right) \\ &= \left(\log^*(n) + \text{poly}\left(\Delta^{\frac{1}{\varepsilon}}\right)\right) \cdot \text{poly}\left(\frac{1}{\varepsilon}\right) \cdot \text{poly}\left(W^{\frac{1}{\varepsilon}}\right), \end{aligned}$$

and the lemma follows. □

9 Future Work

In the full version we present an improved algorithm for the $(1 - \varepsilon)$ -approximate MWM. This improved algorithm computes a $(1 - \varepsilon)$ -approximate MWM within

$$O\left(\frac{1}{\varepsilon^2} \cdot \log\frac{1}{\varepsilon}\right) \cdot \log^* n + \Delta^{O(1/\varepsilon)} \cdot \log\left(\frac{1}{w_{\min}(\varepsilon)}\right)$$

rounds.

References

- [ARVX12] N. Alon, R. Rubinfeld, S. Vardi, and N. Xie. Space-efficient local computation algorithms. In *SODA*, pages 1132–1139, 2012.
- [BE09] L. Barenboim and M. Elkin. Distributed $(\Delta + 1)$ -coloring in linear (in Δ) time. In *STOC*, pages 111–120, 2009.
- [CHW08] A. Czygrinow, M. Hańćkowiak, and W. Wawrzyniak. Fast distributed approximations in planar graphs. In *DISC*, pages 78–92. Springer, 2008.
- [CV86] R. Cole and U. Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf. and Cont.*, 70(1):32–53, 1986.
- [EMR14] G. Even, M. Medina, and D. Ron. Deterministic stateless centralized local algorithms for bounded degree graphs. *Accepted to ESA 2014*, 2014.
- [HK73] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SICOMP*, 2(4):225–231, 1973.
- [HV06] S. Hougardy and D. E Vinkemeier. Approximating weighted matchings in parallel. *IPL*, 99(3):119–123, 2006.
- [Kuh09] F. Kuhn. Weak graph colorings: distributed algorithms and applications. In *SPAA*, pages 138–144. ACM, 2009.
- [Lin92] N. Linial. Locality in distributed graph algorithms. *SICOMP*, 21(1):193–201, 1992.
- [LPSP08] Z. Lotker, B. Patt-Shamir, and S. Pettie. Improved distributed approximate matching. In *SPAA*, pages 129–136, 2008.
- [LW08] C. Lenzen and R. Wattenhofer. Leveraging Linial’s locality limit. In *DISC*, pages 394–407. Springer, 2008.
- [MRVX12] Y. Mansour, A. Rubinfeld, S. Vardi, and N. Xie. Converting online algorithms to local computation algorithms. In *ICALP*, pages 653–664. 2012.
- [MV13] Y. Mansour and S. Vardi. A local computation approximation scheme to maximum matching. In *APPROX*, pages 260–273, 2013.
- [NO08] H. N Nguyen and K. Onak. Constant-time approximation algorithms via local improvements. In *FOCS*, pages 327–336, 2008.
- [Ona10] K. Onak. New sublinear methods in the struggle against classical problems, 2010.
- [Pel00] D. Peleg. *Distributed computing: a locality-sensitive approach*, volume 5. SIAM, 2000.
- [PR01] A. Panconesi and R. Rizzi. Some simple distributed algorithms for sparse networks. *Dist. Comp.*, 14(2):97–100, 2001.

- [PR07] M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theo. Comp. Sci.*, 381(1):183–196, 2007.
- [PS04] S. Pettie and P. Sanders. A simpler linear time $2/3$ -epsilon approximation for maximum weight matching. *IPL*, 91(6):271–276, 2004.
- [PS10] A. Panconesi and M. Sozio. Fast primal-dual distributed algorithms for scheduling and matching problems. *Dist. Comp.*, 22(4):269–283, 2010.
- [RTVX11] R. Rubinfeld, G. Tamir, S. Vardi, and N. Xie. Fast local computation algorithms. In *ICS*, pages 223–238, 2011.
- [Suo13] J. Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24:1–24:40, 2013.
- [YYI12] Y. Yoshida, M. Yamamoto, and H. Ito. Improved constant-time approximation algorithms for maximum matchings and other optimization problems. *SICOMP*, 41(4):1074–1093, 2012.

A Pseudo-Code for CentLocal($1 - \varepsilon$)-approximate MWM-Algorithm

Algorithm 6 Global-APX-MWM(G, ε) -Onak’s adaption [Ona10] of the global algorithm of Pettie and Sanders [PS04].

Input: A graph $G = (V, E)$ with edge weights $w(e) \in (0, 1]$ that are integer powers of $1 - \frac{\varepsilon}{3}$ for $0 < \varepsilon < 1$.
Output: A $(1 - \varepsilon)$ -approximate weighted matching

- 1: $k \leftarrow \lceil \frac{3}{\varepsilon} \rceil$.
- 2: $L \leftarrow \Theta\left(\frac{1}{\varepsilon} \cdot \log\left(\frac{1}{\varepsilon}\right)\right)$.
- 3: $T_k = \Theta\left(\frac{1}{\varepsilon} \cdot \log\left(\frac{1}{w_{\min}(\varepsilon)}\right)\right)^{2k+1}$. $\triangleright T_k$ is an upper bound on the number of distinct gains of augmenting paths of length at most $2k + 1$.
- 4: $M \leftarrow \emptyset$.
- 5: **for** $i = 1$ to L **do**
- 6: **for** $j = 1$ to T_k **do**
- 7: $P_{i,j}^*$ is a maximal set of vertex disjoint $(M, [1, k])$ -augmenting paths with gain g_j .
- 8: $M \leftarrow M \oplus E(P_{i,j}^*)$.
- 9: **end for**
- 10: **end for**
- 11: **Return** M .

Algorithm 7 Global-APX-MWM'(G, ε) -Rewriting of Algorithm 6 using the intersection graph $H_{i,j}$.

Input: A graph $G = (V, E)$ with edge weights $w(e) \in (0, 1]$ that are integer powers of $1 - \frac{\varepsilon}{3}$ for $0 < \varepsilon < 1$.

Output: A $(1 - \varepsilon)$ -approximate weighted matching

- 1: k, L, T_k as in Algorithm 6
 - 2: $M_{1,0} \leftarrow \emptyset$.
 - 3: **for** $i = 1$ to L **do**
 - 4: **for** $j = 1$ to T_k **do**
 - 5: $P_{i,j} \leftarrow$ set of $(M_{prev(i,j)}, [1, k])$ -augmenting paths with gain g_j .
 - 6: Construct the intersection graph $H_{i,j}$ over $P_{i,j}$.
 - 7: $P_{i,j}^* \leftarrow \text{MIS}(H_{i,j})$.
 - 8: $M_{i,j} \leftarrow M_{prev(i,j)} \oplus E(P_{i,j}^*)$.
 - 9: **end for**
 - 10: **end for**
 - 11: **Return** M_{L, T_k} .
-

Algorithm 8 $\mathcal{O}_{i,j}(e)$ - a recursive oracle for membership in the approximate weighted matching.

Input: A query $e \in E$.

Output: Is e an edge in the weighted matching $M_{i,j}$?

- 1: If $(i, j) = (1, 0)$ then return false.
 - 2: $\tau \leftarrow \mathcal{O}_{prev(i,j)}(e)$.
 - 3: $\rho \leftarrow \mathcal{A}_{i,j}(e)$.
 - 4: **Return** $\tau \oplus \rho$.
-

Algorithm 9 $\mathcal{A}_{i,j}(e = (u, v))$ - a procedure for checking membership of an edge e in one of the paths in $P_{i,j}^*$.

Input: An edge $e \in E$.

Output: Does e belong to a path $p \in P_{i,j}^*$?

- 1: **Listing step:** ▷ Compute all shortest M_{i-1} -augmenting paths that contain e .
 - 2: $B_u \leftarrow \text{BFS}_G(u)$ with depth $2k$.
 - 3: $B_v \leftarrow \text{BFS}_G(v)$ with depth $2k$.
 - 4: For every edge e' in the subgraph of G induced by $B_u \cup B_v$: $\chi_{e'} \leftarrow \mathcal{O}_{prev(i,j)}(e')$.
 - 5: $P_{i,j}(e) \leftarrow$ all $(M_{prev(i,j)}, [1, k])$ -augmenting paths that contain e with gain g_j
 - 6: **MIS-step:** ▷ Check if one of the augmenting paths is in P_i^* .
 - 7: For every $p \in P_{i,j}(e)$: If L-MIS($H_{i,j}, p$) **Return true**.
 - 8: **Return false**.
-

Algorithm 10 $probe(i, j, p)$ - simulation of a probe to the intersection graph $H_{i,j}$ via probes to G .

Input: A path $p \in P_i$ and the ability to probe G .

Output: The set of $(M_{prev(i,j)}, k^*)$ -augmenting paths with gain g_j that intersect p .

- 1: For every $v \in p$ do
 - 2: $B_v \leftarrow \text{BFS}_G(v)$ with depth $2k + 1$.
 - 3: For every edge $e' \in B_v$: $\chi_{e'} \leftarrow \mathcal{O}_{prev(i,j)}(e')$. ▷ determines whether the path is alternating and whether the endpoints are free.
 - 4: $P_{i,j}(v) \leftarrow$ all $(M_{prev(i,j)}, [1, k])$ -augmenting paths that contain v with gain g_j .
 - 5: **Return** $\bigcup_{v \in p} P_{i,j}(v)$.
-