JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING **32,** 66–73 (1996)
ARTICLE NO. 0005

Parallel Algorithms for the Edge-Coloring and Edge-Coloring Update Problems

WEIFA LIANG,*,[1] XIAOJUN SHEN,†,[2] AND QING HU†,[3]

*Department of Computer Science, Australian National University, Canberra, ACT 0200, Australia; and †Computer Science Telecommunications Program, University of Missouri–Kansas City, 5100 Rockhill Road, Kansas City, Missouri 64110

Let $G(V, E)$ be a simple undirected graph with a maximum vertex degree $\Delta(G)$ (or Δ for short), $|V| = n$ and $|E| = m$. An edge-coloring of G is an assignment to each edge in G a color such that all edges sharing a common vertex have different colors. The minimum number of colors needed is denoted by $\chi'(G)$ (called the *chromatic index*). For a simple graph G, it is known that $\Delta \le \chi'(G) \le \Delta + 1$. This paper studies two edge-coloring problems. The first problem is to perform edge-coloring for an existing edge-colored graph G with $\Delta + 1$ colors stemming from the addition of a new vertex into G. The proposed parallel algorithm for this problem runs in $O(\Delta^{3/2} \log^3 \Delta + \Delta \log n)$ time using $O(\max\{n\Delta, \Delta^3\})$ processors. The second problem is to color the edges of a given uncolored graph G with $\Delta + 1$ colors. For this problem, our first parallel algorithm requires $O(\Delta^{5.5} \log^3 \Delta \log n + \Delta^5 \log^4 n)$ time and $O(\max\{n^2\Delta, n\Delta^3\})$ processors, which is a slight improvement on the algorithm by H. J. Karloff and D. B. Shmoys [*J. Algorithms* 8 (1987), 39–52]. Their algorithm costs $O(\Delta^6 \log^4 n)$ time and $O(n^2\Delta)$ processors if we use the fastest known algorithm for finding maximal independent sets by M. Goldberg and T. Spencer [*SIAM J. Discrete Math.* 2 (1989), 322–328]. Our second algorithm requires $O(\Delta^{4.5} \log^3 \Delta \log n + \Delta^4 \log^4 n)$ time and $O(\max\{n^2, n\Delta^3\})$ processors. Finally, we present our third algorithm by incorporating the second algorithm as a subroutine. This algorithm requires $O(\Delta^{3.5} \log^3 \Delta \log n + \Delta^3 \log^4 n)$ time and $O(\max\{n^2 \log \Delta, n\Delta^3\})$ processors, which improves, by an $O(\Delta^{2.5})$ factor in time, on Karloff and Shmoys' algorithm. All of these algorithms run in the COMMON CRCW PRAM model. © 1996 Academic Press, Inc.

1. INTRODUCTION

Let $G(V, E)$ be a simple graph with a maximum vertex degree $\Delta(G)$ (or Δ for short), $|V| = n$ and $|E| = m$. An *edge coloring* of G is an assignment of colors to its edges such that all edges sharing a common vertex are assigned different colors. Furthermore, we wish to use as few colors as possible. The minimum number of colors needed is denoted by $\chi'(G)$ (called the *chromatic index*). For a simple graph G, Vizing showed that $\Delta \le \chi'(G) \le \Delta + 1$ [18]. In fact, Vizing's proof implies an $O(nm)$ time algorithm with $\Delta + 1$ colors for the edge-coloring problem. However, Holyer has shown that deciding whether a graph requires Δ or $\Delta + 1$ colors is *NP-complete* [10]. For a multigraph G, Shannon showed that $\chi'(G) \le 3\Delta/2$ [16].

A number of parallel algorithms exist for the edge-coloring problem. Lev *et al.* [14] presented parallel edge-coloring algorithms with Δ colors for bipartite multigraphs. When $\Delta = 2^k$, their algorithm requires $O(\log \Delta \log n)$ time and $O(n\Delta)$ processors. Otherwise, their algorithm requires $O(\log \Delta \log^2 n)$ time and $O(n\Delta)$ processors on the EREW PRAM. Gibbons *et al.* [5] and Gibbons and Rytter [6] suggested algorithms for some other special graphs such as trees, outerplanar graphs, and Halin graphs. For trees, their algorithm requires $O(\log n)$ time and $O(n)$ processors; for outplanar graphs, their algorithm requires $O(\log^3 n)$ time and $O(n^2)$ processors; for Halin graphs, their algorithm requires $O(\log^2 n)$ time and $O(n)$ processors. All of their algorithms run in the CREW PRAM, and the number of colors used is at most Δ. For planar graphs, Chrobak and Yung [3] presented an edge-coloring algorithm with $\max\{\Delta, 19\}$ colors. Their algorithm runs in $O(\log^2 n)$ time and uses $O(n)$ processors on the EREW PRAM [3]. Later, Chrobak and Nishizeki [4] improved the algorithm in [3] by reducing the number of colors to $\max\{\Delta, 8\}$. Their algorithm requires $O(\log^3 n)$ time and $O(n^2)$ processors [4]. He [9] also discussed the edge-coloring problem with Δ colors for planar graphs, his algorithm has the same complexity as that of [3]. For general graphs, the pioneer work was done by Karloff and Shmoys [13]. They presented an edge-coloring algorithm with $\Delta + 1$ colors for simple graphs, which requires $O(\Delta^6 \log^4 n)$ time and $O(n^2\Delta)$ processors, assuming the fastest known algorithm for finding maximal independent sets [7] is used. They also gave a randomized edge-coloring algorithm with $\Delta + 20\Delta^{1/2+\varepsilon}$ colors, ($\varepsilon \le 1/4$), which runs in $O(\log^{O(1)} n)$ expected time and uses $O(n^{O(1)})$ processors (independent of ε). For multigraphs, Upfal once presented an $O(\log^3 \Delta n)$ time algorithm with $3\lceil \Delta/2 \rceil$ colors by using $O(\Delta n)$ processors (appears in [13]). For the special class of multigraphs of $\Delta = 3$, Karloff and Shmoys [13] presented an algorithm which runs in $O(\log n)$ time and uses $O(n)$ processors.

In this paper we study two problems. The first problem

[1] E-mail: wliang@cs.anu.edu.au.
[2] E-mail: xshen@cstp.umkc.edu.
[3] E-mail: qhu@cstp.umkc.edu.

is to deal with the edge-coloring updating when a new vertex is added to an existing edge-colored graph G with $(\Delta + 1)$ colors. Let the degree of this new vertex be at most Δ. The proposed parallel algorithm for this problem runs in $O(\Delta^{3/2} \log^3 \Delta + \Delta \log n)$ time using $O(\max\{n\Delta, \Delta^3\})$ processors. Later we use it as a basic subroutine for solving the second problem. The second problem is to color the edges of an arbitrary simple graph G with $\Delta + 1$ colors. We suggest three parallel algorithms for this problem. The first algorithm requires $O(\Delta^5 \log^4 n + \Delta^{5.5} \log^3 \Delta \log n)$ time and $O(\max\{n^2\Delta, n\Delta^3\})$ processors, which is a slight improvement on Karloff and Shmoys' algorithm. Then we present the second algorithm which requires $O(\Delta^4 \log^4 n + \Delta^{4.5} \log^3 \Delta \log n)$ time and $O(\max\{n^2, n\Delta^3\})$ processors. Finally, we present the third algorithm by using the second algorithm as a subroutine. This algorithm requires $O(\Delta^3 \log^4 n + \Delta^{3.5} \log^3 \Delta \log n)$ time and $O(\max\{n^2 \log \Delta, n\Delta^3\})$ processors, which improves by an $O(\Delta^{2.5})$ factor in time on Karloff and Shmoys' algorithm, and uses fewer processors than theirs when $\Delta \le O(n^{1/2})$. Both our third algorithm and Karloff and Shmoys' algorithm run in the COMMON CRCW PRAM model in which Concurrent Read and Concurrent Write can take place simultaneously, and Concurrent Writers are allowed only when the values being written are identical.

2. CONCEPTS AND NOTATIONS ABOUT EDGE COLORING

In this paper, we consider only a simple, connected, undirected graph $G(V, E)$, since the proposed algorithm can be applied to each connected component of G if G is not connected. First of all, we introduce some terminology for later use. An *independent set* is a vertex set in which no edge exists between any two vertices in the set. A *maximal independent set* is an independent set such that no more vertices can be added to it without introducing an edge. A *matching* is a set of edges such that any vertex in the graph G is incident to at most one edge in this set. A vertex is called *free* if it is not incident to any edge in the matching. A *maximal matching M* is a matching such that there is no edge between any two free vertices. A *maximal path* starting from a vertex r is a simple path P such that P cannot be extended without encountering a vertex that is already on the path [1].

VIZING'S THEOREM [18]. *Any simple graph $G(V, E)$ can be edge-colored with $\Delta + 1$ colors.*

Let G be an edge-colored graph. A color α is said to be *incident* to a vertex v if there exists an edge of color α which is incident to v; otherwise, α is free at v. Since there are $\Delta + 1$ colors and the degree of any vertex is at most Δ, there always exists a free color at each vertex. The proof of Vizing's Theorem relies on two recoloring techniques, *chain* and *fan*, which are explained below.

A graph G is *partially colored* if some edges in G are colored and some are not. Consider the subgraph of G induced by the edges of color α and the edges of color β. This subgraph must consist of a set of vertex-disjoint paths and simple cycles. Therefore, exchanging the colors α and β, for edges in such a path or such a cycle, yields a new partial coloring. Consider such a path from u to v, where α is free at u and β is free at v. If there exists an uncolored edge (w, u) in G such that β is free at w, but α is not, then after we exchange the colors on the path (called *$\alpha\beta$-path of u*), w and u will have a common free color β, and (w, u) can be colored with β. This procedure is called *chain recoloring*, or *inverting* the $\alpha\beta$-path.

There are many definitions of a fan. Here, we use the definition by Misra and Gries [15]. Let u be a vertex called the *fan center*. A *fan* $\langle v..w \rangle$ of u is an ordered sequence of vertices that satisfies all the following conditions: (i) $\langle v..w \rangle$ is a nonempty sequence of distinct neighboring vertices of u; (ii) the edge (u, v) is uncolored, and the edge (u, s) is colored for any vertex s other than v in the sequence; (iii) if s and t are two consecutive vertices of the sequence, denoted by $s \preceq t$ (s is the immediate predecessor of t in the sequence), then the color of the edge (u, t) is free at s.

Given an uncolored edge (u, v), the following procedure constructs a fan **f** of u that is *maximal* in that it cannot be extended:

```
Procedure Construct a fan f of u,
  f := ⟨v⟩; /* f is a queue, and let w be the rear of f. */
  While there exists an s: (u, s) ∈ E, s ∉ f, and the color
      of (u, s) is free at w do
      add s to the rear of f;
      w := s
  Endwhile
```

Given a fan $\langle v..w \rangle$ centered at u, as defined above, rotating this fan is to recolor edge (u, s) with the color of edge (u, t) for any pair of consecutive vertices s and $t (s \preceq t)$ in the fan. The purpose of rotating is to color edge (u, v). As a result, edge (u, w) becomes uncolored.

We now color an uncolored edge (u, v) after constructing a maximal fan of u. Let $\langle v..w \rangle$ be a maximal fan of u, and let α be free at u and β be free at w. The algorithm by Misra and Gries [15] is as follows:

Step 1. Exchange the colors on the $\alpha\beta$-path of u so that β becomes free at u;

Step 2. There exists an s which satisfies $s \in \langle v..w \rangle$, $\langle v..s \rangle$ is a fan of u, and β is free at s. Rotate fan $\langle v..s \rangle$ of u, and assign color β to edge (u, s).

3. THE EDGE-COLORING UPDATE ALGORITHM

Given an edge-colored graph $G(V, E)$ with $(\Delta + 1)$ colors, we consider the edge-coloring update problem when a new vertex v of degree K is added to G $(K \le \Delta)$, where Δ is the maximum degree of the new graph. It is not difficult to design an $O(Km)$ time sequential algorithm for this problem. However, designing a fast parallel algorithm for it does seem difficult. In this section, we present an algorithm for this problem which requires $O(\Delta^{3/2} \log^3 \Delta + \Delta \log n)$ time and $O(\max\{n\Delta, \Delta^3\})$ processors. Our algorithm is composed of two stages. The first stage colors as many edges incident to v as possible without changing the original edge-coloring of G. Thus, there will be no common free color at both v and u for any remaining uncolored edge (v, u). In the second stage, a *fan graph* FG is constructed first. Then, for an uncolored edge (v, u), a maximal path on FG is found which corresponds to a maximal fan centered at v, and the edge (v, u) is colored by rotating this maximal fan. This process is repeated until all uncolored edges incident to v are colored.

Now we discuss the first stage. Let v be the new vertex adjacent to vertices $v_1, v_2, ..., v_K$ in G. Firstly we construct a *color request graph* $G_1(X, Y, E_1)$, where $X = \{v_1, v_2, ..., v_K\}$, Y is the set of $\Delta + 1$ colors, and edge $(v_i, \alpha) \in E_1$ iff $\alpha \in Y$ is free at v_i. Obviously, G_1 is a bipartite graph and it can be constructed in $O(1)$ time with $O(K\Delta)$ processors. Secondly we find a maximal matching M in G_1 using the algorithm by Israeli and Shiloach [11, 12], which requires $O(\log^3 \Delta)$ time and $O(\Delta^2)$ processors. For every edge $(v_i, \alpha) \in M$, we color the edge (v, v_i) with α. After that, the two endpoints of any uncolored edge have no common free color.

The second stage is to color the remaining uncolored edges by applying the fan operation. For an uncolored edge (v, v_i), we construct a maximal fan centered at v. The construction of this fan seems to be "inherently" sequential, so, in their algorithm, Karloff and Shmoys sequentially construct a maximal fan. Here we present another approach to constructing a maximal fan in parallel, which is the crucial component of the proposed algorithm. Let $U = \{v_i \mid (v, v_i) \textit{ is not colored}\}$ and $N(v_i) = \{c \mid \text{color } c \text{ is free at } v_i\}$. Assume $U = \{u_1, u_2, ..., u_3\}$, where $1 \le s < K$. We construct an auxiliary directed graph $G_2(V_2, E_2)$, (i.e., the FG), where $V_2 = U \cup Y$, edge $\langle \alpha, \beta \rangle$ in E_2 if there exists v_i such that edge (v, v_i) is colored α and $\beta \in N(v_i)$, and each edge $\langle u_j, \gamma \rangle$ belongs to E_2, when $u_j \in U$ and $\gamma \in N(u_j)$, $1 \le j \le s$. If an edge (v, v_i) is colored α, then we call v_i, denoted by $f(\alpha)$, the *image* of α and α the *source* of v_i, because they are uniquely determined by each other.

LEMMA 3.1. *Let G_2 be defined as above, then a maximal directed path starting at u_i corresponds to a maximal fan centered at v, for an uncolored edge (v, u_i), $1 \le i \le s$.*

Proof. Let P be a directed maximal path of G_2 starting at u_i and ending at a color vertex δ (i.e., $\delta \in Y$). Note that every color vertex β (except δ) on the path P corresponds to a unique edge colored β. Assume σ and $\gamma (\gamma \ne \delta)$ are two consecutive vertices on P, and let edges (v, x) and (v, y) be colored σ and γ respectively; by the definition of G_2, the color γ is free at vertex x. Furthermore, the ending vertex δ of P implies that either there is an edge (v, v_i) colored δ and each color in $N(v_i)$ has already appeared in P or no edge (v, v_j) incident to v is colored δ, $j \ne i$. Therefore, if $P = \langle u_i, \alpha_1, \alpha_2, ..., \alpha_t \rangle$, then $\langle u_i, x_1, x_2, ..., x_t \rangle$ is a maximal fan, where α_i is the color of edge $\langle v, x_i \rangle$ and α_i is free at x_{i-1}, $(x_0 = u_i)$. On the other hand, let $u_i, x_1, x_2, ..., x_t$ be a maximal fan centered at v. Then color α_j is free at x_{j-1}, $1 < j \le t$, and α_1 is free at u_i. Therefore, $u_i, \alpha_1, \alpha_2, ..., \alpha_t$ is a directed path in G_2 by the construction of G_2. This path is also a maximal path. ■

Now we color the uncolored edge (v, u_i). We begin by finding a maximal directed path P starting at u_i in G_2. This can be implemented by directly applying the algorithm by Goldberg *et al.* [8]. Given such a path P, let P' be the corresponding maximal fan $\langle u_i..w \rangle$ centered at v, α be free at u_i, and β be free at w. We apply the rotating fan operation to color edge (v, u_i). That is, invert the $\alpha\beta$-path of v, update $N(v_j)$ for every vertex v_j on the $\alpha\beta$-path, find the first vertex x in the fan such that $\beta \in N(x)$, rotate the fan $\langle u_i..x \rangle$ and recolor edge (v, x) with color β. In order to find the vertex x, path P is represented by a linked list in which each vertex s has a pointer pointing to its immediate predecessor $F(s)$. Now we show how to find vertex x. Suppose the ending vertex on P is β, we apply the prefix computation algorithm to broadcast β to all other vertices on P. Let the image of a color vertex λ on P be y $(=f(\lambda))$. Then we test whether $\beta \in N(y)$ at each color vertex λ on P. If it does, that means, in the corresponding fan, there exists a vertex y on P' such that edge (v, y) is colored λ, and β is free at y. In this case, we set a special tag *change*(λ) to "1" for vertex λ. Initially, we set *change*(λ) to "0" for each color vertex λ on P. There may be more than one λ on P tagged "1", we find such a vertex φ whose distance to u_i is the shortest among all tagged vertices by using prefix computation. Then, broadcast φ along P toward u_i. When a vertex λ receives φ, it sets *change*(λ) := "2". Note that the image of φ is the vertex x $(=f(\varphi))$. Now $\langle u_i..x \rangle$ becomes a fan centered at v. A vertex s is in $\langle u_i..x \rangle$ iff its corresponding source λ on P has *change*(λ) = "2".

Having finished the preprocessing above, the fan $\langle u_i..x \rangle$ can be characterized by a pair of free colors (α, β). We color the edge (v, u_i) as follows. Construct a subgraph $G_{\alpha,\beta}$ of G induced by the edges with color α or β, find all connected components of $G_{\alpha,\beta}$, invert the $\alpha\beta$-path on which u_i belongs to, and rotate fan $\langle u_i..x \rangle$. All operations above can be done in parallel. Thus, the edge (v, u_i) is colored, and the edge (v, x) is colored with color β.

THEOREM 3.1. *Given an edge-colored graph $G(V, E)$ with $(\Delta + 1)$ colors and a new graph G' obtained from G by adding a new vertex of degree K $(K \le \Delta)$ to G, the edge-coloring of G' requires $O(K\sqrt{\Delta} \log^3 \Delta + K \log n)$ time and $O(\max\{n\Delta, \Delta^3\})$ processors.*

Proof. From the above discussion, we know the proposed algorithm for this problem is composed of two stages. In the first stage, constructing the color request graph G_1 needs $O(1)$ time and $O(K\Delta)$ processors, and finding a maximal matching in G_1, requires $O(\log^3 \Delta)$ time and $O(K\Delta)$ processors [11]. In the second stage, constructing G_2 requires $O(1)$ time and $O(\Delta^3)$ processors; finding a maximal directed path in G_2 that corresponds to a maximal fan requires $O(\sqrt{\Delta}\log^3\Delta)$ time and $O(\Delta^2)$ processors [8]; the prefix computation on P requires $O(\log\Delta)$ time and $O(\Delta)$ processors; computing the connected components of G_2 and inverting the $\alpha\beta$-path require $O(\log n)$ time and $O(n\Delta)$ processors [17]. In addition, the second stage needs at most K iterations. ■

4. A NEW EDGE-COLORING ALGORITHM

4.1. A Simple Version of the Karloff–Shmoys' Algorithm

Now we begin to discuss the edge-coloring problem with $\Delta + 1$ colors. The parallel algorithm by Karloff and Shmoys [13] is based on the original proof of Vizing's Theorem. Here we give a simple parallel algorithm. The basic idea of our algorithm is the same as theirs. The difference is that ours is based on a new constructive proof of Vizing's Theorem by Misra and Gries [15].

We introduce some definitions first. If G is partially colored, then a vertex v is said to be active if there is at least one uncolored edge incident to v. Let S be a vertex subset. Denote by G_S the graph induced by S. Now we define an undirected graph $G^2 = (V, E^2)$ from $G(V, E)$, where $E^2 = E \cup \{(u, v) \mid$ there exists a vertex w such that $(u, w) \in E, (w, v) \in E\}$. This means that there is an edge in G^2 between vertices u and v iff there is a path of no more than two edges between u and v in G. Obviously G^2 can be constructed in $O(\log\Delta)$ time with $O(n^2\Delta)$ processors or $O(1)$ time with $O(n^3)$ processors.

Denote by $\mathscr{A}$ the set of active vertices. The proposed algorithm consists of $O(\Delta^5\log n)$ phases. In each phase we try to color as many uncolored edges as possible. However, there may exist edge-coloring conflicts which mean that one edge may be assigned different colors at the same time when the operations of edge-coloring are executed in parallel. We will show that the edges adjacent to $\Omega(|\mathscr{A}|/\Delta^4)$ active vertices can be colored in parallel without any edge-coloring conflict. We also know that each active vertex has at most Δ uncolored edges. Therefore, an $\Omega(1/\Delta^5)$ factor of the remaining uncolored edges will be colored in each phase. As a result, $O(\Delta^5\log n)$ phases are enough to color all uncolored edges incident to active vertices. The detailed algorithm is as follows.

Algorithm 1: Color_Edge_1.

Step 0. Initialization, i.e., $\mathscr{A} := V$.

Step 1. Construct $G^2_{\mathscr{A}}$ (induced by $\mathscr{A}$), and find a maximal independent set I in $G^2_{\mathscr{A}}$ by using the algorithm of Goldberg and Spencer [7] which requires $O(\log^3 n)$ time and $O(n\Delta/\log n)$ processors.

Step 2. For each vertex $v \in I$, choose an uncolored edge $e(v)$ incident on v. The set of these chosen uncolored edges is denoted by M.

Step 3. Set $em(v) = u$ for each edge $e(v) = (v, u) \in M$. If there exists a color α which is free at both v and $em(v)$, then color edge $(v, em(v))$ with color α and remove v from I. Let the remaining vertex set be I^* $(\subseteq I)$. Construct a directed graph G_2 as discussed in Section 3 and find a maximal fan centered at v for each vertex $v \in I^*$ in parallel.

Step 4. For each vertex $v \in I^*$, we have a maximal fan centered at v, characterized by a pair of colors $(\alpha(v), \beta(v))$, where $\alpha(v)$ is free at v and $\beta(v)$ is free at the rear of this fan. Let $I_{\alpha,\beta} = \{v \mid v \in I^*$, and the maximal fan at v is characterized by $(\alpha, \beta)\}$. Denote by I^{**} the largest $I_{\alpha,\beta}$, i.e., $|I_{\alpha,\beta}| = \max_{\delta,\gamma\in\{\Delta+1\text{colors}\}} |I_{\delta,\gamma}|$.

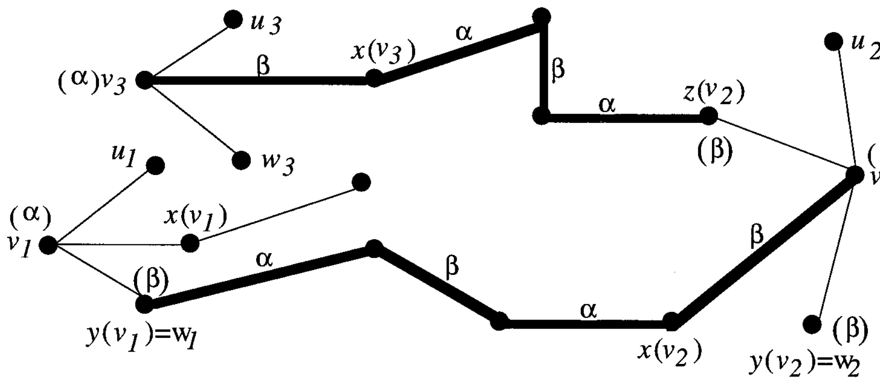
Let $\langle v..w\rangle$ be a maximal fan at u, α be free at u, and β be free at w. Suppose t is the first vertex in $\langle v..w\rangle$ such that edge (u, t) is colored with β. Then β must be free at t's immediate predecessor s $(s \preceq t)$. Assign $x(v) := t$, $y(v) := w$, and $z(v) := s$. Otherwise, none of the edges incident to v is colored with β. In this case, we set $x(v) := \phi$, $y(v) := w$, and $z(v) := \phi$. Before applying the algorithm by Misra and Gries (described in Section 2) to each vertex $v \in I^{**}$, let us recall the precondition of Step 2 in their algorithm. It states that there exists an s that satisfies $s \in \langle v..w\rangle$, $\langle v..s\rangle$ is a fan of u and β is free at s. However, after inverting the $\alpha\beta$-path of v for all $v \in I^{**}$ in parallel, this precondition may not hold any more for all $v \in I^{**}$. For example, in Fig. 1, there are three maximal fans $\langle u_i..w_i\rangle$ at v_i, $i = 1, 2, 3$, characterized by a pair of colors (α, β).

Assume $x(v_1) = \phi$, $y(v) = w_1$ and $z(v_1) = \phi$, $x(v_2) \neq \phi$, $y(v_2) = w_2$ and $z(v_2) \neq \phi$, $x(v_3) \neq \phi$, $y(v_3) = w_2$ and $z(v_3) \neq \phi$. Now we invert the $\alpha\beta$-path starting at v_i simultaneously, $i = 1, 2, 3$. As a result, the color β is no longer free at both vertex w_1 and $z(v_2)$, and the precondition of Step 2 of Misra and Gries' algorithm does not hold any more at vertex v_1 and v_2. In order to avoid this situation, we construct a conflict graph $G_{\text{conflict}}(I^{**}, E^{**})$ such that (u, v) is an edge in E^{**} iff the precondition does not hold when we invert their $\alpha\beta$-paths of u and v in parallel. Let I^{***} be a maximal independent set of G_{conflict}. It is obvious that the precondition will hold after we invert the $\alpha\beta$-path of v for all $v \in I^{***}$. I^{***} can be calculated as follows. We first construct the subgraph $G_{\alpha,\beta}$ of G induced by edges colored α or β. Then we construct the graph $G_{\text{conflict}}(I^{**}, E^{**})$, where $(v, u) \in E^{**}$ iff either vertices $x(v)$ $(\neq\phi)$ and $y(u)$ belong to the same connected component of $G_{\alpha,\beta}$ or vertices $x(v)$ and $z(u)$ $(\neq\phi)$ belong to the same connected component of $G_{\alpha,\beta}$. In fact, there exists an edge between u and v in G_{conflict} iff there is an $\alpha\beta$-path in $G_{\alpha,\beta}$ whose two endpoints are $x(v)$ and $y(u)$ or $x(v)$ and $z(u)$, respectively. Obviously $\Delta(G_{\text{conflict}}) \leq 3$, because there are at most three edges incident to vertex v, which are generated by $x(v)$, $y(v)$, or $z(v)$. A maximal independent set I^{***} in graph $G_{\text{conflict}}(I^{**}, E^{**})$ can be found by using the algorithm of Goldberg and Spencer [7].

Step 5. Apply Misra and Gries's algorithm now. That is, invert the $\alpha\beta$-path of v, rotate the fan centered at v, and color edge $(v, em(v)) \in M$ with color β for each $v \in I^{***}$.

Step 6. If $v \in \mathscr{A}$ and all edges incident to v have been colored, then $\mathscr{A} := \mathscr{A} - \{v\}$.

Step 7. If $\mathscr{A} \neq \phi$ goto Step 1.



━━━	The edge is colored with α or β.
───	The edge is colored with neither α nor β.
● (β)	Color β is free at this vertex.

FIG. 1. Inverting the $\alpha\beta$-path of v_2 affects the free color of w_1 in a fan of v_1, and inverting the $\alpha\beta$-path of v_3 affects the free color of $z(v_2)$ in a fan of v_2.

THEOREM 4.1. *Given a simple undirected graph $G(V, E)$, $|V| = n$. The edge-coloring of G by Algorithm 1 requires $O(\Delta^5 \log^4 n + \Delta^{5.5} \log n \log^3 \Delta)$ time and $O(\max\{n\Delta^3, n^2\Delta\})$ processors.*

Proof. Let $|\mathcal{A}| = n_0$. Because $\Delta(G^2_{\mathcal{A}}) \leq \Delta^2$, we have $|I| \geq \Omega(n_0/\Delta^2)$ and $I^* \subseteq I$. There are Δ^2 pairs of colors and so $|I^{**}| \geq |I|/\Delta^2 = \Omega(n_0/\Delta^4)$. We also know $\Delta(G_{\text{conflict}}) \leq 3$. Thus, $|I^{***}| \geq |I^{**}|/4 = \Omega(n_0/\Delta^4)$. That means, $\Omega(n_0/\Delta^4)$ uncolored edges can be colored in one iteration of Steps 1–7 of Algorithm 1. Initially $n_0 = n$. All other operations can be done in $O(\log^3 \Delta n)$ time with $O(n\Delta)$ processors except that (i) the construction of G_2 requires $O(1)$ time and $O(\Delta^3)$ processors and finding a maximal fan requires $O(\sqrt{\Delta} \log^3 \Delta)$ time and $O(\Delta^2)$ processors for each $v \in I^*$; (ii) the constructions of $G^2_{\mathcal{A}}$ and G_{conflict} require $O(\log \Delta)$ time and $O(n^2\Delta)$ processors and $O(1)$ time and $O(n^2)$ processors, respectively. We have proven at the beginning of this section that the number of iterations of Steps 1–7 is $O(\Delta^5 \log n)$. So, the proposed algorithm requires $O(\Delta^5 \log^4 n + \Delta^{5.5} \log^3 \Delta \log n)$ time and $O(\max\{n\Delta^3, n^2\Delta\})$ processors. ■

4.2. An Improved Edge-Coloring Algorithm

In this section, an improved algorithm for the edge coloring problem is introduced. This algorithm is basically the same as that of Karloff and Shmoys [13] except that: (i) a maximal independent set is found by using the auxiliary graph G_3 instead of G^2. As a result, we can find a maximal independent set of size $\Omega(n/\Delta)$ which is larger than $\Omega(n/\Delta^2)$ of theirs; (ii) the new algorithm for constructing maximal fans in Section 3 is employed. The basic idea of the proposed algorithm is as follows. First, we use more than $\Delta + 1$ colors to color the edges of G quickly. Actually, we use $3\lceil \Delta/2 \rceil$ colors to color the edges of G. Then we remove all extra colors from the colored graph until $\Delta + 1$ colors are left. Consequently, we obtain a partially edge-colored graph with $\Delta + 1$ colors. Finally, we color the remaining uncolored edges phase by phase. In each phase we only color those uncolored edges which were colored with the same color initially. Denote by M this uncolored edge set. It is obvious that M is a matching of G. We try to choose as many edges in M as possible such that one of the endpoints of each chosen edge serves as the center of a maximal fan and no edge-coloring conflict will occur when the rotating fan operation is applied to each of these fans in parallel. In order to choose the edges, we construct an auxiliary undirected graph $G_3(V_3, E_3)$, where a vertex of V_3 represents an edge in M, $V_3 = M$. Let $e_1 = (u_1, v_1) \in M$ and $e_2 = (u_2, v_2) \in M$. Edge (e_1, e_2) is in E_3 iff one of edges in $\{(u_1, u_2), (u_1, v_2), (v_1, u_2), (v_1, v_2)\}$ is in E. The detailed algorithm is as follows.

ALGORITHM 2: Color_Edge_2.

```
Step 1. Check whether $G$ is a Eulerian graph. If not, add a new vertex $v$ to $G$, and add an edge between $v$ and any one vertex with odd degree in $G$. As a result, we obtain a Eulerian graph $G'$.
Step 2. Apply the Upfal's edge-coloring algorithm for multigraphs [13] to color $G'$ with $3\lceil \Delta/2 \rceil$ colors.
Step 3. Delete all colored edges incident to $v$ and calculate the number of edges in $G$ colored with color $\alpha$, $count(\alpha)$, for each color $\alpha$. Let $q$ be the number of colors used. Sort these colors in increasing order of their counting numbers, and assign each color $\alpha$ a unique index $index(\alpha)$ between 1 and $q$. Thus, $index(\alpha) > index(\beta)$ if $count(\beta) > count(\alpha)$.
Step 4. **While** $q > \Delta + 1$ **do**
  Step 5. Remove the color from all edges in $M$, where $M = \{$edges with color indexed $q\}$.
  **Repeat**
    Step 6. If both endpoints of an uncolored edge in $M$ share a free valid color (colors indexed between 1 and
$\Delta + 1$ are considered to be valid), then assign this edge with this free color. Let the remaining uncolored edge set be $M^*$, construct $G_3$ for $M^*$.
        Step 7.  Find a maximal independent set $IE$ of $G_3$. Note that each vertex in $IE$ is an edge in $G$. For each vertex (an edge in $G$) in $IE$, choose one endpoint of this edge as the fan center. Denote by $I^*$ the set of fan centers, find a maximal fan $\langle u..w \rangle$ at $v$ by constructing a directed graph $G_2$ for each $v \in I^*$ in parallel.
        Step 8.  The same as Step 4 of Algorithm 1.
        Step 9.  The same as Step 5 of Algorithm 1.
        Step 10.  $M := M^*$-{all colored edges in Step 9}
    Until $M = \phi$;

    $q := q - 1$

Endwhile
```

LEMMA 4.1. *Given a fixed q_0, the number of iterations of Steps* 6–10 *in Algorithm* 2 *is $O(\Delta^3 \log n)$.*

Proof. Initially $|M| \leq \lfloor n/2 \rfloor$ because M is a matching of G. By the construction of graph G_3, we know that $\Delta(G_3)$ is 2Δ. So, the size of any maximal independent set of G_3 is at least $|M|/(2\Delta + 1)$, which corresponds to the same number of maximal fans. There are $\Delta(\Delta - 1)/2 < \Delta^2/2$ pairs of colors for these maximal fans. Therefore, in each iteration of Steps 6–10, we can rotate at least $|I^{***}| \geq |M|/(2\Delta^3 + \Delta^2)$ fans, which means $\Omega(|M|/\Delta^3)$ uncolored edges are colored. Since the size of M reduces by a factor of $(1 - 1/\Delta^3)$ in each iteration, coloring all edges in M requires $O(\Delta^3 \log n)$ iterations of Steps 6–10. ■

THEOREM 4.2. *Given a simple undirected graph $G(V, E)$, $|V| = n$, edge-coloring of G with $\Delta + 1$ colors by Algorithm* 2 *requires $O(\Delta^4 \log^4 n + \Delta^{4.5} \log^3 \Delta \log n)$ time and $O(\max\{n^2, n\Delta^3\})$ processors.*

Proof. By Algorithm 2, we know all operations in Steps 1–3 require at most $O(\log^3(n\Delta))$ time and $O(n\Delta)$ processors. We also know that we need to execute the **While** loop t times, where $t = (\Delta + 3)/2$ if Δ is odd, otherwise $t = \Delta/2 - 1$. Therefore, we need to execute the **Repeat** loop a total of $O(\Delta^4 \log n)$ times by Lemma 4.1. The construction of G_3 in Step 6 requires $O(1)$ time and $O(n^2)$ processors. Step 7 needs $O(\log^3 n)$ time and $O(n\Delta^3)$ processors, because finding a maximal independent set in G_3 needs $O(\log^3 n)$ time and $O(n\Delta/\log n)$ processors; constructing G_2 and finding a maximal fan in G_2 for every $v \in I^*$ require $O\sqrt{\Delta} \log^3 \Delta)$ time and $O(\Delta^2)$ processors [8] and $O(1)$ time and $O(\Delta^3)$ processors, respectively. From the discussion in Section 4.1, Steps 8 and 9 need $O(\log^3 n)$ time and $O(n\Delta)$ processors. Thus, the entire algorithm requires $O(\Delta^4 \log^4 n + \Delta^{4.5} \log^3 \Delta \log n)$ time and $O(\max\{n^2, n\Delta^3\})$ processors. ■

The above algorithm improves by an $O(\Delta^{1.5})$ factor in time over Karloff and Shmoys' algorithm. In addition, our algorithm needs fewer processors when $\Delta \leq O(n^{1/2})$.

We now propose a more efficient algorithm for the edge-coloring problem. The main idea is to decompose G into many edge-disjoint subgraphs first, then color and merge these edge-colored subgraphs in parallel.

LEMMA 4.2. *Given two simple undirected graphs $GA_1(VA_1, EA_1)$ and $GB_2(VB_2, EB_2)$, $VA_1 \subseteq V$, $VB_2 \subseteq V$, $|V| = n$, $EA_1 \cap EB_2 = \phi$. Assume GA_1 and GB_2 are edge-colored graphs with $\Delta(GA_1) + 1$ and $\Delta(GB_2) + 1$ colors, respectively. Let their color sets be C_1 and C_2 respectively, where $C_1 \cap C_2 = \phi$. Then edge-coloring the graph $GA_1 \cup GB_2 = (VA_1 \cup VB_2, EA_1 \cup EB_2)$ with $\Delta_{AB} + 1$ colors requires $O(\Delta_{AB}^{3.5} \log^3 \Delta_{AB} \log n + \Delta_{AB}^3 \log^4 n)$ time and $O(\max\{n^2, n\Delta_{AB}^3\})$ processors, where $\Delta_{AB} = \Delta(GA_1) + \Delta(GB_2)$.*

Proof. Because $C_1 \cap C_2 = \phi$, the edges in graph $GA_1 \cup GB_2$ have been colored with $|C_1 \cup C_2| = \Delta(GA_1) + \Delta(GB_2) + 2$ colors. Now we remove a color α from the color set $C_1 \cup C_2$, and decolor those edges which were colored with α in $GA_1 \cup GB_2$. The remaining task is to color these uncolored edges by the colors in $C_1 \cup C_2$-$\{\alpha\}$, and such edge-coloring can be done by Vizing's Theorem. The implementation is the same as that of Steps 6–10 in Algorithm 2. Now let us analyze its complexity. The number of iterations of Steps 6–10 in Algorithm 2 is $O(\Delta_{AB}^3 \log n)$ by Lemma 4.1. Finding a maximal independent set requires $O(\log^3 n)$ time and $O(n\Delta_{AB}/\log n)$ processors [7]. The construction of G_2 and G_{conflict} require $O(1)$ time and $O(\Delta_{AB}^3)$ processors and $O(1)$ time and $O(n^2)$ processors, respectively. Finding a maximal fan for G_2 requires $O(\Delta_{AB}^{1/2} \log^3 \Delta_{AB})$ time and $O(\Delta_{AB}^2)$ processors [8]. Consequently, coloring all uncolored edges in $GA_1 \cup GB_2$ requires $O(\Delta_{AB}^{3.5} \log^3 \Delta_{AB} \log n + \Delta_{AB}^3 \log^4 n)$ time and $O(\max\{n^2, n\Delta_{AB}^3\})$ processors. ■

Now we are ready to show our third algorithm.

ALGORITHM 3: Color_Edge_3.

Step 1. Form a bipartite graph $G_B(X', Y', E_B)$ as follows. If G is not a Eulerian graph, then add a new vertex v that is adjacent to all vertices of odd degree. Find a Eulerian tour of the augmented graph, and view the tour as a directed tour by the algorithm due to Atallah and Vishkin [2]. Let $X' = \{v' | v \in V\}$ and $Y' = \{v'' | v \in V\}$. For each edge $\langle u, v \rangle$ traversed from u to v, add an undirected edge (u', v'') to E_B. As a result, G_B is a bipartite graph with the maximum degree $\lceil \Delta(G)/2 \rceil$.

Step 2. Color G_B's edges with $\lceil \Delta(G)/2 \rceil$ colors by using the optimal algorithm of Lev *et al.* [14].

Step 3. Let $M_i(\subseteq E_B)$ be a set of edges colored c_i, where c_i is a color. Construct a subgraph $SG_i(V(M_i), E_i)$ as follows: for each edge $(u', v'') \in M_i$, put an edge (u, v) into E_i. Since M_i is a matching in G_B, $\Delta(SG_i) \leq 2$; thus SG_i is a collection of disjoint paths and cycles. Now G is decomposed into subgraphs $SG_0, SG_1, ..., SG_{\lceil \Delta/2 \rceil - 1}$;

```
Step 4.  $p := \lceil \Delta/2 \rceil - 1$;
Step 5.  For each $i$: $0 \leq i \leq \lceil \Delta/2 \rceil - 1$ pardo
             color graph $SG_i$ using colors in a color set
```

```
        $C_i$, $|C_i| = 3$, and $C_i \cap C_j = 0$, $i \neq j$;
        $G_i^{(0)} := SG_i$; $C_i^{(0)} := C_i$
    Endfor;
Step 6.  For $k := 0$ to $\lceil \log\Delta - 1 \rceil$ do
           For each $i$: $0 \leq i \leq \lfloor p/2 \rfloor$ pardo
             arbitrarily select a color $\alpha_i$, decolor those
             edges whose current color is $\alpha_i$ in graph
             $G_i^{(k+1)} := G_{2i}^{(k)} \cup G_{2i+1}^{(k)}$, where $\alpha_i \in C_{2i}^{(k)} \cup C_{2i+1}^{(k)}$.
             Let the uncolored edge set in graph $G_i^{(k+1)}$ be $M$,
             color edges in $M$ with colors
             in $C_i^{(k+1)} := C_{2i}^{(k)} \cup C_{2i+1}^{(k)}$-$\{\alpha_i\}$ by Lemma 4.2
           Endfor;
           If $p$ is even then
             $oldcount := p$; $p := p/2 + 1$;
         $G_p^{(k+1)} := G_{\text{oldcount}}^{(k)}$;
         /* the unpaired subgraph is assigned a new
         index */
           else $p := \lceil p/2 \rceil$
    Endfor
```

THEOREM 4.3. *Given a simple undirected graph $G(V, E)$, $|V| = n$, edge-coloring G with $\Delta + 1$ colors by Algorithm 3 requires $O(\Delta^{3.5} \log^3 \Delta \log n + \Delta^3 \log^4 n)$ time and $O(\max\{n^2 \log \Delta, n\Delta^3\})$ processors.*

Proof. By Algorithm 3, we know that Steps 1–5 require at most $O(\log^3 (n\Delta))$ time and $O(n\Delta)$ processors [2, 14]. Now we analyze the computational complexity of Step 6. Given a fixed $k = k_0$, we know $\Delta(G_i^{(k_0)}) \leq 2^{(k_0+1)}$, $0 \leq i \leq p$. So, edge-coloring graph $G_i^{(k_0)}$ requires $O((2^{(k_0+1)})^{3.5} \log^3 (2^{(k_0+1)}) \log n + (2^{k_0+1)})^3 \log^4 n)$ time and $O(\max\{n^2, n(2^{(k_0+1)})^3\})$ processors by Lemma 4.2. Therefore, Step 6 requires $\sum_{k=0}^{\lceil \log\Delta - 1 \rceil} O((2^{(k+1)})^{3.5} \log^3 (2^{(k+1)}) \log n + (2^{(k+1)})^3 \log^4 n) = O(\Delta^{3.5} \log^3 \Delta \log n + \Delta^3 \log^4 n)$ time and $O(\max\{n^2 \log \Delta, n\Delta^3\})$ processors. ∎

5. CONCLUSION

In this paper, we deal with two edge-coloring problems with $\Delta + 1$ colors. One is concerned with the edge-coloring update problem where a new vertex is added to an edge-colored graph. The other is the traditional edge-coloring problem. We have proposed parallel algorithms for solving these problems. Indeed, when $\Delta = O(\log^{O(1)} n)$, the proposed algorithms are efficient. Compared with the algorithm by Karloff and Shmoys, our algorithm for the same problem improves by an $O(\Delta^{2.5})$ factor in time over theirs and uses fewer processors when $\Delta \leq O(n^{1/2})$.

ACKNOWLEDGMENTS

We thank the anonymous referees for their valuable comments and suggestions. We also thank Stephanie Griffin for her great help in improving the presentation of the paper.

REFERENCES

1. R. J. Anderson, A parallel algorithm for the maximal path problem. *Proc. 17th ACM Symp. Theory of Computing,* 1985, pp. 33–37. Also see *Combinatorica* **7** (1987).
2. M. Atallah and U. Vishkin, Finding Euler tours in parallel. *J. Comput. System Sci.* **29** (1984), 330–337.
3. M. Chrobak and M. Yung, Fast algorithms for edge-coloring planar graphs. *J. Algorithms* **10** (1989), 35–51.
4. M. Chrobak and T. Nishizeki, Improved edge-coloring algorithms for planar graphs. *J. Algorithms* **11** (1990), 102–116.
5. A. M. Gibbons, A. Israeli, and W. Rytter, Parallel $O(\log n)$ time edge-coloring of trees and Halin graphs. *Inform. Process. Lett.* **27** (1988), 43–51.
6. A. Gibbons and W. Rytter, *Efficient Parallel Algorithms.* Cambridge Univ. Press, Cambridge, UK, 1988.
7. M. Goldberg and T. Spencer, Constructing a maximal independent set in parallel. *SIAM J. Discrete Math.* **2** (1989), 322–328.
8. A. V. Goldberg, S. A. Plotkin, and P. M. Vaidya, Sublinear-time parallel algorithms for matching and related problems. *J. Algorithms* **14,** 2 (1993), 180–213.
9. X. He, An efficient algorithm for edge coloring planar graphs with Δ colors. *Theoret. Comput. Sci.* **74,** 3 (1990), 299–312.
10. I. Holyer, The *NP*-completeness of edge coloring. *SIAM J. Comput.* **10,** 4 (1981), 718–720.
11. A. Israeli and Y. Shiloach, An improved algorithm for maximal matching, *Inform. Process. Lett.* **33** (1986), 57–60.
12. A. Israeli and Y. Shiloach, A fast and simple randomized parallel algorithm for maximal matching. *Inform. Process. Lett.* **22** (1985), 70–80.
13. H. J. Karloff and D. B. Shmoys, Efficient parallel algorithms for edge coloring problems. *J. Algorithms* **8** (1987), 39–52.
14. G. F. Lev, N. Pippenger, and L. G. Valiant, A fast parallel algorithm for routing in permutation networks. *IEEE Trans. Comput.* **C-30,** 2 (1981), 93–100.
15. J. Misra and D. Gries, A constructive proof of Vizing's theorem. *Inform. Process. Lett.* **41** (1992), 131–133.
16. C. E. Shannon, A theorem on coloring lines of a network. *J. Math. Phys.* **28** (1949), 148–151.
17. Y. Shiloach and U. Vishkin, An $O(\log n)$ parallel connectivity algorithm. *J. Algorithms* **3** (1982), 57–67.
18. V. G. Vizing, On an estimate of the chromatic class of a p-graph. *Diskret. Anal.* **3** (1964), 25–30. [In Russian]

WEIFA LIANG received his B.S. degree in computer science from Wuhan University, China in 1984 and his M.S. degree in computer science from the University of Science and Technology of China in 1989. He was a visiting scholar in the Computer Science Telecommunications Program at the University of Missouri–Kansas City from 1991 to 1993. Currently, he is a Ph.D candidate in the Department of Computer Science at the Australian National University. His current research interests include parallel processing, parallel and distributed algorithms, and interconnection networks.

XIAOJUN SHEN received his Ph.D. degree in computer science from the University of Illinois, Urbana–Champaign, in 1989. He received his M.S. degree in computer science from the Nanjing University of Science and Technology, China in 1982 and his B.S. degree in numerical analysis from the Qinghua University, Beijing, China in 1968. He is currently an associate professor in the Computer Science Telecommunications Program at the University of Missouri–Kansas City. His current research interests include parallel processing, discrete mathematics, algorithms, and computer networking.

QING HU received his B.S. and M.S. degrees in computer science from Nanjing University of Science and Technology (formerly East China Institute of Technology), China in 1982 and 1989, respectively. Then he

was with the Computer Science Department of that university until 1991. In September of 1991, he came to United States to start his doctorate program. Currently, he is a Ph.D. candidate in the Computer Science Telecommunications Program, University of Missouri–Kansas City. His current research interests include parallel processing, computer networking, image processing, and pattern recognition. He is also working at the development and implementation of a distributed medical transcription database system.

Received October 25, 1993; revised January 17, 1995; accepted June 8, 1995