Parallel Approximate Edge Coloring Revisited

Weifa Liang[1,2] George Havas[1] Anne Street[2]

[1] School of Information Technology
The University of Queensland, Brisbane, QLD 4072, Australia
[2] Department of Mathematics
The University of Queensland, Brisbane, QLD 4072, Australia

Abstract. Let $G = (V, E)$ be a simple graph with n vertices and m edges and let its maximum vertex degree be Δ. The edge coloring problem for G is to assign colors to the edges of G such that all edges sharing a common vertex have different colors. The objective is to use as few colors as possible to color G.

This paper presents a fast NC algorithm for edge coloring G with $\Delta + q/k + 1$ colors, where q is the number of subgraphs containing odd cycles in Π, k is a fixed integer with $k \geq 1$, $q \leq \lceil \Delta/2 \rceil$, and $\Pi = \{G_1, G_2, \ldots, G_{\lceil \Delta/2 \rceil}\}$ is a partition of $G(V, E)$ such that $\cup_{i=1}^{\lceil \Delta/2 \rceil} G_i = G$ and the maximum vertex degree of each G_i is at most 2. The algorithm runs in $O((2k)^7 \log^2 n + \log^2 n \log \Delta)$ time using $O(m + n)$ processors on a CREW PRAM. In particular, when G contains C odd cycles and C is a constant, it follows that edge coloring G with at most $\Delta + q/2 + 1$ colors is in NC, and $q \leq C/2$. By generalizing the above algorithm, we obtain a parallel algorithm for edge coloring G with $\Delta + \lfloor \Delta^\alpha/2 \rfloor + 3$ colors. The proposed algorithm requires $O(T(n, 2\Delta^{1-\alpha}) + \log^2 n \log \Delta)$ time using $O(\Delta^\alpha P(n, 2\Delta^{1-\alpha}))$ processors on a PRAM, where $T(n, \Delta)$ is the time complexity of the fastest known parallel algorithm to edge-color a graph of n vertices with $\Delta + 1$ colors using $P(n, \Delta)$ processors on a PRAM, where $0 < \alpha < 1$. Compared with an earlier result, our algorithm runs as fast as that algorithm with the same number of processors in the big O sense, but reduces the number of colors by an additive term $3\Delta^\alpha/2 - 3$. The parallel computational models used are the CREW PRAM in which concurrent read is allowed but concurrent write is forbidden, and the CRCW PRAM in which both concurrent read and concurrent write are allowed.

Keywords: *NC algorithms, approximate edge coloring, graph problems.*

1 Introduction

Given a simple graph $G(V, E)$, where $|V| = n$, $|E| = m$, the *edge coloring* problem is to assign each edge of G a color such that all edges sharing a common vertex have different colors. We wish to use as few colors as possible to color G. The minimum number of colors used is called *chromatic index*, denote by $\chi'(G)$. Let $\Delta(G)$ be the maximum vertex degree of G (Δ for short); obviously $\chi'(G) \geq \Delta$. Vizing [19] showed that $\Delta \leq \chi'(G) \leq \Delta + 1$. In fact, his proof implies an $O(nm)$ sequential time algorithm. However, Holyer [12] showed that

deciding whether a graph requires Δ or $\Delta+1$ colors is NP-complete, even when restricted to the class of cubic graphs.

The edge coloring problem appears in many places (e.g. [4, 9, 15]) and is an interesting graph-theoretic problem. There are a number of NC algorithms for the edge coloring problem on restricted graphs such as planar graphs [2, 3], bipartite graphs [16], and Halin graphs [6]. For general graphs, the pioneering work is due to Karloff and Shmoys [14], who presented a parallel edge coloring algorithm with $\Delta+1$ colors. Their algorithm requires $O(\Delta^6 \log^4 n)$ time and $O(n^2\Delta)$ processors, provided the fastest known NC algorithm for finding a maximal independent set is employed [8]. Following the idea of Karloff and Shmoys and using some new techniques, Liang et al. [18] improved Karloff and Shmoys' algorithm by an $O(\Delta^{2.5})$ factor in time, using $O(n^2+n\Delta^3)$ processors on the same model. However, deciding whether the edge coloring problem with $\Delta+1$ colors is in NC is still an important open problem.

In this paper we study the *approximate edge coloring* problem in the parallel environment, that is, we use more than $\Delta+1$ colors to color G such that all edges sharing a common vertex have different colors. Obviously this is a relaxed version of the classic edge coloring problem. For this problem, a number of deterministic and randomized NC algorithms for edge coloring G with $\lceil c\Delta \rceil$ colors, $c>1$, have been suggested. For example, Upfal uses $3\lceil \Delta/2 \rceil$ colors to edge color G. His algorithm needs $O(\log^3 n\Delta)$ time using $O(n\Delta)$ processors [14] on a CRCW PRAM. Liang [17] uses 2.5Δ colors to edge color G. His algorithm requires $O(\log n \log \Delta)$ time using $O(m+n)$ processors on a CREW PRAM. Fürer and Raghavachari [5] use $max\{\lceil c\Delta \rceil, \Delta+1\}$ colors to edge color G, where $c>1$ is fixed. Their algorithm needs $O(d^7 \log^2 n)$ time and $O(m+n)$ processors on a CREW PRAM, where $d=\lceil \frac{c+1}{c-1} \rceil$. Recently Grable and Panconesi [10] proposed a randomized NC algorithm with $max\{(1+\epsilon)\Delta, \Delta+1\}$ colors for any $\epsilon>0$. Their algorithm requires $O(\log\log n)$ time using $O(m\Delta)$ processors on a CRCW PRAM, with high probability. Furthermore, a few very fast NC algorithms have been developed if we are allowed to edge color G with Δ^2 colors. Thus, Fürer and Raghavachari [5] presented an NC algorithm with Δ^2 colors, which requires $O(\log^* n)$ time using $O(m+n)$ processors on a CRCW PRAM. Recently Han et al. [11] further improved the result of Fürer and Raghavachari by showing that edge coloring G with $\lceil \frac{\Delta^2}{\log^{c'}(\log^* n)} \rceil$ colors can be done in $O(\log\log^* n)$ time using $O(m+n)$ processors on the same model, where c' is a large constant.

Now we may ask how fast it can be done if we use $\Delta + c\Delta^\alpha$ colors to edge-color G in parallel, where c is a constant and $0<\alpha<1$. Karloff and Shmoys [14] presented a randomized NC algorithm for this problem. Their algorithm uses $\Delta+20\Delta^{1/2+\epsilon'}$ colors, and runs in $O(\log^{O(1)} n)$ expected time using $O(n^{O(1)})$ processors on a CRCW PRAM, where $\epsilon' \le 1/4$. However, the deterministic simulation of their randomized algorithm needs $O(n^{O(\log n/\log\log n)})$ processors [14], so their algorithm is not in NC. Based on balanced degree decomposition (see Section 2) and incorporating the fastest known parallel algorithm with $\Delta+1$ colors [18], Liang [17] presented a parallel algorithm for the problem. His algorithm uses between $\Delta+2\lceil \Delta^\alpha \rceil$ and $\Delta+3\lceil \Delta^\alpha \rceil$ colors, and runs

in $O(T(n, \Delta^{1-\alpha}) + \log n \log \Delta)$ time using $O(\Delta^\alpha P(n, \Delta^{1-\alpha}))$ processors, where $0 < \alpha < 1$, $T(n, \Delta)$ is the time complexity of the fastest known parallel algorithm to edge-color a graph of maximum vertex degree Δ with $\Delta+1$ colors on a PRAM, and $P(n, \Delta)$ is the number of processors used for that coloring. Currently $T(n, \Delta) = O(\Delta^{3.5} \log^3 \Delta \log n + \Delta^3 \log^4 n)$ and $P(n, \Delta) = O(n^2 \log \Delta + n\Delta^3)$ by [18].

In this paper we present a fast NC algorithm for edge coloring G with $\Delta + q/k + 1$ colors, where q is the number of subgraphs containing odd cycles in Π, and $k \geq 1$ is a fixed integer. Here $q \leq \lceil \Delta/2 \rceil$, and $\Pi = \{G_1, G_2, \ldots, G_{\lceil \Delta/2 \rceil}\}$ is a partition of $G(V, E)$ such that $\cup_{i=1}^{\lceil \Delta/2 \rceil} G_i = G$ and the maximum vertex degree of each G_i is at most 2. The algorithm runs in $O((2k)^7 \log^2 n + \log^2 n \log \Delta)$ time using $O(m + n)$ processors on a CREW PRAM. In particular, when G contains C odd cycles and C is a constant, this shows that edge coloring G with at most $\Delta + q/2 + 1$ colors is in NC, and $q \leq C/2$. By assigning $k = \lceil \Delta^{1-\alpha} \rceil$ in the algorithm above, we obtain a parallel algorithm for edge coloring G with $\Delta + \lfloor \Delta^\alpha/2 \rfloor + 3$ colors. The algorithm requires $O(T(n, 2\Delta^{1-\alpha}) + \log^2 n \log \Delta)$ time using $O(\Delta^\alpha P(n, 2\Delta^{1-\alpha}))$ processors on a PRAM, where $0 < \alpha < 1$. Compared with the result of [17], this algorithm runs as fast as that algorithm in the big O sense, but reduces the number of colors by an additive term $3\Delta^\alpha/2 - 3$, provided that the same number of processors are available.

The remainder of this paper is organized as follows. In Section 2 we introduce two different graph decomposition concepts. In Section 3 we analyze an NC algorithm to edge color a constant-degree graph with $\Delta + 1$ colors, assuming that the graph has maximum vertex degree Δ. The parallel algorithms for approximate edge coloring are given in Section 4. We conclude this paper in Section 5 by introducing an open problem.

2 Preliminaries

2.1 The balanced degree decomposition

Given an Eulerian graph G, the *Eulerian decomposition* is a partition of the graph G into edge-disjoint simple cycles. If G is not an Eulerian graph, we add a new vertex v to it, and add an edge between v and each of the odd-degree vertices of G so that the resulting graph G' is an Eulerian graph. Let G (or G') be an Eulerian graph. We first partition G into two edge-disjoint subgraphs G_1 and G_2 such that $\Delta(G_1) = \Delta(G_2)$. This can be easily done by traversing an Euler tour and alternately coloring every edge *red* and *blue* during the Eulerian traversal. Let the subgraph induced by all red edges be G_1, and the subgraph induced by all blue edges be G_2. As a result, $\Delta(G_1) = \Delta(G_2)$ and $|E(G_1)| = |E(G_2)|$. Obviously, for an arbitrary simple graph G, G can be partitioned into two edge-disjoint subgraphs $G_1(V, E_1)$ and $G_2(V, E_2)$ such that

$$\Delta(G_1) \leq \lceil \Delta(G)/2 \rceil \text{ and } \Delta(G_2) \leq \lfloor \Delta(G)/2 \rfloor + 1.$$

We call the partition above a *balanced degree decomposition*. Liang [17] and Fürer and Raghavachari [5] independently proved the following lemma.

Lemma 1. *Given a graph $G(V, E)$, partitioning G into two edge-disjoint subgraphs $G_i(V, E_i)$, $i = 1, 2$, such that $\Delta(G_1) = \lceil \Delta(G)/2 \rceil$, $\Delta(G_2) = \lfloor \Delta(G)/2 \rfloor + 1$, $E_i \subseteq E$, $E_1 \cap E_2 = \emptyset$, $i = 1, 2$, can be done in $O(m + n)$ sequential time, or in $O(\log n)$ time using $O(m + n)$ processors on a CREW PRAM.*

Based on Lemma 1, Liang [17] and Fürer and Raghavachari [5] developed NC algorithms for approximate edge coloring. However, the balanced degree decomposition seems to have a limitation. For example, suppose we use $\Delta + c\Delta^\alpha$ colors to edge color G with fixed c, then run this decomposition routine recursively, and stop at a point that the current graph has maximum vertex degree $\lfloor \Delta^\alpha \rfloor$. Finally we edge color each of these subgraphs with $\lfloor \Delta^\alpha \rfloor + 1$ colors in parallel. It is not difficult to see that in doing so at least $2^{\log \Delta^\alpha} = \Delta^\alpha$ extra colors are needed. In order to remove these extra colors, we introduce another graph decomposition Π as follows.

2.2 Degree-2 decomposition

Now we define another graph decomposition Π, which we call *degree-2 decomposition*. Let $\Pi = \{G_1, G_2, \ldots, G_{\lceil \Delta/2 \rceil}\}$ be a partition of $G(V, E)$ such that $\cup_{i=1}^{\lceil \Delta/2 \rceil} G_i = G$ and the maximum vertex degree of each G_i is at most 2. Π can be obtained by Upfal's algorithm [14] which is introduced in Section 4.

Assume that Π is given. Then, for each subgraph G_i in Π, the connected components of G_i are either simple cycles or simple paths because $\Delta(G_i) \leq 2$. If G_i does not contain any odd-cycles, then it can be edge colored by 2 colors. Otherwise, it can be edge colored by 3 colors. Therefore, we have

Lemma 2. *Each G_i in Π can be edge colored in $O(\log n)$ time using $O(n)$ processors on a CREW PRAM, where $1 \leq i \leq \lceil \Delta/2 \rceil$. The number of colors used is 2 or 3, depending on whether or not G_i contains odd-cycles.*

Proof. It is well known that a linked list or cycle of size n with odd-length can be edge colored with 3 colors in $O(\log n)$ time using $O(n)$ processors on an EREW PRAM [16], and a linked list or cycle of size n with even-length can be edge colored with 2 colors in $O(\log n)$ time using $O(n)$ processors on an EREW PRAM [13]. The lemma follows. □

3 Edge Coloring Constant-degree Graphs with Δ+1 Colors

In order to proceed, we make use of the following lemma.

Lemma 3. *[7] A maximal independent set (MIS) in a constant-degree graph G can be found in $O((\Delta^2 + \log^* n) \log \Delta)$ time using $O(m + n)$ processors on an EREW PRAM.*

Now applying Lemma 3 to the algorithm of Karloff and Shmoys [14], we have

Theorem 4. *Edge coloring a constant-degree graph G with $\Delta+1$ colors can be done in $O(\Delta^7 \log^2 n)$ time using $O(m+n)$ processors on a CREW PRAM.*

Proof. The algorithm of Karloff and Shmoys proceeds in $O(\Delta^5 \log n)$ stages. During each stage, to find an MIS of the current graph needs $O((\Delta^2 + \log^* n) \log \Delta)$ time using $O(m+n)$ processors by Lemma 3. Note that, for this special graph, we use the algorithm by Goldberg et al. [7] instead of the algorithm of Goldberg and Spencer [8] because the latter has high time complexity for this case. The construction of the auxiliary graph G^2 takes $O(\Delta)$ time using $O(m+n)$ processors. The chain-coloring needs $O(\log n)$ time using $O(m+n)$ processors. So edge coloring G with $\Delta+1$ colors takes $O(\Delta^7 \log^2 n)$ time. □

Let $f(.)$ and $g(.)$ be two positive functions, where $f(x) = x^7$ and $g(y) = \log^2 y$. Then the time complexity of Theorem 4 can be expressed as $O(f(\Delta)g(n))$. It is not hard to see that, if $g(n) > f(\Delta)$, i.e, $n > 2^{\Delta^{3.5}}$, $g(n)$ dominates the computational complexity of Theorem 4; otherwise $f(\Delta)$ dominates the computational complexity. For example, when $\Delta = 20$, $g(n) > f(\Delta)$ only if $n > 2^{20^{3.5}} > 2^{3200}$. So, in practice, if we consider edge coloring a constant-degree graph with $\lceil c\Delta \rceil$ colors and c is small, we see that the time complexity of Theorem 4 is dominated by $f(\Delta)$ rather than $g(n)$.

4 Approximate Edge Coloring in Parallel

Fürer and Raghavachari [5] presented an NC algorithm for edge coloring a simple graph $G(V,E)$ with $max\{\lceil c\Delta \rceil, \Delta+1\}$ colors, with fixed $c > 1$. The basic idea of their algorithm is as follows. It repeatedly "splits" the graph into two subgraphs of smaller degree. The edge set E is divided into two parts such that the degree of the subgraph induced by each part is roughly half the degree of G. Then it splits each subgraph recursively. This splitting process stops when the subgraph obtained has a constant degree d where $d \geq \lceil \frac{c+1}{c-1} \rceil$. Finally, it edge-colors each of these subgraphs with $d+1$ colors in parallel, and the color set used for each subgraph is different. So, the total number of colors used by G is $max\{\lceil c\Delta \rceil, \Delta+1\}$. Incorporating Theorem 4, their theorem becomes

Theorem 5. *[5] Let G be a graph with n vertices and m edges. It is possible to compute an edge coloring for G using at most $\lceil c\Delta \rceil$ $(\geq \Delta+1)$ colors for any constant $1 < c \leq 2$ in $O(d^7 \log^2 n)$ time using $O(m+n)$ processors on a CREW PRAM, where $d = \lceil \frac{c+1}{c-1} \rceil$.*

Let $T(c)$ be the running time of the algorithm of Fürer and Raghavachari. Obviously

$$T(c) \propto (\frac{c+1}{c-1})^7 \log^2 n \tag{1}$$

Now we rewrite c as $c = 1+\epsilon$, and $0 < \epsilon \leq 1$. Then, when $\epsilon \to 0$, $d = \lceil \frac{c+1}{c-1} \rceil = \lceil 1 + 2/\epsilon \rceil \to \infty$. In other words, if we use few colors (e.g. c=1.001) to edge color G, then $f(d)$ will dominate the time complexity of Theorem 5 in practice.

Though whether edge coloring G with $\Delta+1$ colors is in NC is still undecided, we already know that there is a trade-off between the number of colors used and the time spent for the coloring [17, 5]. Based on degree-2 decomposition of G, in the following we give a parallel algorithm to edge color G with $\Delta + c\Delta^{\alpha}$ colors, which exhibits a better trade-off between these two key terms (where $0 < \alpha < 1$ and c is fixed). The algorithm is as follows.

Algorithm: $COLOR\_EDGE(G, k)$ /* k is an integer parameter */

Step 1. Form a bipartite graph $G_B(X', Y', E_B)$ as follows, where $X' = \{ v' \mid v \in V\}$ and $Y' = \{ v'' \mid v \in V\}$. If G is not an Eulerian graph, then add a new vertex v that is adjacent to all vertices of odd degree. Find an Eulerian tour of the augmented graph, and view the tour as a directed tour using the algorithm due to Atallah and Vishkin [1]. For each directed edge $< u, v >$ traversed from u to v, add an undirected edge (u', v'') to E_B. As a result, G_B is a bipartite graph with maximum vertex degree $\lceil \Delta(G)/2 \rceil$.
Step 2. Color the edges of G_B with $\lceil \Delta(G)/2 \rceil$ colors, using the optimal algorithm of Lev et al [16].
Step 3. Let M_i ($\subseteq E_B$) be a set of edges colored α_i, where α_i is a color. Construct a subgraph $G_i(V(M_i), E_i)$ as follows: for each edge $(u', v'') \in M_i$, put an edge (u, v) into E_i. Since M_i is a matching of G_B, $\Delta(G_i) \leq 2$. Thus, G_i is a collection of disjoint paths and simple cycles. Now G is decomposed into subgraphs $G_1, G_2, \ldots, G_{\lceil \Delta/2 \rceil}$. Denote this partition of G by Π.
Step 4. For $i := 1$ to $\lceil \Delta/2 \rceil$ do in parallel
if G_i does not contain any odd-cycles, edge color G_i with two colors; otherwise do nothing.
Step 5. Let q subgraphs $G_i \in \Pi$ contain odd-cycles. Assign a new index to each of these subgraphs. Let $G'_1, G'_2, \ldots, G'_q$ be the new indexes of these subgraphs. Divide them into $\lceil q/k \rceil$ groups such that each group contains exactly k subgraphs, except the last group may contain r subgraphs, where $r \leq k$.
Step 6. Each group forms a graph which has maximum vertex degree $2k$, except the graph formed by the last group has maximum vertex degree $2r$. With $2k+1$ colors, edge color the graph formed by each group except the last, where the colors used for each group come from a different color set. Such a coloring is feasible by Vizing's Theorem. Edge color the graph formed by the last group with $2r + 1$ colors.

Note that Steps 1 - 3 used to find the partition Π are due to Upfal (in [14]).

Theorem 6. *Let G be a graph with n vertices and m edges. It is possible to compute an edge coloring for G using at most $2\lceil \Delta/2 \rceil + \lfloor q/k \rfloor + 1$ colors in $O(\log^2 n \log \Delta + (2k)^7 \log^2 n)$ time using $O(m+n)$ processors on a CREW PRAM, where $q \leq \lceil \Delta/2 \rceil$ is the number of subgraphs which contain odd-cycles in Π, Π is the partition obtained in Step 1 of algorithm $COLOR\_EDGE$, and $k \geq 1$ is a fixed integer.*

Proof. Assume that G is stored by double-linked adjacency lists. Step 1 takes $O(\log^2 n)$ time and $O(m+n)$ processors [1]. Step 2 takes $O(\log^2 n \log \Delta)$ time using $O(m+n)$ processors [16]. Steps 3 and 4 take $O(\log n)$ time and $O(m+n)$ processors. Step 5 spends $O(\log n)$ time using $O(n/\log n)$ processors by prefix computation [13]. Step 6 takes $O((2k)^7 \log^2 n)$ time using $O(m+n)$ processors by Theorem 4. Let p be the number of subgraphs not containing odd-cycles in Π. Assume that $q = q' + r$, where q' *mod* $k \equiv 0$ and $r < k$. Then, $p + q = \lceil \Delta/2 \rceil$, i.e, $p + q = p + q' + r$. Then, the number of colors used for edge coloring G by algorithm $COLOR\_EDGE$ is
$2p + (2k+1)q'/k + (2r+1) = 2(p + q' + r) + q'/k + 1 = 2\lceil \Delta/2 \rceil + q'/k + 1$
$\leq \Delta + \frac{\Delta}{2k} + 2 = (1 + \frac{1}{2k})\Delta + 2$. □

Without loss of generality, let $c = 1 + \frac{1}{2k}$. Since $k \geq 1$, $1 < c \leq 1.5$. Now rewriting Theorem 6 using the formula in Theorem 5, we have

Theorem 7. *Let G be a graph with n vertices and m edges. It is possible to compute an edge coloring for G using at most $max\{\lceil c\Delta \rceil, \Delta + 1\}$ colors for any constant $1 < c \leq 1.5$ in $O(d'^7 \log^2 n + \log^2 n \log \Delta)$ time using $O(m+n)$ processors on a CREW PRAM, where $d' = \lceil \frac{1}{c-1} \rceil$.*

From an asymptotic complexity point of view (i.e. $n \to \infty$), the result in Theorem 7 is inferior to the result in Theorem 5. But it is superior in practice, as we argued in Section 3. If the time complexity for edge coloring a constant-degree graph is dominated by $f(d')$ where d' is the maximum vertex degree of the constant-degree graph, then, for a given c, the maximum vertex degree of the constant-degree graph generated by Fürer and Raghavachari's algorithm is $d \geq \lceil \frac{c+1}{c-1} \rceil \geq (c+1)d' \geq 2d'$, i.e, their algorithm is slower than our algorithm. Since $f(d') > g(n)$, $\log n \leq d'^{3.5}$. Thus, the term $d'^7 \log^2 n$ in Theorem 7 dominates the entire time complexity, because $d'^7 \log^2 n \geq d'^7 d'^{3.5} = d'^{10.5}$, while the other term in this theorem is $\log^2 n \log \Delta \leq \log^3 n \leq d'^{10.5}$.

Compared with the algorithm in [5], the proposed algorithm has a particular advantage when G contains a constant number C of odd-cycles. For this special case, their NC algorithm needs $\lceil c\Delta \rceil$ colors, while our NC algorithm needs only $\Delta + C/2$ colors at most. In summary, we have

Corollary 8. *Let $G(V, E)$ be a simple graph containing a constant number C of odd-cycles. Then, edge coloring G with $\Delta + C/2$ colors is in NC.*

Proof. Suppose G contains C odd-cycles. Following the proof in Theorem 6 and setting $k = 1$, it is easy to show that $q \leq C/2$ is a constant too. The corollary then follows. □

Now we further generalize the algorithm above by giving the following theorem.

Theorem 9. *Let G be a graph with n vertices and m edges. Assume that $T(n, \Delta)$ is the time complexity of the fastest known parallel algorithm to edge color a graph of n vertices with $\Delta + 1$ colors, using $P(n, \Delta)$ processors on a PRAM. Then there exists a fast parallel algorithm for edge coloring G with $\Delta + \lfloor \Delta^{\alpha}/2 \rfloor + 3$ colors. The proposed algorithm takes $O(T(n, 2\Delta^{1-\alpha}) + \log^2 n \log \Delta)$ time using $O(\Delta^{\alpha} P(n, 2\Delta^{1-\alpha}))$ processors on the same model, where $0 < \alpha < 1$.*

Proof. In algorithm *COLOR_EDGE*, let $k = \lceil \Delta^{1-\alpha} \rceil$. The time complexity of Step 6 is $T(n, 2\Delta^{1-\alpha})$ because we now edge color a graph which is not of constant degree. The remaining analysis is similar to the proof of Theorem 6, and is omitted. □

In [17], Liang showed that edge coloring G with at least $\Delta + 2\Delta^{\alpha}$ colors requires $O(T(n, \Delta^{1-\alpha}) + \log n \log \Delta)$ time using $O(\Delta^{\alpha} P(n, \Delta^{1-\alpha}) + n\Delta)$ processors on a PRAM. Theorem 9 improves that result by reducing the number of colors by an additive term $3\Delta^{\alpha}/2 - 3$, without increasing the time complexity and the number of processors in the big O- sense. The following corollary can be derived easily from Theorem 9.

Corollary 10. *Let G be a graph with n vertices and m edges. Then there exists a fast parallel algorithm which edge colors G with $\Delta + \sqrt{\Delta}/2 + 1$ colors. The algorithm requires $O(T(n, 2\sqrt{\Delta}) + \log^2 n \log \Delta)$ time and $O(\sqrt{\Delta} P(n, 2\sqrt{\Delta}))$ processors on a PRAM.*

If $1/2 < \alpha < 1$, we can further reduce the number of colors used by increasing the time complexity. This is expressed by the following theorem.

Theorem 11. *Let G be a graph with n vertices and m edges. Assume that $T(n, \Delta)$ is the time complexity of the fastest known parallel algorithm to edge color a graph of n vertices with $\Delta + 1$ colors using $P(n, \Delta)$ processors on a PRAM. Then there exists a fast parallel algorithm for edge coloring G with $\Delta + \lfloor \Delta^{1-\alpha}/2 \rfloor + 3$ colors. The algorithm takes $O(T(n, 2\Delta^{\alpha}) + \log^2 n \log \Delta)$ time using $O(\Delta^{1-\alpha} P(n, 2\Delta^{\alpha}))$ processors on the same model, where $1/2 < \alpha < 1$.*

Proof. The proof is similar to that of Theorem 6. Here we assign $k = \lceil \Delta^{\alpha} \rceil$. □

5 Conclusion

In this paper we study the approximate edge coloring problem from the parallel viewpoint. We show that there exists a better trade-off between the number of colors used and the time spent to color the edges of a graph. Thus, it requires $O(T(n, 2\Delta^{1-\alpha}) + \log^2 n \log \Delta)$ time using $O(\Delta^{\alpha} P(n, 2\Delta^{1-\alpha}))$ processors to edge color G with $\Delta + \Delta^{\alpha}/2 + 3$ colors on a PRAM, where $T(n, \Delta)$ is the time complexity of the fastest known parallel algorithm to edge color a graph of n vertices with $\Delta + 1$ colors using $P(n, \Delta)$ processors on a PRAM, and the graph has the maximum vertex degree Δ. It is well known that deciding whether edge coloring G with $\Delta + 1$ colors is in NC is an important open problem. Even if we allow $\Delta + c\sqrt{\Delta}$ colors to edge color G where c is a constant, whether this approximate edge coloring problem is in NC is still unknown.

References

1. M. J. Atallah and U. Vishkin, Finding Euler tours in parallel, *J. of Computer and System Sciences* **29:**330-337, 1984.
2. M. Chrobak and M. Yung. Fast algorithms for edge coloring planar graphs. *J. Algorithms* **10:**35-51, 1989.
3. M. Chrobak and T. Nishizeki, Improved edge coloring algorithms for planar graphs, *J. Algorithms* **11:**102-116, 1990.
4. M. Fürer and B. Raghavachari. An efficient NC algorithm for finding Hamiltonian cycles in dense directed graphs. *J. Algorithms* **18:**203–220, 1995.
5. M. Fürer and B. Raghavachari. Parallel edge coloring approximation. *Parallel Proc. Lett.* **6:**321–329, 1996.
6. A. M. Gibbons, A. Israeli and W. Rytter, Parallel $O(\log n)$ time edge coloring of trees and Halin graphs, *Inform. Proc. Lett.* **27:**43-51, 1988.
7. A. V. Goldberg, A. Plotkin, and G. E. Shannon. Parallel symmetry breaking in sparse graphs. *SIAM J. Disc. Math.* **1:**434–446, 1988.
8. M. Goldberg and T. Spencer. Constructing a maximal independent set in parallel. *SIAM J. Disc. Math.* **2:**322-328, 1989.
9. T. Gonzalez and S. Sahni. Open shop scheduling to minimize finish time. *J. ACM* **23:**665–679, 1976.
10. D. A. Grable and A. Panconesi. Nearly optimal distributed edge coloring in $O(\log \log n)$ rounds. *Proc. 8th Annual ACM–SIAM Symp. on Discrete Algorithms*, January, 1997, 278–285.
11. Y. Han, W. Liang, and X. Shen. Very fast parallel algorithms for approximate edge coloring. Submitted for publication, April, 1997.
12. I. Holyer. The NP–completeness of edge coloring. *SIAM J. Computing* **10:**718-720, 1981.
13. J. JáJá. *An Introduction to Parallel Algorithms.* Addison-Wesley, 1992.
14. H. J. Karloff and D. B. Shmoys. Efficient parallel algorithms for edge coloring problems. *J. Algorithms* **8:**39-52, 1987.
15. E. L. Lawler and J. Labetoulle. On preemptive scheduling of unrelated parallel processors by linear programming. *J. ACM.* **25:**612–619, 1978.
16. G.F. Lev, N. Pippenger and L. G. Valiant. A fast parallel algorithm for routing in permutation networks. *IEEE Trans. Comput.* **C-30:**93-100, 1981.
17. W. Liang. Fast parallel algorithms for the approximate edge coloring problem, *Inform. Proc. Lett.* **56:**333-338, 1995.
18. W. Liang, X. Shen and Q. Hu. Parallel algorithms for the edge coloring and the edge coloring update problems. *J. of Parallel and Distributed Computing* **32:**66-73, 1996.
19. V. G. Vizing. On an estimate of the chromatic class of a p-graph (in Russian). *Diskret. Anal.* **3:**25-30: 1964.

This article was processed using the LaTeX macro package with LLNCS style