Many combinatorial problems can be efficiently solved in parallel for series-parallel multigraphs. The edge-coloring problem is one of a few combinatorial problems for which no NC parallel algorithm has been obtained for series-parallel multigraphs. This paper gives an NC parallel algorithm for the problem on series-parallel multigraphs $G$. It takes $O(\log n)$ time with $O(D_n^r \log n)$ processors, where $n$ is the number of vertices and $D$ is the maximum degree of $G$.

1. INTRODUCTION

This paper deals with the edge-coloring problem, which asks that all edges of a given graph $G$ be colored, using the minimum number of colors, so that no two adjacent edges are colored with the same color. The minimum number is called the chromatic index of $G$, and denoted by $\chi'(G)$. It is known that many combinatorial problems can be solved very efficiently, say in linear time, for series-parallel multigraphs or partial $k$-trees [2–4, 8, 17]. Such a class of problems has been characterized in terms of “forbidden subgraphs” or “extended monadic second-order logic” [2–4, 8, 17]. The edge-coloring problem does not belong to such a class of the “maximum (or minimum) subgraph problems,” and is indeed one of the “edge-partitioning problems” which do not appear to be efficiently solved for series-parallel multigraphs or partial $k$-trees [4]. However, the

$$\chi'(G)$$

359
current authors have recently obtained a linear-time sequential algorithm for series-parallel multigraphs in the companion paper [21]. On the other hand, He [12] has shown that there exist NC parallel algorithms for many “vertex-type” problems on series-parallel simple graphs, such as vertex-coloring, maximum independent set, and vertex-cover. However, his method is not valid for the edge-coloring problem. Thus NC parallel edge-coloring algorithms have not been obtained so far for series-parallel multigraphs, but NC parallel algorithms have been obtained for the following classes of graphs: planar graph with maximum degree \( \geq 9 \) [7]; outerplanar graphs [5, 11]; series-parallel simple graphs [6]; and partial \( k \)-trees [20]. The algorithm for series-parallel simple graphs in [6] takes \( O(\log \Delta n) \) time with \( O(n) \) processors and is not an optimal parallel algorithm. Every series-parallel simple graph except odd cycles can be edge-colored with \( \Delta \) colors, where \( \Delta \) denotes the maximum degree of a graph [16, 18]. However, this is not the case for series-parallel multigraphs, and the edge-coloring problem for series-parallel multigraphs is much more difficult than for simple graphs. On the other hand, the algorithm for partial \( k \)-trees in [20] takes \( O(\log n) \) time with \( O((6k)^{k(k+1)/2} n / \log n) \) processors for any \( k \), and is an optimal parallel algorithm for bounded \( k \) although the constant \((6k)^{k(k+1)/2}\) is large, say 1728 for the case \( k = 2 \). Note that a series-parallel simple graph is a partial 2-tree but a series-parallel multigraph is not always a partial 2-tree.

In this paper, using a “tree contraction” technique, we give an efficient parallel implementation of our linear sequential algorithm in the companion paper [21] for the edge-coloring problem on series-parallel multigraphs. The parallel computation model we use is an exclusive-read and exclusive-write parallel random access machine (EREW PRAM). Our parallel algorithm takes \( O(\log n) \) time with \( O(\Delta n / \log n) \) processors. This is the first NC algorithm for series-parallel multigraphs and is an optimal parallel algorithm if \( \Delta \) is bounded. Furthermore, combining our algorithm in this paper and the algorithm for a partial \( k \)-tree in [20], one can easily obtain a truly practical and optimal parallel algorithm for series-parallel simple graphs. It takes \( O(\log n) \) time with \( O(n / \log n) \) processors, and hence greatly improves the complexity or the constant over the previously best known ones for series-parallel simple graphs [6, 20]. An early version of this paper was presented at [22].

2. PRELIMINARIES

In this section we give some basic definitions and present some lemmas which were proved in the companion paper [21].
We denote by \( G = (V, E) \) a graph with vertex set \( V \) and edge set \( E \). The set of vertices and the set of edges of \( G \) are often denoted by \( V(G) \) and \( E(G) \), respectively. A \((\text{two-terminal})\) \textit{series–parallel multigraph} is defined recursively as follows:

1. A graph \( G \) of a single edge is a series–parallel multigraph. The ends \( v_s \) and \( v_t \) of the edge are called the \textit{terminals} of \( G \) and denoted by \( v_s(G) \) and \( v_t(G) \).

2. Let \( G_1 \) be a series–parallel multigraph with terminals \( v_s(G_1) \) and \( v_t(G_1) \), and let \( G_2 \) be a series–parallel multigraph with terminals \( v_s(G_2) \) and \( v_t(G_2) \).
   
   (a) A graph \( G \) obtained from \( G_1 \) and \( G_2 \) by identifying vertex \( v_s(G_2) \) with vertex \( v_t(G_2) \) is a series–parallel multigraph whose terminals are \( v_s(G) = v_s(G_1) \) and \( v_t(G) = v_t(G_2) \). Such a connection is called a \textit{series connection}, and \( G \) is denoted by \( G = G_1 \cdot G_2 \).

   (b) A graph \( G \) obtained from \( G_1 \) and \( G_2 \) by identifying \( v_s(G_1) \) with \( v_s(G_2) \) and \( v_t(G_1) \) with \( v_t(G_2) \) is a series–parallel multigraph whose terminals are \( v_s(G) = v_s(G_1) = v_s(G_2) \) and \( v_t(G) = v_t(G_1) = v_t(G_2) \). Such a connection is called a \textit{parallel connection}, and \( G \) is denoted by \( G = G_1 \parallel G_2 \). (See Fig. 1.)

A series–parallel multigraph \( G \) can be represented by a “binary decomposition tree” \[17\]. Figure 2 illustrates a series–parallel multigraph \( G \) and its binary decomposition tree \( T_b \). Labels s and p attached to internal nodes in \( T_b \) indicate series and parallel connections, respectively, and nodes labeled s and p are called \( s \)- and \( p \)-nodes, respectively. A node \( u \) of tree \( T_b \) corresponds to a subgraph of \( G \), which is denoted by \( G_u \). A leaf of \( T_b \), in particular, represents a subgraph of \( G \) induced by two vertices, that is, a set of multiple edges. The set of multiple edges joining vertices \( u \) and \( v \) is denoted by \( E(u, v) \).

The terminals \( v_s(G) \) and \( v_t(G) \) of \( G \) are often denoted simply by \( v_s \) and \( v_t \). We denote by \( d(v) \) the degree of vertex \( v \in V \) in \( G \). The maximum

\[ d(v) = \text{degree of vertex } v \in V \text{ in } G \]

\[ \text{maximum degree} \]
degree of \( G \) is denoted by \( \Delta \). Let \( d(G) = d(v_i) + d(v_j) \), \( \delta\gamma_i(G) = \min(d(v_i), d(v_j)) \) and \( \Delta_k(G) = \max(d(v_i), d(v_j)) \). Thus \( d(G) = \delta\gamma_i(G) + \Delta_k(G) \).

An edge-coloring of \( G \) is an assignment of colors to the edges in \( G \) such that no two adjacent edges have the same color. Figure 3 depicts an edge-coloring of a graph \( G \) in Fig. 2a. The chromatic index \( \chi'(G) \) of a graph \( G \) is the minimum number of colors used by an edge-coloring of \( G \). The number of colors used by an edge-coloring \( \varphi \) is denoted by \( \#\varphi \). Let \( r(\varphi) \) be the number of colors appearing at both \( v_i \) and \( v_j \). For the edge-coloring \( \varphi \) in Fig. 3, only colors 1, 2, and 3 appear at both \( v_i \) and \( v_j \), and hence \( r(\varphi) = 3 \).

The chromatic index \( \chi'(G) \) of a series–parallel multigraph \( G \) cannot be computed directly from \( \chi'(G_1) \) and \( \chi'(G_2) \) when \( G = G_1 \parallel G_2 \) or \( G = G_1 \cdot G_2 \). So we introduced a new invariant \( \chi'(G, i) \) to compute \( \chi'(G) \) by means of a dynamic programming algorithm [21]. For a nonnegative integer \( i \), \( \chi'(G, i) \) is defined to be the minimum number of colors used by an edge-coloring of \( G \) such that exactly \( i \) common colors appear at both \( v_i \) and \( v_j \), that is, \( \chi'(G, i) = \min(\#\varphi \mid \varphi \) is an edge-coloring of \( G \) with
\(r(\phi) = i\) if there exists such a coloring; otherwise, define \(\chi'(G, i) = \infty\).

Clearly \(\chi'(G) = \min \{ \chi'(G, i) \mid 0 \leq i \leq \delta'_n(G) \}\), and \(\chi'(G, i) = \infty\) if \(i < \|E_{\nu_p, r}\|\) or \(i > \delta'_n(G)\). For the graph \(G\) depicted in Fig. 2a, \(\chi'(G, 0) = \infty\), \(\chi'(G, 1) = 7\), \(\chi'(G, 2) = 6\), \(\chi'(G, 3) = 6\), \(\chi'(G, 4) = 7\), \(\chi'(G, 5) = \infty\), and consequently \(\chi'(G) = 6\).

We showed in the companion paper [21] that \(\chi'(G, i)\) is a convex and “unit-staircase” function with respect to \(i\), as illustrated in Fig. 4. Therefore one can consider a kind of inverse functions of \(\chi'(G, i)\), which are called \(i_{\min}(G, j)\) and \(i_{\max}(G, j)\) and defined for an integer \(j\) as follows:

\[
i_{\min}(G, j) = \begin{cases} \min \{ i \mid \chi'(G, i) \leq j \} & \text{if } j \geq \chi'(G); \\ +\infty & \text{otherwise}, \end{cases}
\]

and

\[
i_{\max}(G, j) = \begin{cases} \max \{ i \mid \chi'(G, i) \leq j \} & \text{if } j \geq \chi'(G); \\ -\infty & \text{otherwise}. \end{cases}
\]

Then \(\chi'(G, i) \leq j\) if and only if \(i_{\min}(G, j) \leq i \leq i_{\max}(G, j)\). The current authors proved that if \(\chi'(G, i) \neq \infty\) then \(\chi'(G, i) \leq \chi'(G) + \delta'_n(G)\) [21]. Therefore it suffices to compute \(i_{\min}(G, j)\) and \(i_{\max}(G, j)\) only for all \(j\) such that \(\chi'(G) \leq j \leq \chi'(G) + \delta'_n(G)\).

Define a \(\min\)-\(\max\) \(\text{triple set} \mathcal{T}(G)\) as follows:

\[
\mathcal{T}(G) = \{(j, i_{\min}(G, j), i_{\max}(G, j)) \mid \chi'(G) \leq j \leq \chi'(G) + \delta'_n(G)\}.
\]

Then one can compute \(\chi'(G)\) from \(\mathcal{T}(G)\) since \(\chi'(G) = \min \{ j \mid (j, i_{\min}(G, j), i_{\max}(G, j)) \in \mathcal{T}(G) \}\). Therefore it suffices to give an NC parallel algorithm to compute \(\mathcal{T}(G)\).

![Illustration for functions \(\chi'(G, i), i_{\min}(G, j), \) and \(i_{\max}(G, j)\).](image-url)
The following two lemmas proved in [21] imply that $\mathcal{R}(G)$ can be found from $\mathcal{R}(G_1)$ and $\mathcal{R}(G_2)$ when $G = G_1 \parallel G_2$ or $G_1 \cdot G_2$.

**Lemma 2.1.** Let $G = G_1 \parallel G_2$ and $j \geq \chi'(G)$; then the following hold:

(a) $i_{\min}(G, j) = \max(i_{\min}(G_1, j) + i_{\min}(G_2, j), d(G) - j)$;

(b) $i_{\max}(G, j) = \min(\delta_s(G), d(G_1) - i_{\min}(G_2, j) + i_{\max}(G_2, j), d(G) - i_{\min}(G_2, j) + i_{\max}(G_1, j))$;

(c) $\chi'(G) = \min\{i \geq b_2 \mid i \geq d(G_1) - i_{\max}(G_2, i) + i_{\min}(G_2, i)\}$, where $b_2 = \max(\Delta_s(G), \chi'(G_1), \chi'(G_2))$ is a trivial lower bound on $\chi'(G)$.

**Lemma 2.2.** Let $G = G_1 \cdot G_2$, $v = v_i(G_1) = v_i(G_2)$, and $j \geq \chi'(G)$; then the following hold:

(a) $i_{\min}(G, j) = \max(j, d(G), d(G_1) - i_{\max}(G_2, j) + i_{\min}(G_2, j), d(G_2) - i_{\max}(G_2, j) + i_{\min}(G_1, j)) - j$;

(b) $i_{\max}(G, j) = \min(\delta_s(G), d(G) - i_{\min}(G_1, j) - i_{\min}(G_2, j), d(G) - i_{\min}(G_1, j) - i_{\min}(G_2, j), j - d(v) + i_{\max}(G_2, j) + i_{\max}(G_1, j))$; and

(c) $\chi'(G) = \max(\chi'(G_1), \chi'(G_2), d(v))$.

### 3. NC Parallel Algorithm

In this section we give a parallel algorithm which decides the chromatic index $\chi'(G)$ of a series–parallel multigraph $G$ in $O(\log n)$ time with $O(n \Delta/\log n)$ processors if a decomposition tree of $G$ is given. The parallel computation model we use is an exclusive-read and exclusive-write parallel random access machine (EREW PRAM). It is known that the binary decomposition tree $T_b$ of $G$ can be found either in $O(\log^2 n + \log |E|)$ time with $O(n + |E|)$ EREW processors [12] or in $O(\log n)$ time with $O(|E| \alpha(|E|, n)/\log n)$ CRCW processors [9], where $\alpha$ is the inverse Ackermann function.

Although the algorithm in this section only decides the chromatic index of $G$, it can be easily modified so that it actually edge-colors $G$ with $\chi'(G)$ colors.

We use a tree contraction algorithm. The tree contraction algorithm originally introduced by Miller and Reif [15] takes $O(\log n)$ time with $O(n)$ processors, where $n$ is the number of nodes in the tree. Several authors [1,
[10, 12–14] have improved the algorithm as follows so that it takes \(O(\log n)\) time with \(O(n/\log n)\) processors. A structure is a triple \((\mathcal{L}, F_{\text{node}}, F_{\text{edge}})\) consisting of a set \(\mathcal{L}\), a node function set \(F_{\text{node}} \subseteq \{f : \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}\}\), and an edge function set \(F_{\text{edge}} \subseteq \{f : \mathcal{L} \rightarrow \mathcal{L}\}\). A bottom-up algebraic computation tree on \((\mathcal{L}, F_{\text{node}}, F_{\text{edge}})\) is a binary tree \(T_b\) such that: each leaf node \(v\) of \(T_b\) is labeled by an element \(L(v) \in \mathcal{L}\); each internal node \(u\) of \(T_b\) is labeled by a function \(f_u \in F_{\text{node}}\); and each edge \(e\) of \(T_b\) is labeled by a function \(f_e \in F_{\text{edge}}\). The label \(L(u) \in \mathcal{L}\) of each internal node \(u\) of \(T_b\) is recursively defined as

\[
L(u) = f_u\left(f_{e_1}(L(v_1)), f_{e_2}(L(v_2))\right),
\]

where \(v_1\) and \(v_2\) are the left and right children of \(u\) in \(T_b\), \(e_1 = (v_1, u)\), and \(e_2 = (v_2, u)\). The bottom-up algebraic tree computation (BATC) problem on \(T_b\) is to compute \(L(u)\) for the root \(u\) of \(T_b\). In order to solve the BATC problem in parallel, He and Yesha introduced the following shunt operation [13]. Let \(u\) be a node of \(T_b\) with left child \(v_1\), right child \(v_2\), and parent \(w\). Let \(e_1 = (v_1, u)\), \(e_2 = (v_2, u)\), and \(e_0 = (u, w)\). Suppose \(v_1\) is a leaf node (Fig. 5). A shunt operation on \(v_1\) is defined as follows: delete \(v_1\) and \(u\) from \(T_b\); make \(v_2\) the left child of \(w\) with a new edge \(e = (v_2, w)\); and assign to \(e\) a function \(f_e\) defined by

\[
f_e(L) = f_{e_0}(f_{e_1}(L(v_1)), f_{e_2}(L(v_2)))
\]

for variable \(L \in \mathcal{L}\). If the right child \(v_2\) of \(u\) is a leaf node, then a shunt operation performed on \(v_2\) is defined similarly. Clearly a shunt operation does not affect subsequent evaluation on \(T_b\). The following elegant tree-
contraction algorithm solves the BATC problem [1, 10, 12–15]:

**Tree Contraction Algorithm**

for each leaf \( v \), in parallel, do

\[
\text{index}(v) \leftarrow \text{left-to-right leaf index } v;
\]

(the leaves are numbered in left-to-right order)

repeat \([\log n] - 1\) times

for each leaf \( v \) with odd index(\( v \)) which is the left child of \( v \)'s parent, in parallel, do

if the \( v \)'s parent is not the root of \( T_b \) then shunt \( v \);

for each leaf \( v \) with odd index(\( v \)) which is the right child of \( v \)'s parent, in parallel, do

if the \( v \)'s parent is not the root of \( T_b \) then shunt \( v \);

for each leaf \( v \), in parallel, do

\[
\text{index}(v) \leftarrow \text{index}(v)/2
\]

end-repeat

compute the label of the root in \( T_b \) \( \{T_b \text{ has only three nodes}\} \).

The following theorem was proved in [1, 10, 12–14].

**Theorem 3.1.** The tree contraction algorithm correctly solves the BATC problem. Moreover, if the evaluation of node and edge functions and the shunt operation can be done by \( p \) processors in \( O(1) \) time, this algorithm can be implemented in \( O(\log n) \) time with \( O \left( p n/\log n \right) \) processors, where \( n \) is the number of nodes in \( T_b \).

We are now ready to use the tree contraction algorithm to decide the chromatic index of series–parallel multigraphs \( G \). We assume that a binary decomposition tree \( T_b \) of \( G \) is given. Furthermore we may assume that \( d(v_i(G_u)), d(v_j(G_u)), d(G_u), \delta_u(G_u), \text{ and } \Delta_u(G_u) \) have been known for each node \( u \) of \( T_b \) since they can be easily computed in \( O(\log n) \) time with \( O(n/\log n) \) processors. In order to decide \( \chi'(G) \), we make a “bottom-up” traversal on \( T_b \). We compute the label \( L(u) = (\chi'(G_u), \mathcal{A}(G_u)) \) when we visit a node \( u \) of \( T_b \). This computation can be translated to a bottom-up algebraic tree computation on \( T_b \) as follows. Since \( \chi'(G) \leq 2\Delta \), we have \( \chi'(G, i) \leq \chi'(G) + \Delta \leq 3\Delta \) if \( \chi'(G, i) \neq \infty \). We modify the definition of \( \mathcal{A}(G) \) so that \( L(u) \) can be efficiently computed in parallel, as follows:

\[ \mathcal{A}(G) = \{ (j, i_{\min}(G, j), i_{\max}(G, j)) \mid 1 \leq j \leq 3\Delta \} \]

Let \( I = \{0, 1, \ldots, \Delta, -\infty, +\infty\} \) and \( J = \{1, 2, \ldots, 3\Delta\} \), then \( \mathcal{L} = J \times (J \times I)^{3\Delta} \) since \( \chi'(G) \in J \text{ and } \mathcal{A}(G) \in (J \times I)^{3\Delta} \). For each leaf node \( u \) of \( T_b \), \( \chi'(G_u) \) is equal to the number of edges in \( G_u \) and \( \mathcal{A}(G_u) = \{ (j, i_n, i_x) \mid 1 \leq j \leq 3\Delta \} \) can be decided as follows: if \( j \geq \chi'(G_u) \) then \( i_n = i_x = \chi'(G_u) \); otherwise \( i_n = +\infty \) and \( i_x = -\infty \). (The subscript “\( n \)” indicates min, and “\( x \)” max.) For an edge \( e \) of \( T_b \), the edge function \( f_e \) is initially an identity function, that is, \( f_e(L) = L \) for variable \( L \in \mathcal{L} \). The node function \( f_u \) of a \( p \)-node \( u \)
in $T_b$ is a function given in Lemma 2.1, which computes $L(u) = \{ \chi'(G_u), \mathcal{G}_u \}$ from $L(v_1) = \{ \chi'(G_{v_1}), \mathcal{G}_{v_1} \}$ and $L(v_2) = \{ \chi'(G_{v_2}), \mathcal{G}_{v_2} \}$ where $v_1$ and $v_2$ are the two children of $u$ in $T_b$. Similarly the node function $f_u$ of an $s$-node $u$ in $T_b$ is a function given in Lemma 2.2, which computes $L(u)$ from $L(v_1)$ and $L(v_2)$.

Clearly the size of the binary decomposition tree $T_b$ is $O(n)$ [21]. By Theorem 3.1 the algorithm takes $O(\log n)$ time with $O(\Delta n / \log n)$ processors if the evaluation of node and edge functions and the shunt operation, which involves the composition of two edge functions and one node function, can be done in $O(1)$ time with $O(\Delta)$ processors. We have the following two lemmas.

**Lemma 3.2.** The node function of the algorithm above can be evaluated in $O(1)$ time with $O(\Delta)$ processors.

**Proof.** Let $u$ be a node of the current decomposition tree, and let $v_1$ and $v_2$ be the left child and the right child of $u$ in the original decomposition tree. Let $f_u(L_1, L_2) = \{ \chi'_u, \mathcal{F}_u \}$ for variables $L_1, L_2 \in \mathcal{L}$, where $\chi'_u = \chi'(G_u)$ and $\mathcal{F}_u = \mathcal{G}_u$. The node function $u$ is either an $s$-node or a $p$-node. If $u$ is an $s$-node, then by Lemma 2.2 both $\chi'_u$ and $\mathcal{F}_u$ can be computed from $L_1 = L(v_1) = \{ \chi'_1, \mathcal{F}_1 \}$ and $L_2 = L(v_2) = \{ \chi'_2, \mathcal{F}_2 \}$ in $O(1)$ time with $O(\Delta)$ processors. Therefore we may assume that $u$ is a $p$-node. By Lemma 2.1, $\mathcal{F}_u$ can be computed from $\mathcal{F}_1$ and $\mathcal{F}_2$ in $O(1)$ time with $O(\Delta)$ processors. Thus it suffices to prove that $\chi'_u$ can be computed from $L_1$ and $L_2$ in $O(1)$ time with $O(\Delta)$ processors.

Let

$$J_u = \{ j \mid (j, i_{n_1}, i_{n_2}) \in \mathcal{F}_1, (j, i_{n_2}, i_{n_2}) \in \mathcal{F}_2, j \geq \chi'_1, j \geq \chi'_2, j \geq d(G_{v_1}) - i_{n_1}, j \geq d(G_{v_1}) - i_{n_2} + i_{n_1} \}.$$

Then $\chi'_u = \min\{ j \mid j \in J_u \}$ by Lemma 2.1. Clearly $J_u$ can be found from $L_1$ and $L_2$ in $O(1)$ time with $O(\Delta)$ processors. Values $i_{n_1}$ and $i_{n_2}$ do not increase and $i_{n_1}$ and $i_{n_2}$ do not decrease when $j$ increases. Therefore $J_u$ is a set of consecutive integers, and hence $\chi'_u = \min\{ j \mid j \in J_u \}$ can be computed in $O(1)$ time with $O(\Delta)$ processors.

**Q.E.D.**

**Lemma 3.3.** The following (i)–(iii) hold at all points in the execution of the tree contraction algorithm.

(i) Every edge function $f_e(L) = \{ \chi'_e, \mathcal{F}_e \}$ takes the following form for variable $L = \{ \chi' \}$:

(a) $\chi'_e = \min\{ j \mid (j, i_e, i_s) \in \mathcal{F}, j \geq c_1(j), j \geq \chi', j \geq c_2(j) + i_e, j \geq c_3(j) - i_s \}$, and
(b) $\mathcal{T}_e = \{(j, i_n, i_s) \mid (j, i_n, i_s) \in \mathcal{T}\}$, $i_{ne} = \max(c_4(j), c_5(j) + i_n, c_6(j) - i_s)$, and $i_{xe} = \min\{c_7(j), c_8(j) - i_n, c_9(j) + i_s\}$, where $c_1(j)$, $c_2(j)$, $c_3(j)$, $c_4(j)$, $c_5(j)$, $c_6(j)$, $c_7(j)$, $c_8(j)$, $c_9(j)$ are constants determined by $j$, and furthermore $c_1(j)$, $c_2(j)$, $c_3(j)$, $c_4(j)$ do not increase and $c_7(j)$, $c_8(j)$ and $c_9(j)$ do not decrease when $j$ increases.

(ii) Every edge function can be evaluated in $O(1)$ time with $O(\Delta)$ processors.

(iii) At each shunt operation the composition of edge function $f_e(L)$ can be done in $O(1)$ time with $O(\Delta)$ processors.

Proof. (i) We prove (i) by an induction on the number $k$ of shunt operations which have been performed.

The lemma trivially holds when $k = 0$; in this case every edge function $f_e$ is an identity function; let $c_1(j), c_2(j), c_3(j), c_4(j)$, and $c_5(j)$ be $-\infty$, let $c_7(j)$ and $c_8(j)$ be $0$, and let $c_5(j)$ and $c_9(j)$ be $\infty$, then $\chi_e' = \chi'$ and $\mathcal{T}_e = \mathcal{T}$.

Assuming that (i) holds just after the $(k - 1)$th shunt operation is performed, we prove that (i) holds just after the $k$th shunt operation is performed.

Let $u$ be a node of the current decomposition tree $T_b$ with left child $v_1$, right child $v_2$ and parent $w$. Let $e_1 = (v_1, u)$, $e_2 = (v_2, u)$ and $e_0 = (u, w)$. Let $v'_1$ and $v'_2$ be the left child and the right child of $u$ in the original decomposition tree. Then either $G_u = G_{v'_1} \parallel G_{v'_2}$ or $G_u = G_{v'_1} \cdot G_{v'_2}$. We give a proof only for the case when $G_u = G_{v'_1} \parallel G_{v'_2}$, the proof for the case when $G_u = G_{v'_1} \cdot G_{v'_2}$ is similar to that for the case when $G_u = G_{v'_1} \parallel G_{v'_2}$. One may assume that $v_1$ is a leaf node of $T_b$. Then $L(v_1)$ has been computed, and hence $f_{e_1}(L(v_1))$ can be evaluated in $O(1)$ time with $O(\Delta)$ processors as we claim in (ii). On the other hand, by the inductive hypotheses the edge function $f_{e_2}(L) = \{\chi_e', \mathcal{T}_e\}$ takes the following form for variable $L = (\chi', \mathcal{T})$:

$$\chi_e' = \min\{j \mid (j, i_n, i_s) \in \mathcal{T}, j \geq c_1(j), j \geq \chi', j \geq c_2(j) + i_n \text{ and } j \geq c_3(j) - i_s\},$$

$$\mathcal{T}_e = \{(j, i_{ne}, i_{se}) \mid (j, i_n, i_s) \in \mathcal{T}\},$$

$$i_{ne} = \max\{c_4(j), c_5(j) + i_n, c_6(j) - i_s\},$$

$$i_{se} = \min\{c_7(j), c_8(j) - i_n, c_9(j) + i_s\}.$$

By Lemma 2.1 one can easily know that, for a constant $f_{e_1}(v_1) = \{\chi_e', \mathcal{T}_e\} = L(v_1)$ and a variable $f_{e_2}(L) = \{\chi_e', \mathcal{T}_e\}$, the node function $f_n(f_{e_2}(L(v_1)))$,
\[ f_{e_1}(L) = \{ \chi', \mathcal{T}_u \} \text{ satisfies:} \\
\chi' = \min \{ j \mid (j, k_1(j), k_2(j)) \in \mathcal{T}_{e_1}, (j, i_{n_e}, i_{x_e}) \in \mathcal{T}_{e_2}, j \geq k_3 - k_4, \]
\[ j \geq \chi', j \geq \chi', j \geq k_5 - k_2(j) + i_{x_e}, \text{ and} \]
\[ j \geq k_6 - i_{x_e} + k_1(j) \}, \]
\[ \mathcal{T}_u = \{ (j, i_{n_u}, i_{x_u}) \mid (j, i_{n_e}, i_{x_e}) \in \mathcal{T}_{e_2} \}; \]
\[ i_{n_u} = \max\{k_1(j) + i_{n_e}, k_3 - j \}; \text{ and} \]
\[ i_{x_u} = \min\{k_4, k_5 - k_1(j) + i_{x_e}, k_6 - i_{n_e} + k_2(j) \}, \]
where \( k_1(j) = i_{\min}(G_{x_1}, j), k_2(j) = i_{\max}(G_{x_1}, j), k_3 = d(G_u), k_4 = \delta_{\min}(G_u), k_5 = \Delta_{\min}(G_u), k_6 = d(G_{x_2}), \) and \( k_7 = d(G_{x_2}). \) Note that \( k_3, \ldots, k_6 \) are constants, \( k_3(j) \) and \( k_6(j) \) are constants determined by \( j \), and \( k_3(j) \) does not increase and \( k_6(j) \) does not decrease when \( j \) increases. Substitute (1)–(4) into (5)–(8), then we have
\[ f_u(f_{e_1}(L(v_1)), f_{e_2}(L)) = \{ \chi', \mathcal{T}_u \}, \]
\[ \chi' = \min \{ j \mid (j, i_{n}, i_{x}) \in \mathcal{T}, j \geq a_1(j), \]
\[ j \geq \chi', j \geq a_2(j) + i_{n}, j \geq a_3(j) - i_{x} \}, \]
\[ \mathcal{T}_u = \{ (j, i_{n_u}, i_{x_u}) \mid (j, i_{n}, i_{x}) \in \mathcal{T} \}, \]
\[ i_{n_u} = \max\{a_4(j), a_5(j) + i_{n}, a_6(j) - i_{x} \}, \text{ and} \]
\[ i_{x_u} = \min\{a_1(j), a_8(j) - i_{n}, a_9(j) + i_{x} \}, \]
where
\[ a_1(j) = \max\{k_3 - k_4, \chi', c_4(j), k_5 - k_2(j) + c_4(j), k_6 - c_2(j) + k_1(j) \}, \]
\[ a_2(j) = \max\{c_2(j), k_5 - k_2(j) + c_5(j), k_6 - c_5(j) + k_1(j) \}, \]
\[ a_3(j) = \max\{c_3(j), k_5 - k_2(j) + c_5(j), k_6 - c_5(j) + k_1(j) \}, \]
\[ a_4(j) = \max\{k_1(j) + c_4(j), k_3 - j \}, \]
\[ a_5(j) = k_1(j) + c_5(j), \]
\[ a_6(j) = k_1(j) + c_6(j), \]
\[ a_7(j) = \min\{k_2, k_5 - k_1(j) + c_7(j), k_6 - c_4(j) + k_2(j) \}, \]
\[ a_8(j) = \min\{k_5 - k_1(j) + c_8(j), k_6 - c_8(j) + k_2(j) \}, \text{ and} \]
\[ a_9(j) = \min\{k_5 - k_1(j) + c_9(j), k_6 - c_9(j) + k_2(j) \}. \]
Clearly $a_1(j), a_2(j), \ldots, a_9(j)$ are constants determined by $j$, and $a_1(j), a_2(j), \ldots, a_8(j)$ do not increase and $a_7(j)$, $a_9(j)$ do not decrease when $j$ increases.

By the inductive hypothesis, the edge function $f_e(L(u)) = \{ \chi'_e, \mathcal{T}_e \}$ takes the following form for variable $L(u) = \{ \chi'_u, \mathcal{T}_u \}$:

\[
\chi'_e = \min \{ j \mid (j, i_{nu}, i_{ux}) \in \mathcal{T}_u, j \geq b_1(j), j \geq \chi'_u, j \geq b_2(j) + i_{nu} \text{ and } j \geq b_3(j) - i_{ux} \},
\]

\[
\mathcal{T}_e = \{ (j, i_{ne}, i_{xe}) \mid (j, i_{nu}, i_{ux}) \in \mathcal{T}_u \},
\]

\[
i_{ne} = \max \{ b_4(j), b_5(j) + i_{nu}, b_6(j) - i_{ux} \}, \quad \text{and} \quad i_{xe} = \min \{ b_7(j), b_8(j) - i_{nu}, b_9(j) + i_{ux} \},
\]

where $b_4(j), b_5(j), \ldots, b_9(j)$ are constants determined by $j$, and $b_4(j), b_5(j), b_6(j), b_7(j), b_8(j), b_9(j)$ do not increase when $j$ increases. Substitute (9)–(13) into (14)–(17), then for variable $L = \{ \chi', \mathcal{T} \}$ we have

\[
f_e(L) = f_e \left( f_e(L(v_1)), f_e(L) \right) = \{ \chi'_e, \mathcal{T}_e \},
\]

\[
\chi'_e = \min \{ j \mid (j, i_n, i_x) \in \mathcal{T}, j \geq d_1(j), j \geq \chi'_u, j \geq d_2(j) + i_n, j \geq d_3(j) - i_x \},
\]

\[
\mathcal{T}_e = \{ (j, i_{ne}, i_{xe}) \mid (j, i_n, i_x) \in \mathcal{T} \},
\]

\[
i_{ne} = \max \{ d_4(j), d_5(j) + i_n, d_6(j) - i_x \}, \quad \text{and} \quad i_{xe} = \min \{ d_7(j), d_8(j) - i_n, d_9(j) + i_x \},
\]

where

\[
d_1(j) = \max \{ b_1(j), a_1(j), b_2(j) + a_4(j), b_3(j) - a_7(j) \},
\]

\[
d_2(j) = \max \{ a_2(j), b_2(j) + a_5(j), b_3(j) - a_9(j) \},
\]

\[
d_3(j) = \max \{ a_3(j), b_2(j) + a_6(j), b_3(j) - a_9(j) \},
\]

\[
d_4(j) = \max \{ b_4(j), b_3(j) + a_4(j), b_6(j) - a_7(j) \},
\]

\[
d_5(j) = \max \{ b_5(j) + a_5(j), b_6(j) - a_8(j) \},
\]

\[
d_6(j) = \max \{ b_7(j) + a_4(j), b_9(j) - a_9(j) \},
\]

\[
d_7(j) = \min \{ b_7(j), b_9(j) - a_4(j), b_9(j) + a_7(j) \},
\]

\[
d_8(j) = \min \{ b_8(j) - a_4(j), b_9(j) + a_6(j) \}, \quad \text{and} \quad d_9(j) = \min \{ b_9(j) - a_6(j), b_9(j) + a_9(j) \}.
\]
Clearly $d_1(j), d_2(j), \ldots, d_9(j)$ are constants determined by $j$, and $d_1(j), d_2(j), \ldots, d_9(j)$ do not increase and $d_1(j), d_3(j)$ and $d_9(j)$ do not decrease when $j$ increases.

(iii) By (i) above the edge function $f_e(L) = \{\chi_e, \mathcal{T}_e\}$ takes the following form for variable $L = (\chi', \mathcal{T})$:

\[
\chi_e' = \min\{j \mid (j, i_n, i_x) \in \mathcal{T}, j \geq c_2(j), j \geq \chi', j \geq c_2(j) + i_n \text{ and } j \geq c_3(j) - i_x\}, \tag{18}
\]

\[
\mathcal{T}_e = \{(j, i_n, i_x) \mid (j, i_n, i_x) \in \mathcal{T}\}, \tag{19}
\]

\[
i_{ne} = \max\{c_4(j), c_5(j) + i_n, c_6(j) - i_x\}, \quad \text{and} \tag{20}
\]

\[
i_{se} = \min\{c_7(j), c_8(j) - i_n, c_9(j) + i_x\}. \tag{21}
\]

where $c_1(j), c_2(j), \ldots, c_9(j)$ are constants determined by $j$, and furthermore $c_2(j), c_4(j), \ldots, c_9(j)$ do not increase and $c_3(j), c_5(j)$, and $c_7(j)$ do not decrease when $j$ increases. Clearly $\mathcal{T}_e$ can be computed from $L$ in $O(1)$ time with $O(\Delta)$ processors. Therefore it suffices to show that $\chi_e'$ can be computed from $L$ in $O(1)$ time with $O(\Delta)$ processors.

Let

\[
J_e = \{j \mid (j, i_n, i_x) \in \mathcal{T} \}
\]

Then $\chi_e' = \min\{j \mid j \in J_e\}$ by (18). Clearly set $J_e$ can be found from $L$ in $O(1)$ time with $O(\Delta)$ processors. Values $c_4(j), c_5(j)$, and $i_n$ do not increase and $i_n$ does not decrease when $j$ increases. Therefore $J_e$ is a set of consecutive integers, and hence $\chi_e' = \min\{j \mid j \in J_e\}$ can be computed in $O(1)$ time with $O(\Delta)$ processors.

(iii) As shown in the proof of (i), one can compute the intermediate parameters $a_1(j), a_2(j), \ldots, a_2(j)$ and $d_1(j), d_3(j), \ldots, d_9(j)$ for all $j, 1 \leq j \leq 3\Delta$, total in $O(1)$ time using $O(\Delta)$ processors, and hence at each shunt operation the composition of edge function $f_e(L)$ can be done in $O(1)$ time with $O(\Delta)$ processors. Q.E.D.

By Theorem 3.1, Lemmas 3.2 and 3.3, we conclude the following theorem.

**Theorem 3.4.** Let $G$ be a series-parallel multigraph with maximum degree $\Delta$ given by its binary decomposition tree. Then the chromatic index of $G$ can be decided in $O(\log n)$ time with $O(\Delta n/\log n)$ processors.
4. CONCLUSION

In this paper we have given an NC parallel algorithm for the edge-coloring problem on series-parallel multigraphs $G = (V, E)$. Our algorithm is the first NC algorithm for the problem, and takes $O(\log n)$ time with $O(\Delta n / \log n)$ processors. The time complexity is optimal within a constant factor, but the number of processors is expected to be improved to $O(|E| / \log n)$.

Zhou, Nakano, and Nishizeki recently obtained a linear sequential algorithm and an optimal parallel algorithm for the edge-coloring problem on partial $k$-trees for fixed $k$. The algorithms decompose a partial $k$-tree with large maximum degree to several edge-disjoint subgraphs with small maximum degrees [19, 20]. The parallel algorithm takes $O(\log n)$ time with $O((6k)^{k(k+1)/2} n / \log n)$ processors. The constant $(6k)^{k(k+1)/2}$ is bounded but large.

Combining our algorithm for series-parallel multigraphs and the parallel algorithm above for partial $k$-trees [20] one can immediately obtain a truly practical and optimal parallel algorithms to solve the edge-coloring problem for series-parallel simple graphs in $O(\log n)$ time with $O(n / \log n)$ processors. The constant is very small. The combined algorithm is essentially the same as one for partial $k$-trees except that it finds edge-colorings of decomposed series-parallel simple graphs with small $\Delta ( \leq 12)$ in $O(\log n)$ time with $O(\Delta n / \log n)$ processors by the algorithm in this paper in place of the dynamic programming algorithm, which was the obstruction to reducing the constant.

Our algorithm in this paper solves a single particular problem, that is, the edge-coloring problem. However, the methods which we developed in this paper appear to be useful for many other problems, especially for the "edge-partition problem with respect to property $\pi$" which asks that the edge set of a given graph can be partitioned into a minimum number of subsets so that the subgraph induced by each subset satisfies the property $\pi$. For the edge-coloring problem, $\pi$ is indeed a matching.

ACKNOWLEDGMENTS

We thank Dr. Shin-ichi Nakano for various comments and for help in preparing this paper. We thank the anonymous referees for many useful comments. This research is partly supported by a Grant in Aid for Scientific Research of the Ministry of Education, Science, and Culture of Japan under General Research (C) 04650300.
REFERENCES


5. Y. Caspi and E. Dekel, A near-optimal parallel algorithm for edge-coloring outerplanar graphs, manuscript, Computer Science Program, University of Texas at Dallas, Richardson, TX, 1992.


