Distributed Approximation Algorithm for Matching

Subhas Kumar Ghosh

Honeywell Technology Solutions Lab.,
151/1, Doraisanipalya, Bannerghatta Road,
Bangalore, INDIA-560076

Abstract

We present a randomized distributed approximation algorithm for maximum matching. Given a graph $G = (V, E)$ with vertex set V, our algorithm (with high probability) approximates maximum matching to within a factor arbitrarily close to $3/2$ and has running time of $\mathcal{O}\left(\log^2 |V|\right)$.

Key words: Graph algorithms, distributed algorithm, approximation algorithm, maximum matching.

1 Introduction

Given an undirected graph $G = (V, E)$ with vertex set V and edge set E, where $|V| = n$, a *matching* M is a subset of the edges such that no two edges in M have common endpoints. The *maximum matching* problem is to find a matching in G that maximizes cardinality of the matching.

Edmonds [1] in 1965 obtained the first polynomial time sequential algorithm for the maximum weight matching problem. Since then matching played one of the most important role in the development of tools and techniques for sequential algorithms and has been studied extensively. However, unlike sequential algorithms for matching, until recently only a few efficient distributed algorithms were known for computing maximal matching or approximate maximum weight matching. Israeli and Itai [2] presented first randomized distributed algorithm to compute *maximal matching*, with expected running

Email address: `subhas.kumar@honeywell.com` (Subhas Kumar Ghosh).

time $\mathcal{O}(\log n)$. There are deterministic distributed algorithms for computing the maximal matching [3] in $\mathcal{O}\left(\log^4 n\right)$ time. A deterministic distributed algorithm having 1.5-approximation of unweighted maximum matching was presented in [4], improving, and using techniques from [3], which has running time $\mathcal{O}\left(\log^4 n\right)$.

There are RNC^2 algorithms for maximum matching [5], however maximum matching is not known to be in NC. Recently, NC approximation algorithms with an approximation factor $(1+\epsilon)$ for weighted matching in $\mathsf{CREW\ PRAM}$ model has been presented in [6], improving and using techniques of [7]. Finally we note that in [8], Kuhn et al. proved a $\Omega(\sqrt{\log n/\log\log n}+\log\Delta/\log\log\Delta)$ lower bound on the time complexity for (possibly randomized) distributed algorithms achieving a constant factor approximation for maximum matching, even with unbounded message size.

In this paper, we develop a randomized distributed approximation algorithm for maximum matching which with high probability approximates maximum matching to within a factor arbitrarily close to 1.5 and has running time of $\mathcal{O}\left(\log^2 n\right)$. This can be compared with deterministic algorithm of [4] having same approximation ratio but running time $\mathcal{O}\left(\log^4 n\right)$.

Rest of the paper is organized as follows. In Section-2 we present the model, required definitions and known results which will be used in following sections. In Section-3 we present distributed approximation algorithm for maximum matching problem. Finally, in Section-4 we conclude the paper.

2 Preliminaries

2.1 Model

We consider the classical $\mathcal{LOCAL}$ distributed model of computation (cf. [9, Chapter 2],[10]). In specific, we model the network as a connected undirected graph $G=(V,E), |V|=n$. Vertices V of the graph are processing units, and edges are communication links. We assume that communication is synchronous, i.e. each node follows a global clock. Thus round of the distributed algorithm is determined by the global clock. In each round nodes can send and receive message and local computation in nodes are atomic. Time complexity of the algorithm is determined by the rounds as measured by the global clock in worst case. In a round a node send messages to any subset of its neighbors. We also assume that all messages sent in a round are received and processed in the same round.

2.2 Short Augmentation

We will use standard graph theoretic terminology. Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E, where $|V| = n$. Given a set of edges $X \subseteq E(G)$, let $G[X]$ denote the subgraph of G induced by X. With $d_G(u)$ we will denote the degree of a vertex $u \in V(G)$. A *matching* M is a subset of the edges such that no two edges in M have common endpoints. A vertex (resp. an edge) is M-saturated, if it is endpoint of an edge in M (resp. if it intersects with and edge in M). We will denote a maximum matching by M^*. A matching M is a γ-approximation of maximum matching if $\gamma \cdot |M| \geq |M^*|$.

Given a graph G and a matching M on G, a path or cycle is *alternating* if it consists of edges taken from M and $E \setminus M$ alternately. An alternating path or cycle P is said to be an *augmentation* if $M \oplus P$ is also a matching on G, where $A \oplus B = (A \setminus B) \cup (B \setminus A)$. For a set of vertex-disjoint alternating paths or cycles A we extend the notion of augmentation naturally and write it as $M \oplus A$. An augmentation with at most k non-M edges is a k-augmentation. Length of an augmentation is the number of edges in it. For unweighted graph, gain of an alternating path or cycle P is $g(P) \triangleq |P \setminus M| - |P \cap M|$. In sequel we state a useful result:

Theorem 2.1 ([4]) *Let M be any matching on G.*

(1) If M admits no k-augmentation, then $|M| \geq k/(k+1)\,|M^*|$.
(2) On the other hand, let M be any maximal matching, and let A be a collection of maximal vertex disjoint k-augmentations of positive gain augmenting M. Further, let $M' = M \oplus A$. Then there are no k-augmentations augmenting M' in G.

2-augmentations will play important role in our algorithm. We note that 2-augmentation of length 3, for which no end is M-saturated by matching M, are only type of 2-augmentation useful for maximum matching, as augmenting them increases cardinality of matching.

3 Approximation Algorithm for Maximum Matching

Our algorithm is based on the techniques presented by Czygrinow, Hanckowiak and Szymanska [4]. Algorithm presented in [4] is a deterministic distributed algorithm having 1.5-approximation of maximum matching and has running time $\mathcal{O}\left(\log^4 n\right)$. In their algorithm the dominating factor of $\mathcal{O}\left(\log^3 n\right)$ is contributed by distributed construction of a structure called (α, K)–spanner (defined below). In this section we present a randomized distributed algorithm for constructing an (α, K)–spanner in $\mathcal{O}(\log n)$ rounds, thus making the overall running time of this (randomized) algorithm $\mathcal{O}\left(\log^2 n\right)$. First, we briefly present the algorithm of Czygrinow et al. [4], and then describe the changes, namely construction of (α, K)–spanner in detail.

First step of the algorithm is to compute a maximal matching in $\mathcal{O}(\log n)$ rounds, for which we will use the randomized algorithm of [2]. Let, M be the maximal matching returned by this algorithm. Next we will consider augmenting this matching with set of vertex disjoint 2–augmentations P of length three, for which no end is M–saturated. Following [4], we will call such augmentation $(M, 3)$–path. By Theorem-2.1, if we obtain P^*, a maximal independent set of $(M, 3)$–paths, and compute $M' = M \oplus P^*$, then $|M'| \geq \frac{2}{3}|M^*|$, and M' is the desired matching. This will be achieved in $\mathcal{O}(\log n)$ iterations. In each iteration we will compute a constant factor approximation of any P^*, a maximal independent set of $(M, 3)$–paths, using $\mathcal{O}(\log n)$ rounds.

Let, $\mathcal{P}_3(M)$ denote the set of all $(M, 3)$–paths in G, with respect to a matching M. Observe that $|\mathcal{P}_3(M)|$ is $\mathcal{O}(n^4)$. For a set of edges $X \subseteq E(G)$, let $\mathsf{touch}(X)$ denote the set of edges in $E(G)$ that share at least one common vertex with some edge in X. Now consider if in each iteration i if we compute a set of vertex disjoint set of $(M, 3)$–paths P_i such that $|\mathsf{touch}(E(P_i))| \geq \gamma \cdot |\mathcal{P}_3(M)|$, for some constant $\gamma > 0$, remove $\mathsf{touch}(E(P_i)) \cup E(P_i)$ to obtain a new graph G_i as input for the next iteration, then after $4 \log_c n$ (with $c = 1/\gamma$) iterations there will be no $(M, 3)$–paths left in G. So we output $P = \cup_i P_i$ as maximal independent set of $(M, 3)$–paths and compute $M' = M \oplus P$.

In following, for $\gamma > 0$, a set P of vertex disjoint $(M, 3)$–paths will be called γ–substantial in G if $|\mathsf{touch}(E(P))| \geq \gamma \cdot |\mathcal{P}_3(M)|$. Let $H = (L, R, E)$ be a bipartite multigraph. For $i = 0, 1, \ldots, \log n$, D_i–block of H is a sub-multigraph of H induced by the edges incident to vertices $L_i = \{u \in L : D_i/2 < d_H(u) \leq D_i\}$ for $D_i = 2^i$, and is denoted by $H_i = (L_i, \mathcal{N}(L_i), E_i)$. Observe that for $i = 0, 1, \ldots, \log n$, L_i is a partition of vertices in L, while vertices in $\mathcal{N}(L_i) \subseteq R$ can be in more than one D_i–block of H. Given a D–block $B = (L, \mathcal{N}(L), E)$ for some D, for a subgraph $S \subseteq B(L, \mathcal{N}(L), E)$, we use the notation $l(S) \triangleq V(S) \cap L$ and $r(S) \triangleq V(S) \cap \mathcal{N}(L)$. We need following definition:

Definition 3.1 *Let $B = (L, \mathcal{N}(L), E)$ be a D–block for some D. An (α, K)–spanner of B is a subgraph $S \subseteq B$ such that*

(1) $|l(S)| \geq \alpha \cdot |L|$, and
(2) $\forall u \in l(S), d_S(u) \in [1, 4]$, and
(3) $\forall v \in r(S), d_S(v) < \frac{K}{D} \cdot d_B(v) + 1$.

Following proposition clarifies how an efficient construction of an (α, K)–spanner of a D–block B will be useful in designing efficient $\mathsf{UWM}_{3/2}$.

Proposition 3.1 *Let $c > 0$ be a constant, and a multigraph $B = (L, R, E)$ be a D–block for some D such that $|E(B)| \leq n^c$ and $n = |L| + |R|$. Given a distributed algorithm to construct an (α, K)–spanner of B in $\mathcal{O}(t(n))$ many rounds, for $K = K(c)$, and $0 < \alpha \leq 1/2$, there is a distributed procedure which in $\mathcal{O}(t(n) \cdot \log n)$ many rounds produces a matching M such that $|M| \geq \frac{2}{3}|M^*|$.*

In specific taking $c = 4$, proof of this proposition follows from [4, Theorem 3], hence in following we describe efficient randomized construction of an (α, K)–spanner of a multigraph $B = (L, R, E)$ that is a D–block for some D such that $|E(B)| \leq n^c$ and $n = |L| + |R|$ in $\mathcal{O}(\log n)$ rounds.

Let $B = (L, R, E)$ be a D–block defined by the set of left-hand vertices

$$L \triangleq \{u : D/2 < d_B(u) \leq D\}.$$

Procedure Spanner works as follows (see Algorithm-1(a)). Given B, a 2 - decomposition of B into a collection of disjoint paths and cycles can be computed as follows. In parallel, every vertex $u \in V(B)$ splits itself into vertices $\{u_1, \ldots, u_m\}$, each having degree two (by pairing any two incident edges) and one vertex having degree one if $d_B(u)$ is odd. vertices $\{u_1, \ldots, u_m\}$ will be called *siblings* of *parent* u. Obtained graph (a 2–decomposition of B) has $|E|$ edges, and this decomposition can be computed in constant number of rounds. Let, $C_1, \ldots, C_p$ be connected paths and cycles that are formed. Subsequently, a procedure Matching (see Algorithm-1(b)) with parameter r is invoked for every paths and cycles, where every vertex (as a representative of its sibling) in parallel participates in computing a matching J_i of component C_i. Procedure Matching is same as one given in [11]. Denoting $B^0 = B$ and $L_0 = l(B^0)$, the procedure generates a sequence of blocks $B^0, B^1, \ldots, B^k$, such that $L_k \subseteq L_{k-1} \subseteq \ldots \subseteq L_0$, where block B^{j+1} is obtained from block B^j by removing all matched edges $J = \cup_i J_i$ obtained in jth iteration, and all vertices for which all incident edges belongs to J. In following we analyze the procedure Spanner and show that after kth round block B^k is the desired spanner. For a vertex $v \in V(B^j)$, let $d_j(v)$ denote the degree of vertex v in subgraph B^j.

Lemma 3.1 *For all iterations $j = 0, \ldots, k-1$ of procedure* Spanner*, for all $v \in B^{j+1}$*

$$\frac{1}{2}(d_j(v) - 1) \leq \mathbf{E}[d_{j+1}(v)] \leq \left(\frac{3}{4}\right)^r d_j(v)$$

Proof: Let, $\{v_1, \ldots, v_m\}$ be siblings of vertex v in $2-$ decomposition of B^j. We have $d_j(v) = \sum_{i=1}^m d_j(v_i)$, and similarly $d_{j+1}(v) = \sum_{i=1}^m d_{j+1}(v_i)$. Then, by linearity of expectation: $\mathbf{E}[d_{j+1}(v)] = \sum_{i=1}^m \mathbf{E}[d_{j+1}(v_i)]$. Now, for a sibling v_i, at most one edge incident to it is dropped, and probability that an edge is selected after r round is at least $\left(1 - \left(\frac{3}{4}\right)^r\right)$: A vertex v in 2–

```
procedure Spanner $(B, r)$
    $B^0 \leftarrow B$, $L_0 \leftarrow l\left(B^0\right)$.
    for $j \leftarrow 0$ to $k = \mathcal{O}(\log D)$ do
        $J \leftarrow \emptyset$.
        $\forall u \in V(B)$, in parallel, split by pairing-up edges to obtain a 2-decomposition of $B^j$.
        $J_i \leftarrow$ Matching $(C_i, r)$.        ▷ Let $C_i$ be a path or a cycle in 2-decomposition
        Mark all edges in $J = \bigcup_i J_i$
        Mark $v$ to be removed from $L_j$ to obtain $L_{j+1}$ if all edges incident on $v$ is marked.
        Remove $J$ from $E\left(B^j\right)$ to obtain $B^{j+1}$.
    end for
end procedure
```

(a) (α, K)–Spanner

```
procedure Matching $(C, r)$
    $M \leftarrow \emptyset$, $i \leftarrow 0$.
    while $i < r$ do
        Choose uniformly at random one incident edge $e = (u, v) \in C$.  ▷ In every sibling node $u$
        request over $e$.
        if received from $v$ then
            $M \leftarrow M \cup \{e\}$.
            Remove touch $(e)$.
            return.
        end if
    end while
    Let $M$ be the matching returned.
end procedure
```

(b) Matching (C, r)

Fig. 1. Distributed algorithm to to construct a (α, K)–spanner

decomposition has at most 2 incident edges. Thus, during the execution of procedure Matching each vertex selects a incident edge with probability at least 1/2. While vertices choose independently, an edge is chosen with probability at least 1/4. Hence, an edge is selected in matching after r iterations is at least $\sum_{i=1}^{r} (3/4)^i \, 1/4 = (1 - (3/4)^r)$. Hence we have, $\mathbf{E}\left[d_{j+1}(v)\right] \leq \left(\frac{3}{4}\right)^r d_j(v)$. The lower bound, on the other hand follows by considering $d_j(v)$ to be odd, and assuming that each of its sibling is matched. □

Lemma 3.2 *Let Δ_j (resp. δ_j) denote the maximum (resp. minimum) degree* *of any vertex in $l\left(B^j\right)$. For all iterations $j = 0, \ldots, k-1$ of procedure* Spanner

$$\mathbf{E}\left[\left|L_{j+1}\right|\right] \geq \left|L_j\right| \left(1 - \frac{\delta_j}{\Delta_j}\right)$$

Proof: In the procedure Spanner, we drop those vertices from B^j in step j, for which all incident edges are marked. Let X_v denote the random variable taking values in $\{0, 1\}$ such that $X_v = 1$ iff $v \in l\left(B^j\right)$ has $d_{j+1}(v) \geq 1$, and $X_v = 0$ otherwise. Then, $\mathbf{E}\left[\left|L_{j+1}\right|\right] = \sum_{v \in l(B^j)} \mathbf{E}\left[X_v\right] = \left|L_j\right| \mathbf{Pr}\left[d_{j+1}(v) \geq 1\right] = \left|L_j\right| \left(1 - \mathbf{Pr}\left[d_{j+1}(v) < 1\right]\right)$. Now, observe that $d_{j+1}(v)$ is a nonnegative random variable, and possible values that $d_{j+1}(v)$ can get are integers. First we show following. Let T be the largest integer that is smaller than $\mathbf{E}\left[d_{j+1}(v)\right]$, and let $\mathbf{Pr}\left[d_{j+1}(v) \geq \mathbf{E}\left[d_{j+1}(v)\right]\right] = \alpha$ then:

$$\begin{aligned}
&\mathbf{E}\left[d_{j+1}(v)\right] \\
&= \sum_{x \geq \mathbf{E}[d_{j+1}(v)]} x \cdot \mathbf{Pr}\left[d_{j+1}(v) = x\right] + \sum_{x < \mathbf{E}[d_{j+1}(v)]} x \cdot \mathbf{Pr}\left[d_{j+1}(v) = x\right] \\
&\leq d_j(v) \cdot \mathbf{Pr}\left[d_{j+1}(v) \geq \mathbf{E}\left[d_{j+1}(v)\right]\right] + T \cdot \mathbf{Pr}\left[d_{j+1}(v) < \mathbf{E}\left[d_{j+1}(v)\right]\right] \\
&\leq \alpha d_j(v) + (1-\alpha).T
\end{aligned}$$

From the definition of T, we have $T = \lceil \mathbf{E}\left[d_{j+1}(v)\right] - 1 \rceil \leq \mathbf{E}\left[d_{j+1}(v)\right] - 1/4$, as $\mathbf{E}\left[d_{j+1}(v)\right] \leq 3/4 \cdot d_j(v)$. With these two inequalities we obtain that $\mathbf{Pr}\left[d_{j+1}(v) \geq \mathbf{E}\left[d_{j+1}(v)\right]\right] = \alpha \geq 1/\left(d_j(v) + 1\right)$. Using the bound on α, we obtain following upper bound on $\mathbf{Pr}\left[d_{j+1}(v) < 1\right]$.

$$\mathbf{Pr}\left[d_{j+1}(v) < 1\right] = 1 - \mathbf{Pr}\left[d_{j+1}(v) \geq 1\right] \leq 1 - \alpha \leq \frac{\delta_j}{\Delta_j + 1}$$

Since $\delta_j \leq d_j(v) \leq \Delta_j$. Thus, we have

$$\mathbf{E}\left[\left|L_{j+1}\right|\right] \geq \left|L_j\right| \left(1 - \frac{\delta_j}{\Delta_j}\right)$$

□

Finally, we present the analysis of the procedure Spanner.

Theorem 3.1 *Let $c > 0$ be a constant, and let $B = (L, R, E)$ be a D–block for some D such that $|E(B)| \leq n^c$ and $n = |L| + |R|$. Given B, the distributed procedure* Spanner *constructs an (α, K)–spanner of B in $\mathcal{O}(\log n)$ many rounds, such that $K = K(c) = (4/3)^{c+1}$, and $\alpha = 1/4$.*

Proof: Let us fix $k > 0$ and observe that $D/2 < d_0(v) \leq D$. By induction over Lemma-3.1 we have:

$$\left(\frac{1}{2}\right)^{k+1} D \leq \mathbf{E}\left[d_k\left(v\right)\right] \leq \left(\frac{3}{4}\right)^{rk} D$$

Choosing $k = \left\lceil \log_{4/3} D \right\rceil - c$, $r = 1$ and with $c = 4$, we have:

$$\left(\frac{1}{2}\right)^{k+1} D > 0 \text{ and, } \left(\frac{3}{4}\right)^{rk} D < 2^{c-2} = 4$$

In other words, vertices from L_k, will have degree in $[1, 4]$. This establishes condition 2 of spanner, for condition 3, with the choice of parameters observe that from the upper bound on degree of any vertex, we have $K\left(c\right) = \left(4/3\right)^{c+1}$. Finally, using lemma-3.2, we show condition 1 for spanner. We have for all iterations j of procedure Spanner

$$\mathbf{E}\left[\left|L_{j+1}\right|\right] \geq \left|L_j\right| \left(1 - \frac{\delta_j}{\Delta_j}\right)$$

Applying this recursively we obtain:

$$\mathbf{E}\left[\left|L_k\right|\right] \geq \left|L_0\right| \prod_{j=1}^{k-1} \left(1 - \frac{\delta_j}{\Delta_j}\right)$$

Since

$$\frac{\delta_j}{\Delta_j} = \frac{\left(\frac{1}{2}\right)^{j+1} D}{\left(\frac{3}{4}\right)^{j} D} = \frac{1}{2} \cdot \left(\frac{2}{3}\right)^{j}$$

with above inequalities, we have,

$$\mathbf{E}\left[\left|L_k\right|\right] \geq \left|L_0\right| \prod_{j=1}^{k-1} \left(1 - \frac{1}{2}\left(\frac{2}{3}\right)^{j}\right) \geq \frac{1}{4}\left|L_0\right|$$

□

Using Theorem-3.1, with Proposition-3.1, we have following high-level result.

Theorem 3.2 *There is a distributed randomized procedure which produces a matching M such that $|M| \geq \frac{2}{3}|M^*|$ w.h.p, and has $\mathcal{O}\left(\log^2 n\right)$ deterministic round complexity.*

4 Concluding remarks

Obtaining a distributed approximation algorithm (possibly randomized) with $\mathcal{O}\left(\log n\right)$ time complexity having $(1+\epsilon)$–approximation factor is a natural open problem. Finally, obtaining a $\mathcal{O}\left(\log n\right)$ time deterministic distributed algorithm for maximal matching in the same model remains elusive.

References

[1] J. Edmonds, Path, trees, and flowers, Canadian J. Math. 17 (1965) 449–467.

[2] A. Israeli, A. Itai, A fast and simple randomized parallel algorithm for maximal matching, Inf. Process. Lett. 22 (2) (1986) 77–80.

[3] M. Hańćkowiak, M. Karonski, A. Panconesi, On the distributed complexity of computing maximal matchings, SIAM J. Discret. Math. 15 (1) (2002) 41–57.

[4] A. Czygrinow, M. Hańćkowiak, E. Szymańska, A fast distributed algorithm for approximating the maximum matching, in: S. Albers, T. Radzik (Eds.), ESA, Vol. 3221 of Lecture Notes in Computer Science, Springer, 2004, pp. 252–263.

[5] K. Mulmuley, U. V. Vazirani, V. V. Vazirani, Matching is as easy as matrix inversion, in: STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing, ACM Press, New York, NY, USA, 1987, pp. 345–354.

[6] S. Hougardy, D. E. Vinkemeier, Approximating weighted matchings in parallel, Inf. Process. Lett. 99 (3) (2006) 119–123.

[7] R. Uehara, Z.-Z. Chen, Parallel approximation algorithms for maximum weighted matching in general graphs, Inf. Process. Lett. 76 (1-2) (2000) 13–17.

[8] F. Kuhn, T. Moscibroda, R. Wattenhofer, The price of being near-sighted, in: SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, ACM Press, New York, NY, USA, 2006, pp. 980–989.

[9] D. Peleg, Distributed computing: a locality-sensitive approach, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[10] N. Linial, Locality in distributed graph algorithms, SIAM J. Comput. 21 (1) (1992) 193–201.

[11] M. Wattenhofer, R. Wattenhofer, Distributed weighted matching, in: R. Guerraoui (Ed.), DISC, Vol. 3274 of Lecture Notes in Computer Science, Springer, 2004, pp. 335–348.