

Distributed $O(\Delta \log n)$ -edge-coloring algorithm ^{*}

A. Czygrinow¹, M. Hańkowiak², and M. Karoński³

¹ Faculty of Mathematics and Computer Science
Adam Mickiewicz University, Poznań, Poland
and Arizona State University, Tempe, Az 85287-1804, USA
`andrzej@math.la.asu.edu`

² Faculty of Mathematics and Computer Science
Adam Mickiewicz University, Poznań, Poland
`mhanckow@main.amu.edu.pl`

³ Faculty of Mathematics and Computer Science
Adam Mickiewicz University, Poznań, Poland
`karonski@main.amu.edu.pl`
and Emory University, Atlanta, Ga 30322, USA
`micchal@mathcs.emory.edu`

Abstract. Let $G = (V, E)$ be a graph on n vertices and let Δ denote the maximum degree in G . We present a distributed algorithm that finds a $O(\Delta \log n)$ -edge-coloring of G in time $O(\log^4 n)$.

1 Introduction

In this paper, we consider a problem of edge-coloring of a graph in a distributed model of computations. In our model a network is represented by an undirected graph $G = (V, E)$ where each vertex represents a processor of the network and an edge corresponds to a connections between processors. We assume full synchronization of the network: in every step, each processor sends messages to all its neighbors, receives messages from all of its neighbors, and can perform some local computations. The number of steps should be polylogarithmic in the size of the graph, and in addition we insist that the local computations of each processor are performed in polynomial time. The above model is more restrictive than a classical distributed model introduced by Linial in [Li92]. In Linial's model there is no restriction on local computations performed by processors (for example processors can perform computations in exponential time). By default, all processors have different IDs, each processor knows $|V|$, the number of vertices in G , and $\Delta(G)$, the maximal degree in G .

In the edge-coloring problem the goal of a distributed algorithm is to properly color the edges of G in a polylogarithmic (in $n = |V|$) number of steps. The main difficulty of designing such an algorithm comes from the fact that in such a "short time" a vertex v can learn only about vertices and edges that are within a "small" distance from v and based on this local information, a proper

^{*} This work was supported by KBN GRANT 7 T11C 032 20

coloring of E must be obtained. Let Δ denote the maximum degree of G . By Vizing's theorem there is a proper edge-coloring of G with $\Delta + 1$ colors but the known proofs of this theorem don't lead to a distributed algorithm. It is therefore natural to aim for an algorithm that uses $O(\Delta)$ colors. In [Li92], Linial presented an algorithm which, in $\log^* n$ number of steps, colors vertices of G using $O(\Delta^2)$ colors. Linial's procedure can be used to obtain a $O(\Delta^2)$ -edge-coloring of G . Very recently, De Marco and Pelc claimed that an algorithm presented in [MaPe01] colors the vertices of G in $O(\Delta)$ colors. Unfortunately, no complete proof of the main lemma in [MaPe01] has been offered. In addition, in their algorithm the amount of local computations is not polylogarithmically bounded.

In this paper, we present a distributed algorithm which colors edges of graph in $O(\Delta \log n)$ colors. Our approach is based on computing a family of spanners of G . It turns out that this family can be used to color a constant fraction of edges of G using $O(\Delta)$ colors. Iterating this process $O(\log n)$ steps leads to a proper coloring of E . However in each iteration a palette of $O(\Delta)$ new colors is needed. Spanners were previously successfully used by Hańćkowiak, Karoński and Panconesi [HKP99], to design a distributed algorithm for a maximal matching problem.

The rest of this paper is structured as follows. In Section 2, we present a procedure that constructs a family of spanners. Section 3 contains the description of our main algorithm and the proof of its correctness.

2 Family of Spanners

In this section, we present an algorithm that finds a family of spanners that are used to color our graph. The main idea of the algorithm is as follows. Suppose for a moment that all vertices in the graph have degree that are powers of two. In order to color the edges, we find a subgraph such that each vertex has degree (in this subgraph) equal to half of the original degree. We assign one to edges of the subgraph and zero to remaining edges. Next we repeat the procedure in "one-subgraph" and in "zero-subgraph". As a result we obtain a sequence of zeros and ones on each edge. This sequence is a color of an edge. Note that in the distributed model we can not split a graph in such an exact way, but we can do it approximately.

Let us start with some definitions. A bipartite graph $H = (A, B, E)$ is called a D -block if for every vertex $a \in A$,

$$\frac{D}{2} < \deg_H(a) \leq D.$$

Definition 1. An (α, β) -spanner of a D -block $H = (A, B, E)$ is a subgraph $S = (A', B, E')$ of H such that the following conditions are satisfied.

1. $|A'| \geq \alpha|A|$.
2. For every vertex $a \in A'$, $\deg_S(a) = 1$.
3. For every vertex $b \in B$, $\deg_S(b) < \frac{\beta}{D} \deg_H(b) + 1$.

Procedure FINDSPANNERS finds a family of $O(D)$ edge-disjoint (α, β) - spanners in a D -block for some constants α and β . In each iteration of the main loop of FINDSPANNERS we invoke a procedure SPLITTER that, in parallel, to each edge e assigns the labels $bit(e) \in \{0, 1\}$ and $bad(e) \in \{Yes, No\}$. As a result, on every edge e we obtain a sequence of bits, that will be denoted by $BitsSeq(e)$. In each iteration, we add (concatenate) a new bit to $BitsSeq(e)$ increasing its length by one. During the execution of the algorithm some of the edges will be excluded from further considerations. Such edges will be marked by concatenating letter "N" to the end of $BitsSeq(e)$. When FINDSPANNERS quits then the sequences $BitsSeq$ define a family of spanners. If the sequence $BitsSeq(e)$ does not end with "N" then $BitsSeq(e)$ is the number of a spanner that contains edge e . If the sequence $BitsSeq(e)$ ends with "N" then e does not belong to any spanner of the family. By $S_{\langle i_1, \dots, i_k \rangle}$, where $i_j \in \{0, 1\}$, we denote a subgraph of D -block $H = (A, B, E)$ induced by these edges e for which $BitsSeq(e)$ starts with a sequence (i_1, \dots, i_k) . By $S_{\langle \rangle}$ we denote the whole block H . Let $N_J(v)$ denote the set of neighbors of v in a graph J and let $d_J(v) = |N_J(v)|$.

PROCEDURE FINDSPANNERS

1. For $j := 0, \dots, \log D - 3$ do:
 - In parallel, for every subgraph $J := S_{\langle i_1, \dots, i_j \rangle}$, where $\langle i_1, \dots, i_j \rangle$ is an arbitrary sequence of bits, do:
 - Invoke procedure SPLITTER in J , which determines two functions:
 - $bit : E(J) \mapsto \{0, 1\}$
 - $bad : E(J) \mapsto \{Yes, No\}$
 - In parallel, for every $v \in A$, do:
 - If the number of edges $\{v, u\}$, $u \in N_J(v)$, such that $bad(v, u) = Yes$ is larger than $d_J(v)/\log n$, then for every $u \in N_J(v)$, do:
 - $BitsSeq(v, u) := BitsSeq(v, u) \circ "N"$,
 - else, for every $u \in N_J(v)$, do:
 - $BitsSeq(v, u) := BitsSeq(v, u) \circ bit(v, u)$.
2. Let $j := \log D - 4$. In parallel, for every subgraph $J := S_{\langle i_1, \dots, i_j \rangle}$, where $\langle i_1, \dots, i_j \rangle$ is an arbitrary sequence of bits, do:
 - For every vertex $v \in V(J) \cap A$ do:
 - If $d_J(v) \geq 2$ then change the j th bit in all but one edges incident to v to "N".
3. The result of this procedure is a set of subgraphs $S_{\langle i_1, \dots, i_{\log D - 4} \rangle}$, where $\langle i_1, \dots, i_{\log D - 4} \rangle$ is an arbitrary sequence of bits. Every such subgraph is a spanner.

Before we describe procedure SPLITTER let us define a *vertex-splitting* operation. For a vertex v let e_0, \dots, e_{k-1} denote edges incident to v . If k is even then we replace v by vertices $v_0, \dots, v_{\lfloor k/2 \rfloor - 1}$, where each v_i has degree two. If k is odd then in addition we add one vertex $v_{\lfloor k/2 \rfloor}$ of degree one. Then, for $i \leq \lfloor k/2 \rfloor - 1$, edges e_{2i}, e_{2i+1} are incident to v_i and if k is odd then e_{k-1} is incident to $v_{\lfloor k/2 \rfloor}$.

After splitting the vertices we obtain a graph of maximum degree two, that is a union of paths and cycles. Some of the vertices on a cycle (or path) will be marked as *border* vertices. Paths that connect two consecutive border vertices are called segments. Also the path-segments that connect an endpoint of a path with its closest border vertex are called segments.

In the procedure SPLITTER, given below, we will use procedures from [HKP99] to partition paths and cycles into segments of length at least $\ell := 400 \log^2 n$, in addition, marked by an alternating sequence of bits 0 and 1. Using the algorithms from [HKP99] such a partition can be found distributively in time $O(\log^2 n)$.

PROCEDURE SPLITTER(J)

1. Split every vertex of a graph J into groups of vertices of degree two (and possibly one of degree one) to obtain a union of paths and cycles.
2. Using procedures from [HKP99] (see procedures LONGARROWS and SPLITTER) find in time $O(\log^2 n)$ segments of length which is greater than or equal to $\ell := 400 \log^2 n$. In addition the procedures from [HKP99] assigns bits 0 and 1 to edges of a segment, so that edges that are adjacent in the segment have different bits assigned to them.
3. For every vertex w in the union of paths and cycles that corresponds to some vertex $v \in V(J) \cap B$ in the original graph, do: If w is a border vertex and both edges incident to w have the same bit assigned then flip the value of one of them (i.e., $0 \mapsto 1$ or $1 \mapsto 0$).
4. As a result two functions are obtained:
 $bit : E(J) \mapsto \{1, 0\}$
 $bad : E(J) \mapsto \{Yes, No\}$,
 where $bit(e)$ is equal to the bit assigned to edge e in step 2.
 If e is incident to a border vertex then put $bad(e) = Yes$, else $bad(e) = No$.
 (In particular, if e is an endpoint of a path then $bad(e) = No$).

An edge e is called *bad* if $bad(e) = Yes$. If edge is not bad then it is called *good*.

A vertex $v \in V(J) \cap A$ is called *nasty* if there are more than $d_J(v) / \log n$ bad edges incident to v . Otherwise a vertex is called *pliable*. Observe that if an edge e is incident to a nasty vertex then FINDSPANNERS assigns "N" to e .

Let $\langle i_1, \dots, i_j, i_{j+1} \rangle$ be a sequence of bits (i.e. $i_k \in \{0, 1\}$). Denote by $J := S_{\langle i_1, \dots, i_j \rangle}$, $d_j(v) := d_J(v)$, and $d_{j+1}(v) := d_{S_{\langle i_1, \dots, i_j, i_{j+1} \rangle}}(v)$. In addition, let $P_j := V(J) \cap A$ and $P_{j+1} := V(S_{\langle i_1, \dots, i_j, i_{j+1} \rangle}) \cap A$. Note that P_{j+1} is the set of pliable vertices in graph J obtained by SPLITTER in the j th iteration of FINDSPANNER.

Lemma 1. For every vertex $v \in P_{j+1}$,

$$\frac{1}{2} \left(\left(1 - \frac{2}{\log n} \right) d_j(v) - 1 \right) \leq d_{j+1}(v) \leq \frac{1}{2} \left(\left(1 + \frac{2}{\log n} \right) d_j(v) + 1 \right), \quad (1)$$

and for every $v \in B$,

$$d_{j+1}(v) \leq \frac{1}{2} (d_j(v) + 1). \quad (2)$$

Proof. Let e_+ and e_- denote the number of good and bad vertices (respectively) incident to v . Of course $d_j(v) = e_+ + e_-$. First we prove (1). Let $v \in P_{j+1}$ and let v_1, \dots, v_k denote vertices obtained from v after vertex-splitting operation. Then, since v is pliable, $e_- \leq d_j(v)/\log n$. To obtain the upper bound for $d_{j+1}(v)$, observe that the worst case will arise if the following conditions are satisfied: there is no vertex v_i which is incident to two bad edges, vertices from $\{v_1, \dots, v_k\}$ that are incident to one bad edge have on both edges incident to them bit i_{j+1} , $d_j(v)$ is odd, and the bit on the edge incident to v_k (vertex of degree one) is i_{j+1} . Consequently,

$$d_{j+1}(v) \leq 2e_- + (e_+ - e_- - 1)/2 + 1 \leq \frac{1}{2} \left(\left(1 + \frac{2}{\log n}\right) d_j(v) + 1 \right).$$

To obtain a lower bound for $d_{j+1}(v)$, notice that the worst scenario for this case will arise if the following conditions are satisfied: there is no vertex v_i that is incident to two bad edges, vertices from $\{v_1, \dots, v_k\}$ that are incident to one bad edge have on both edges incident to them bit $(1 - i_{j+1})$, $d_j(v)$ is odd, and the bit on the edge incident to v_k is $(1 - i_{j+1})$. Then,

$$d_{j+1}(v) \geq (e_+ - e_- - 1)/2 \geq \frac{1}{2} \left(\left(1 - \frac{2}{\log n}\right) d_j(v) - 1 \right).$$

Inequality (2) follows from step 3 of procedure SPLITTER.

Lemma 2. *In the j th iteration of procedure FINDSPANNERS ($j = 0, \dots, \log D - 3$), the following condition is satisfied:*

$$|P_{j+1}| \geq |P_j| \left(1 - \frac{1}{200 \log n} \frac{\Delta_j}{\delta_j}\right),$$

where, $\Delta_j = \max\{d_j(v) : v \in P_j\}$ and $\delta_j = \min\{d_j(v) : v \in P_j\}$.

Proof. Proof is the same as the proof of a similar lemma in [HKP99]. Recall that P_{j+1} is the set of pliable vertices obtained after execution of SPLITTER in the j th iteration of procedure FINDSPANNERS. Let N_{j+1} be the set of nasty vertices and let $be[N_{j+1}]$ denote the number of bad edges incident to vertices from N_{j+1} . Observe that $|P_{j+1}| + |N_{j+1}| = |P_j|$. First we establish a lower bound for $be[N_{j+1}]$. Note that if $v \in N_{j+1}$ then there are at least $\delta_j/\log n$ bad edges incident to v . Therefore,

$$be[N_{j+1}] \geq |N_{j+1}| \frac{\delta_j}{\log n}.$$

Observe that $be[N_{j+1}]$ can be bounded from above by the number of all bad edges in graph J . Since $|E(J)| \leq \Delta_j |P_j|$ and every segment has length at least $\ell = 400 \log^2 n$ and contains at most two bad edges,

$$be[N_{j+1}] \leq \frac{2|P_j|\Delta_j}{400 \log^2 n}.$$

Combining last two inequalities yields

$$|N_{j+1}| \leq \frac{1}{200 \log n} \frac{\Delta_j}{\delta_j} |P_j|$$

and the lemma follows.

Theorem 1. *Let $H = (A, B, E)$ be a D -block. Procedure FINDSPANNERS finds in $O(\log^3 n)$ rounds a family of $\frac{1}{16}D$, $(\frac{1}{2}, 16)$ -spanners of H .*

Proof. There are $O(\log D)$ iterations in procedure FINDSPANNERS. In each iteration, main computations are performed by SPLITTER which runs in $O(\log^2 n)$ rounds. Thus the number of rounds is $O(\log^3 n)$.

Recall that $d_j(v) := d_{S_{\langle i_1, \dots, i_j \rangle}}(v)$, $P_j := V(S_{\langle i_1, \dots, i_j \rangle}) \cap A$. Let $k := \log D - 4$. We will show that for every sequence of bits $\langle i_1, \dots, i_k \rangle$, graph $S := S_{\langle i_1, \dots, i_k \rangle}$ is a $(\frac{1}{2}, 16)$ -spanner. Hence, we have to verify that S satisfies the following three conditions.

- $|P_k| > \frac{1}{2}|P_0|$
- $\forall v \in P_k : d_k(v) = 1$
- $\forall v \in B : d_k(v) < 16d_0(v)\frac{1}{D} + 1$

First we observe that the third condition is satisfied. By (2) in Lemma 1, we see that for every $v \in B$,

$$d_k(v) \leq \left(\frac{1}{2}\right)^k (d_0(v) - 1) + 1$$

which shows that the third condition is satisfied. Applying Lemma 1 (part (1)) k times shows that for every $v \in P_k$,

$$q_-^k (d_0(v) + 1) - 1 \leq d_k(v) \leq q_+^k (d_0(v) - 1) + 1$$

where

$$q_- = \frac{1}{2} \left(1 - \frac{2}{\log n}\right), \quad q_+ = \frac{1}{2} \left(1 + \frac{2}{\log n}\right)$$

Since $\forall v \in A : D/2 < d_0(v) \leq D$, we have that for every $v \in P_k$,

$$q_-^k \left(\frac{D}{2} + 1\right) - 1 \leq d_k(v) \leq q_+^k (D - 1) + 1.$$

So, for sufficiently large n ,

$$q_-^k \left(\frac{D}{2} + 1 \right) - 1 > 8e^{-2(1 + \frac{2}{\log n})} - 1 > 0$$

and

$$q_+^k (D - 1) + 1 < 16e^2 + 1 < 120.$$

Thus for every $v \in P_k$, $d_k(v) > 0$. Since in step 2 of FINDSPANNERS we disregard all but one edge incident to v , the second condition is satisfied. Finally we verify that the first condition is satisfied as well. Indeed, apply Lemma 2 k times. Then

$$|P_k| \geq |P_0| \prod_{j=0}^{k-1} \left(1 - \frac{1}{200 \log n} \frac{\Delta_j}{\delta_j} \right).$$

Note that

$$\frac{\Delta_j}{\delta_j} \leq \frac{q_+^j (D - 1) + 1}{q_-^j (D/2 + 1) - 1} \leq 120$$

since the fraction in the middle is an increasing function of j for $j \leq k$. Therefore, for n sufficiently large, we have

$$|P_k| \geq |P_0| \left(1 - \frac{3}{5 \log n} \right)^{\log n} \geq \frac{1}{2} |P_0|.$$

3 Algorithm

In this section, we present and analyze our edge-coloring algorithm. The algorithm runs in $O(\log^4 n)$ rounds and uses $O(\Delta \log n)$ colors. The algorithm can be divided into four smaller procedures: FINDSPANNERS, COLORSPANNER, COLORBIPARTITE, COLORGRAPH. Procedure FINDSPANNERS finds a family of edge-disjoint spanners of a block and was presented in a previous section. Let $H = (A, B, E)$ be a D -block, $m = |A| + |B|$, $\alpha = \frac{1}{2}$ and $\beta = 16$. Then, by Theorem 1, FINDSPANNERS finds in $O(\log^3 m)$ rounds a family of D/β edge-disjoint (α, β) -spanners. First, we describe a procedure that colors a family of spanners found by FINDSPANNERS. Note that a single spanner is simply a collection of vertex-disjoint stars, with centers of stars in set B , and so vertices of B (in parallel) can color edges of the spanner provided they have enough colors available. We need some additional notation. Let

$$K := \frac{\beta}{D} \Delta + 1,$$

and for $j := 1, \dots, D/\beta$ let

$$I_j := \{(j-1)K, \dots, jK-1\}.$$

PROCEDURE COLORSPANNER

1. Find a collection of spanners $S_1, \dots, S_{D/\beta}$ using FINDSPANNERS.
2. In parallel for every vertex $b \in B$ do:
 - For every $j := 1, \dots, D/\beta$ do:
 - Color edges of S_j incident to b using colors from I_j .

Lemma 3. *Let $H = (A, B, E)$ be a D -block, $m = |A| + |B|$. Procedure COLORSPANNER colors at least $\frac{\alpha}{\beta}|E|$ edges of H using $O(\Delta)$ colors and runs in $O(\log^3 m)$ rounds .*

Proof. Indeed, since every (α, β) -spanner contains at least $\alpha|A|$ vertices of degree one in A , the number of edges in the family of spanners is at least $\alpha|A|\frac{1}{\beta}D \geq \frac{\alpha}{\beta}|E|$. Also, for every $j = 1, \dots, D/\beta$ and every vertex $b \in B$, $\deg_{S_j}(b) \leq K = |I_j|$ and so edges within one spanner are colored properly. Since we use different colors to color different spanners, coloring obtained by COLORSPANNER is proper. Hence $O(\Delta)$ colors is suffices, because at each vertex of B we need $\frac{1}{\beta}DK$ colors and $D \leq \Delta$.

We can now consider bipartite graphs. Let $G = (L, R, E)$ be a bipartite graph with $|L| = |R| = n$. For $e = \{u, v\} \in E$ with $u \in L, v \in R$ define a weight of e as

$$\omega(e) := \deg_G(u) \tag{3}$$

and let

$$\omega(\bar{E}) := \sum_{e \in \bar{E}} \omega(e) \tag{4}$$

be a total weight of a set $\bar{E} \subset E$. Trivially, we have

$$\omega(E) \leq n^3.$$

Let $D_i := \Delta/2^i$, where $i = 0, \dots, \log \Delta$, and consider D_i -blocks $H_i = (A_i, B, E_i)$ where

$$A_i := \{u \in L : \frac{D_i}{2} < \deg_G(u) \leq D_i\}, \quad B := R,$$

and E_i contains all edges of G that have an endpoint in A_i .

PROCEDURE COLORBIPARTITE

1. In parallel, for $i := 0, \dots, \log \Delta$, color every H_i using COLORSPANNER.
2. In parallel for every vertex $v \in R$ do:
 - For every color c do:
 - (a) Define $E_{c,v}$ to be the set of edges incident to v that are colored with color c .
 - (b) Find in $E_{c,v}$ an edge e with maximum weight and uncolor all edges in $E_{c,v}$ except e .

Lemma 4. *Let $G = (L, R, E)$ be a bipartite graph with $n = |L| = |R|$. Procedure COLORBIPARTITE properly colors a set $\bar{E} \subset E$ such that $\omega(\bar{E}) \geq \alpha\omega(E)/(6\beta)$ using $O(\Delta)$ colors and runs in $O(\log^3 n)$ rounds.*

Proof. Sets A_i are disjoint and so after step 1 of the algorithm, there will be no edges of the same color that are incident to vertices of L . After step 2, every vertex $v \in R$ has at most one edge of a given color incident to it. Let $\bar{E}_i \subset E_i$ denote the set of edges of the i th block that are colored in step 1. From Lemma 3, $|\bar{E}_i| \geq \alpha|E_i|/\beta$, and since the minimum degree in H_i is larger than $D_i/2$, we have $\omega(\bar{E}_i) \geq \alpha|E_i|D_i/(2\beta)$. Consequently

$$\omega(\bar{E}_0 \cup \dots \cup \bar{E}_{\log \Delta}) \geq \frac{\alpha}{2\beta} \sum_i |E_i|D_i \geq \frac{\alpha}{2\beta} \omega(E).$$

We need to argue that at least one third of this weight is maintained after "correction" done in the second step of the procedure. Fix a vertex $v \in R$ and color c . Let $M := M(v, c)$ denote the maximum weight of an edge incident to v in color c . Since edges incident to v that have the same color must belong to different blocks, weight of edges incident to v that have color c is less than

$$M + \sum_{i \geq 0} \frac{M}{2^i} < 3M.$$

Therefore, the total weight of edges that remain colored is at least $\frac{\alpha}{6\beta} \omega(E)$.

Before we proceed, we need one additional concept. Let $P = (V, E)$ be a path or cycle on at least two vertices. A partition V_1, \dots, V_t of V is called *short* if the following conditions are satisfied.

1. Every graph $P_i = P[V_i]$ induced by V_i is a path or cycle.
2. For every $i = 1, \dots, t$, $2 \leq |V_i| = O(\log |V|)$.
3. For every $i = 1, \dots, t-1$, $|V_i \cap V_{i+1}| = 1$. If P is a cycle than $|V_0 \cap V_t| = 1$.

Vertices that belong to $V_i \cap V_{i+1}$ for some i are called border vertices. In [AGLP89], the following fact is proved.

Lemma 5. *Let $P = (V, E)$ be a path or cycle with $|V| \geq 2$. There is a procedure that finds a short partition of P in $O(\log |V|)$ rounds.*

Let us now consider an arbitrary graph $G = (V, E)$. Our first task is to obtain an auxiliary bipartite graph from G which will be colored using COLORBIPARTITE. Define a bipartite graph $G' := (L, R, E')$ as follows. Every vertex splits itself into two vertices $(v, 0)$ and $(v, 1)$. Let $L := \{(v, 0) : v \in V\}$, $R := \{(v, 1) : v \in V\}$ and $\{(u, 0), (v, 1)\} \in E'$ if and only if $u < v$ and $\{u, v\} \in E$.

PROCEDURE COLORGRAPH

1. Obtain $G' = (L, R, E')$ from $G = (V, E)$ as described above.
2. Apply COLORBIPARTITE to G' .
3. Obtain a coloring of G by merging vertices $(v, 0)$, $(v, 1)$ that correspond to a single vertex $v \in V$. (This operation can result in some monochromatic paths and cycles.)

4. In parallel for every monochromatic path and cycle P do:
 - (a) Find a short partition of P into segments P_1, \dots, P_t .
 - (b) In parallel for every segment P_i do:
 - Find a matching M_1 that saturates all but at most one vertex in P_i and let $M_2 := E(P_i) \setminus M_1$.
 - If $\omega(M_1) \geq \omega(M_2)$ then uncolor edges of M_2 . Otherwise uncolor edges of M_1 .
 - (c) In parallel for every border vertex v do:
 - If both edges incident to v have the same color then uncolor an edge with a smaller weight.

A weight $\omega(\{u, v\})$ of an edge $\{u, v\} \in E$ with $u < v$ is defined as the number of vertices $w \in V$ such that $u < w$ and $\{u, w\} \in E$. Thus the weight of $\{u, v\}$ is exactly equal to the weight of edge $\{(u, 0), (v, 1)\} \in E'$. In the same way as in (4) we can define $\omega(\bar{E})$ for any set $\bar{E} \subset E$.

Lemma 6. *Let $G = (V, E)$ be a graph with $n = |V|$. Procedure COLORGRAPH colors a set $\bar{E} \subset E$ such that $\omega(\bar{E}) \geq \alpha\omega(E)/(24\beta)$ using $O(\Delta)$ colors and runs in $O(\log^3 n)$ rounds.*

Proof. From Lemma 4, we know that after step 2 we colored a set $\bar{E}' \subset E'$ such that $\omega(\bar{E}') \geq \alpha\omega(E')/(6\beta)$. Thus the total weight of edges that remain colored after step 4(b) is at least $\omega(\bar{E}')/2$. Finally, the total weight of edges that remain colored after step 4(c) is at least $\omega(\bar{E}')/4$. Since $\omega(E') = \omega(E)$, procedure COLORGRAPH colors a set \bar{E} such that $\omega(\bar{E}) \geq \alpha\omega(E)/(24\beta)$.

Iterating COLORGRAPH $O(\log n)$ times yields a proper coloring of graph $G = (V, E)$.

PROCEDURE COLOR

1. Run $O(\log n)$ times:
 - Use procedure COLORGRAPH with a palette of $O(\Delta)$ new colors (different than previously used colors) to properly color set $\bar{E} \subset E$.
 - Let $E := E \setminus \bar{E}$.

Theorem 2. *Let $G = (V, E)$ be a graph with $n = |V|$. Procedure COLOR properly colors edges of G using $O(\Delta \log n)$ colors and runs in $O(\log^4 n)$ rounds.*

Proof. By Lemma 6 in each iteration of the procedure \bar{E} is properly colored using $O(\Delta)$ colors. Since COLOR uses different colors in each iteration, obtained coloring is proper.

To complete the proof, we show that E is empty after $O(\log n)$ iterations of the main loop in COLOR. Notice that $E \neq \emptyset$ is equivalent to $\omega(E) \geq 1$. Let $p = (1 - \frac{\alpha}{\beta} \frac{1}{24})^{-1}$ and let ω denote the weight of the edge set of graph G . By Lemma 6, the total weight of edges left after the k th iteration is at most ω/p^k . Since $\omega \leq n^3$, the right hand side of the last inequality is less than one if $k > \frac{3 \log n}{\log p}$.

4 Acknowledgement

This work was supported by KBN GRANT 7 T11C 032 20.

References

- [Li92] N. Linial, *Locality in distributed graph algorithms*, SIAM Journal on Computing, 1992, 21(1), pp. 193-201.
- [MaPe01] G. De Marco, A. Pelc, *Fast distributed graph coloring with $O(\Delta)$ colors*, SODA 2001.
- [HKP99] M. Hańćkowiak, M. Karoński, A. Panconesi, *A faster distributed algorithm for computing maximal matching deterministically*, Proceedings of PODC 99, the Eighteen Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, pp. 219-228.
- [AGLP89] B. Awerbuch, A.V. Goldberg, M. Luby, and S. Plotkin, *Network decomposition and locality in distributed computing*, Proceedings of the 30th Symposium on Foundations of Computer Science (FOCS 1989), pp. 364-369.