JOURNAL OF ALGORITHMS 7, 79–104 (1986)

A Better than "Best Possible" Algorithm to Edge Color Multigraphs

DORIT S. HOCHBAUM*

School of Business Administration, University of California, Berkeley, California 94720

TAKAO NISHIZEKI

Department of Electrical Communications, Tohoku University, Sendai 980, Japan

AND

DAVID B. SHMOYS†

Division of Applied Sciences, Harvard University, Cambridge, Massachusetts 02138

Received September 27, 1983

By a result of Holyer, unless $P = NP$, there does not exist a polynomial-time approximation algorithm to edge color a multigraph that always uses fewer than $(\frac{4}{3})\ \chi'$ colors, where χ' is the optimal number of colors. This makes it appear that finding provably good edge colorings is extremely difficult. However, in this paper we present an algorithm to find an edge coloring of a multigraph that never uses more than $\lfloor \frac{9}{8}\chi' + \frac{3}{4} \rfloor$ colors. In addition, if $\chi' \geq \lfloor \frac{9}{8}\Delta + \frac{3}{4} \rfloor$ then the algorithm *optimally* colors the graph in polynomial time. Furthermore, this algorithm never uses more than $(\frac{4}{3})\chi'$ colors and runs in $O(|E|(|V| + \Delta))$ time, where E is the set of edges, and V is the set of vertices.

1. INTRODUCTION

Edge coloring problems arise in many applications, including various scheduling and partitioning problems [2]. The edge coloring problem is

*Supported in part by the National Science Foundation under Grants ECS-8501988 and ECS-8204695.

†Supported in part by the National Science Foundation by a graduate fellowship and under Grant MCS-8311422 and in part by DARPA Order 4031, monitored by Naval Electronic System Command under Contract N00039-C-0235. Present address: Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

0196-6774/86 $3.00

simply stated: color the edges of a given multigraph $G = (V, E)$ using as few colors as possible, so that no two adjacent edges receive the same color. In view of potential applications, it would be useful to have an efficient algorithm capable of coloring any multigraph G with this minimum number of colors (called the *chromatic index* of G and denoted by $\chi'(G)$.) Unfortunately, no such efficient algorithm is known. Moreover, Holyer has shown that the edge coloring problem is *NP*-hard, and therefore it seems unlikely that any such polynomial-time algorithm exists [8].

Let $\Delta(G)$ denote the maximum degree of a vertex. By a well-known result due to Vizing, the chromatic index of a simple graph (i.e., one without multiple edges) is at most $\Delta(G) + 1$ [11]. The situation for graphs that are not simple, often called multigraphs, is more complicated, and it is this case that will be considered here. To avoid confusion, in this paper we will always use the term "simple graph" when we wish to exclude multiple edges.

In the paper mentioned above, Holyer proved a result stronger than the *NP*-hardness of computing the chromatic index; he showed that deciding whether a simple graph has chromatic index of 3 or less is *NP*-complete. This implies that unless $P = NP$, there does not exist an approximation algorithm that finds an edge coloring for multigraphs that uses at most $(\frac{4}{3} - \varepsilon)\chi'(G)$, for any $\varepsilon > 0$ [8]. In contrast to this negative result, there are a number of surprisingly positive results. In this paper we present an $o(|E|(|V| + \Delta))$-time algorithm to find an edge coloring of G that uses no more than $(\frac{4}{3})\chi'$ colors; more surprisingly, it uses at most $\lfloor(9\chi' + 6)/8\rfloor$ colors. This result improves on an algorithm of Nishizeki and Sato [9], which achieves the $(\frac{4}{3})\chi'$ bound as well, but uses as many as $\lfloor(5\chi' + 2)/4\rfloor$ colors, and an algorithm of Hochbaum and Shmoys, in an earlier version of this paper, which uses at most $\lfloor(7\chi' + 4)/6\rfloor$ colors. We will show that this sequence of algorithms can be viewed as part of a family of approximation algorithms, where the ultimate algorithm in this family colors a multigraph using at most $\chi' + 1$ colors.

It should be noted that for the special case of bipartite graphs, multigraphs are no harder than simple graphs, and that such graphs can all be colored with Δ colors. Furthermore, there exist polynomial-time algorithms to find edge colorings using Δ colors, and there is rich literature exploring these questions [1, 3, 7].

The contrasting complexity results have many interesting implications. Following the notation of [4], we let $R_A(G)$ denote the ratio $\chi'_A(G)/\chi'(G)$ where $\chi'_A(G)$ is the number of colors required by approximation algorithm A. Furthermore, we define the *absolute performance ratio*, R_A, of an approximation algorithm A,

$$R_A = \inf\{r \geq 1 | R_A(G) \leq r \text{ for all graphs } G\},$$

and the *asymptotic performance ratio* R_A^∞ for A to be

$$R_A^\infty = \inf\{ r \geq 1 \mid \text{for some } N \in Z^+,\ R_A(G) \leq r \text{ for all } G \text{ s.t. } \chi'(G) \geq N \}.$$

In this notation, the algorithm *color* that we give has $R_{\text{color}} = \frac{4}{3}$ and $R_{\text{color}}^\infty = \frac{9}{8}$. Furthermore, the negative result says that unless $P = NP$, there does not exist a polynomial-time algorithm that achieves $R_A = \frac{4}{3} - \varepsilon$. Most of the negative results for approximation algorithms for other NP-complete problems have been of this form; that is, they deal with the *absolute* performance ratio. Here we have an example of a problem where the negative result belies the true nature of the problem. Therefore, hardness results about approximation algorithms that use the absolute performance ratio should be viewed with slight skepticism since a result with *superior* asymptotic performance might still be obtained. By the same reasoning, results that exclude the possibility of approximation algorithms with certain asymptotic performance ratios (assuming that $P \neq NP$), like those for the vertex coloring and independent set problems [4], are especially desirable.

The algorithm presented here follows a long history of research dating back to Shannon [10], who proved that $\chi' \leq \lfloor 3\Delta/2 \rfloor$. If $\chi'(G - e) = \chi'(G) - 1$ for every edge e in G then G is said to be χ'-*critical*. Most of the earlier results were not concerned with algorithms, but with characterizations of χ'-critical graphs that have chromatic index greater than $\alpha\Delta + \beta$ for some α and β. Recently, Goldberg [5] has shown that if G is a χ'-critical graph such that $\chi(G) > (9\Delta + 6)/8$, then G contains at most 7 vertices. This result does not yield an efficient approximation algorithm, but many of his techniques are useful in the construction of our algorithm. In [6], Goldberg independently claimed algorithmic results similar to those contained here. In Section 4 we will discuss why we believe that our approach would probably lead to a polynomial-time algorithm to edge color multigraphs using at most $\chi' + 1$ colors.

2. The Algorithm in Brief

The most naive approach to edge coloring a graph might be described as follows: choose an arbitrary edge of the graph and, if possible, color it with one of the colors used already; otherwise use a new color to color the edge. The most obvious improvement is to use an interchange approach: for each edge, check if some "simple" recoloring of the colored edges would eliminate the need for an additional color. This simple approach is the basis for most of the results about the chromatic index, and dates back to Shannon, and is the approach that is used here. In order to prove better bounds, the

"simple" recolorings become more complicated. A pidgin PASCAL description of the algorithm is given below; the "simple" recolorings are hidden in the subroutine recolor. The remainder of this paper is dedicated to a description of this procedure.

```
procedure color(G):
begin {G = (V, E)}
  if Δ(G) ≤ 2 then color G with χ'(G) colors by a trivial method (DFS)
    else {Δ ≥ 3}
      begin
        q := ⌊(9Δ + 6)/8⌋; {q colors are currently available}
        G' := (V, ∅):
        for each e ∈ E do
          begin
            G' := G' + e; {add an uncolored edge e = (x, y)}
            let a and b be any missing colors of x and y, respectively;
            recolor(x, y, a, b) {update coloring}
          end
      end
end
```

Before proceeding with a detailed description of the subroutine *recolor*, we first introduce a few preliminary definitions. Assume that $G = (V, E)$ is edge colored with a set of q colors. Throughout this paper, n and m denote $|V|$ and $|E|$, respectively. A color c is said to be *missing* at vertex v if color c is not assigned to any of the edges incident to v. The set of all colors missing at v is called the *missing color set* of v, and is denoted by $M(v)$. The set of colors assigned to the multiple edges joining vertices u and v is denoted by $C(u, v)$. An edge colored c is called a c-edge. For two colors a and b, the edge subgraph of G induced by all of the edges colored a or b is called an *ab-subgraph*, and is denoted by $G[a, b]$. Each connected component of $G[a, b]$ is either a path or a cycle, in which the edges are alternately colored a and b. Such a path (or cycle) is called an *ab-alternating path* (*cycle*). A vertex x is an endpoint of such a path if and only if exactly one of a and b is missing at x. Note that interchanging the colors a and b of the edges in an *ab*-alternating path or cycle yields another q-coloring of G. If $a \in M(x)$ for a vertex x, then *Apath* (x, a, b) denotes the *ab*-alternating path starting from x. As in the algorithm, G' will always denote the graph induced by all of the colored edges and a single uncolored edge (x, y). If, in G', $a \in M(x)$ and $b \in M(y)$, then the *ab*-alternating path between x and y, if any, is called an *ab-critical path*. Note that every critical path contains an odd number of vertices. If H is a subgraph of G, then let $V(H)$ and $E(H)$ denote, respectively, the vertex and edge sets of H. Finally, define a *k-cycle* to be a cycle containing k vertices.

Let us now consider some easy lower bounds on χ'.

FACT 1. $\chi'(G) \geq \Delta(G)$.

FACT 2. *Let* $\mu_k(G)$ *be the maximum number of edges in any subgraph induced on* $2k + 1$ *vertices. Then* $\chi'(G) \geq \lceil \mu_k(G)/k \rceil$.

This follows directly from the fact that each color can be used to color at most k edges of the subgraph. For our purposes it will be convenient to define $\tau(G) = \max_{k=1,2,3} \lceil \mu_k(G)/k \rceil$. The following fact is an immediate corollary of Fact 2.

FACT 3. $\chi'(G) \geq \tau(G)$.

LEMMA 1. *Suppose that the routine recolor has the following property: if a new color is added to the color set, a subgraph induced on* 3, 5, *or* 7 *vertices has been identified that shows that the number of colors previously used is strictly less than* $\tau(G)$. *Then the procedure color uses at most* $\min\{\lfloor (9\chi' + 6)/8 \rfloor, (\frac{4}{3})\chi'\}$ *colors. Furthermore, if* $\chi' \geq \lfloor (9\Delta + 6)/8 \rfloor$, *then color yields an optimal coloring.*

Proof. Consider again the algorithm *color*. We assume that if the number of colors is ever increased by the routine *recolor* then the number of colors previously used was strictly less than $\tau(G)$. Given this, the algorithm has the very strong property that if an additional color is introduced (beyond the initial $\lfloor (9\Delta + 6)/8 \rfloor$), then the coloring produced is *optimal* (by Fact 3). Thus, the only other case is that exactly $\lfloor (9\Delta + 6)/8 \rfloor$ colors are used; but by Fact 1, this implies that at most $\lfloor (9\chi' + 6)/8 \rfloor$ colors are ever used. Furthermore, if $\Delta \leq 2$ we color optimally, and if $\Delta \geq 3$, $\lfloor (9\Delta + 6)/8 \rfloor \leq 4\Delta/3 \leq 4\chi'/3$. Therefore, to prove the claimed performance guarantees we need only show that *recolor* satisfies the property mentioned in Lemma 1.

There are five basic cases handled by *recolor*:

Case 1. Either x and y have a common missing color, or G' has no *ab*-critical path.

Case 2. The *ab*-critical path Q contains two vertices u and v that have a common missing color.

Case 3. The *ab*-critical path Q contains seven vertices.

Case 4. The *ab*-critical path Q contains five vertices.

Case 5. The *ab*-critical path Q contains three vertices.

We first show that these five cases are indeed sufficient. Note that if *recolor*

is called, then $\Delta \geq 3$, and therefore $\lfloor (9\Delta + 6)/8 \rfloor \geq \Delta + 1$. Thus, every vertex of G' has at least one missing color, whereas the endpoints of the uncolored edge of G', (x, y), have at least two missing colors. Using a straightforward counting argument, we obtain the following lemma.

LEMMA 2. *Suppose that* $\Delta(G) \geq 3$, *and that all of the edges of* G' *except* $e = (x, y)$ *are colored with at least* $\lfloor (9\Delta + 6)/8 \rfloor$ *colors. Let* S *be any subset of* V *such that no two vertices have a common missing color:*

(a) *If* $x, y \in S$, *then* $|S| \leq 8$.

(b) *If* $x \in S$, *then* $|S| \leq 16$.

Proof. (a) Assume that q colors are used, where $q \geq \lfloor (9\Delta + 6)/8 \rfloor$. Clearly, $|M(x)|$ and $|M(y)|$ are each at least $q - \Delta + 1$, and $|M(v)| \geq q - \Delta$ for every vertex v other than x and y. Since $\Sigma_{v \in S} |M(v)| \leq q$, we have that

$$2(q - \Delta + 1) + (|S| - 2)(q - \Delta) \leq q,$$

and using straightforward manipulations we obtain

$$|S| \leq (q - 2)/(q - \Delta).$$

Noting that $\Delta \geq 3$ and $q > (9\Delta + 6)/8 - 1$, we get that $|S| < 9$.

(b) This proof is almost identical to (a). □

Lemma 2 implies that if the *ab*-critical path Q contains nine or more vertices, then Q contains two vertices that have a common missing color among the first 17 vertices, and hence Case 2 must occur. Thus, if Cases 1 and 2 do not occur, then Q contains either seven, five, or three vertices (Cases 3, 4, and 5, respectively).

3. THE ALGORITHM IN DETAIL

In this section we give a explicit description of the algorithm for each of the five cases.

Case 1. Either x and y have a common missing color or G' has no *ab*-critical path. In this case we execute the following procedure *npath*:

```
procedure npath(x, y, a, b):
begin
  if G' has no ab-critical path then
    begin
      Q := Apath(x, a, b):
```

```
      {Q does not end at y, a ∈ M(x) and b ∈ M(y)}
      interchange the colors a and b of Q
    end;
  {x and y now have a common missing color}
  assign a common missing color of x and y to the uncolored edge
  e = (x, y);
  {the resulting coloring is a q-coloring of the entire graph G'.}
end
```

Case 2. The *ab*-critical path Q contains two vertices u and v with a common missing color c. We reduce this case to Case 1 above. If u is followed by v on the path Q, then one can break the critical path Q simply by recoloring the edge (u, v) with color c. There no longer exists an *ab*-critical path in the resulting q-coloring of G', so *npath* is applicable. If u is not followed by v on the path Q, the situation is slightly more complicated. The essence of procedure *cpath* given below is that the coloring of G' can be altered so that two new vertices of Q have a common missing color and these new vertices do appear consecutively on Q:

```
procedure cpath(x, y, a, b):
begin
  find two vertices u and v on Q with a common missing color c such
  that no vertex between u and v has a missing color c;
  {assume that x, u, v, and y appear on Q in that order}
  {possibly x = u or v = y}
  while u is not followed by v on Q do
    begin
      let u' be the vertex following u on Q (traversing Q from x);
      let c' be any missing color of u';
      interchange the colors of Apath(u', c', c);
      if Apath(u', c', c) ends at u then u := u' else v := u'
      {new u and v have a common missing color c, and the distance
      between them decreases by at least one.}
    end;
  {Q has two successive vertices u and v of a common missing color c}
  assign color c to the edge (u, v) on Q;
  {there is no ab-critical path}
  npath(x, y, a, b)
end
```

Before presenting the details for the remaining three cases, we give two useful subroutines. Suppose that there are two critical paths Q and R together with two vertices u and v, where u is on Q and v is on R, and u

and v have a common missing color. Then the following procedure *twopath* completes the coloring of G' by recoloring some alternating paths:

procedure *twopath*(Q, R, u, v, c):
begin
{Q and R are critical paths from x to y, $u \in Q$, $v \in R$, and $c \in M(u) \cap M(v)$}
assume that neither Q nor R contains two vertices of a common missing color;
{Otherwise execute *cpath*. Thus $u \notin R$, $v \notin Q$.}
let Q be an *ab*-critical path, $a \in M(x)$, $b \in M(y)$;
let R be an *fg*-critical path, $f \in M(x)$, $g \in M(y)$;
{$a \neq f$ or $b \neq g$}
assume that $a \neq f$; {otherwise, interchange the roles of x and y}
let u' be the vertex of Q next to x;
let v' be the vertex of R next to x;
let c' and c'' be arbitrary missing colors of u' and v', respectively;
{if $u' = v'$, let $c' = c''$}
if [$u \neq u'$ **and** *Apath*(u', c', c) does not end at u] **or**
[$v \neq v'$ **and** *Apath*(v', c'', c) does not end at v]
then
begin
{Note that if $u' = v'$, then $u' \neq u, v$, and one of the above conditions must hold, since u and v are assumed distinct}
assume WLOG that the first condition holds;
{clearly they are symmetric}
interchange the colors of *Apath*(u', c', c);
{$c \in M(u') \cap M(u)$}
cpath(x, y, a, b)
end
else
begin
if $u \neq u'$ **then** interchange the colors of *Apath*(u', c', c);
if $v \neq v'$ **then** interchange the colors of *Apath*(v', c'', c);
{$c \in M(u') \cap M(v')$}
if *Apath*(x, a, c) ends at v' **then**
begin
recolor *Apath*(x, a, c);
{$c \in M(x)$ and $a \in M(v')$}
assign color c to the edge (x, u') of Q;
assign color b to the uncolored edge (x, y)
end

```
    else {Apath(x, a, c) does not end at v'}
      begin
        recolor Apath(x, a, c);
        {Since a, c ≠ f, g, R is still a critical path. R contains two
        vertices x and v' of a common missing color c}
        cpath(x, y, f, g)
      end
  end
end
```

Recall that q is the number of colors currently available for the edge coloring. Suppose that the *ab*-critical path Q between x and y does not contain two vertices of a common missing color. Let H be the subgraph of G' induced by the vertices of Q. An edge of G' is said to *leave* H at vertex z if it joins a vertex z in H to a vertex not in H.

By Fact 2, if $q < |E(H)|/((|V(H)| - 1)/2)$, then $q < \tau(G)$ and hence we can and must introduce a new color $q + 1$ for (x, y). Thus one may assume that

$$q \geq \frac{|E(H)|}{(|V(H)| - 1)/2}.$$

Since $|V(H)|$ is odd, each of the q colors is assigned to at most $(|V(H)| - 1)/2$ edges of H. Since H has an uncolored edge (x, y), there is a color c assigned to at most $(|V(H)| - 1)/2 - 1$ edges of H. Thus, H contains three or more vertices at which no induced c-edge is incident. At most one of them is missing color c within G'. Therefore, there must be at least two c-edges that leave H. (Furthermore, if c is not missing at any vertex of H, then at least three c-edges leave H.) In this case the following procedure *leave* modifies the coloring of G' so that an edge colored with a missing color of x or y leaves H:

```
procedure leave:
begin
  {q ≥ |E(H)|/((|V(H)| - 1)/2). There are two or more edges leaving H
  that are colored with the same color.}
  if G' has no edge leaving H that is colored with a missing color of a
  vertex of H
    then
    begin
      {G' has three or more c-edges leaving H, where c is not missing at
      any vertex of H}
      let (u, u') be a c-edge leaving H; {u ∈ H, u' ∉ H}
      let f be a missing color of u different from a, b and c;
```

```
    {Note that x and y have two missing colors, so that if no such f
    exists, we could use cpath}
    let (v, v') be the final c-edge of Apath(u, f, c) such that v ∈ H and
    v' ∉ H as the path is traversed from u;
    if u = v then interchange the colors of Apath(u, f, c)
                {c ∈ M(u) and there is a c-edge leaving H}
      else {u ≠ v}
        begin
          let g be a missing color of v different from a, b, c and f;
          {If no such color exists, this is an easier case and cpath could
          be used}
          {since no f - or g-edge leaves H, Apath(v, g, f) does not
          leave H and must end at u}
          interchange the colors of Apath(v, g, f); {f ∈ M(v)}
          interchange the colors of Apath(v, f, c)
          {c ∈ M(v) and there is a c-edge leaving H}
        end
    end
  {G' has an edge leaving H that is colored with a missing color of a
  vertex of H}
  redefine color c and vertices u, v ∈ H and u' ∉ H so that c ∈ M(v)
  and a c-edge (u, u') leaves H;
  if v ≠ x, y then
    begin
      assume WLOG that vertices x, v, u, and y appear on Q in this
      order;
      {otherwise interchange the roles of x and y}
      let w be the vertex preceding v on Q (traversing Q from x);
      erase the color of edge (w, v) of Q;
      interchange the colors of the subpath of Q between x and w;
      color (x, y) with b;
      {edge (w, v) is an uncolored edge, c ∈ M(v) and there exists a
      c-edge (u, u') leaving H}
      redefine (w, v) to be the uncolored edge of G'
    end
  {color c is missing at the new x or y, and a c-edge leaves H}
end
```

We are now ready to present the details of the remaining three cases.

Case 3. The *ab*-critical path Q contains seven vertices. We can reduce this case to one of the preceding two cases by the following procedure

seven:

```
procedure seven:
begin
  if q < |E(H)|/((|V(H)| − 1)/2) then
    begin
      {q < τ(G)}
      assign a new color q + 1 to edge (x, y);
      q := q + 1
    end
  else {there are two or more edges leaving H that are colored with the
        same color}
    begin
      leave;
      assume that c ∈ M(x) and the c-edge (u, u') leaves H;
      {c is not missing at any vertex of Q except x. At least two c-edges
      leave H.}
      let Q be (x, z1, z2, ..., z5, y);
      {Note that u = z1, z2, ..., z5 or y}
      if there is no cb-critical path then npath(x, y, c, b)
      else {there is a cb-critical path R}
        if the cb-critical path R contains two vertices of a common
        missing color
          then cpath(x, y, c, b)
          else {R contains at most seven vertices}
            if there exist two vertices, w, v that have a common missing
              color where w is on Q and v is on R;
            then twopath
            else {Since |V(Q ∪ R)| ≤ 8 by Lemma 2, R does not leave
                 H. (Note that it is impossible to add just one vertex by
                 leaving H, since b is a color common to both Q and R.)
                 Hence u ≠ x, y, z1}
              if a component (≠ R) of G'[b, c] which is not a 2-cycle
                contains exactly one of the two b-edges (z2, z3) and
                (z4, z5)
              then
                begin
                  {the component has two c-edges leaving H}
                  interchange the colors of the component;
                  {the ab-critical path, if any, has nine or more vertices}
                  if there exists no ab-critical path then npath(x, y, a, b)
                  else cpath(x, y, a, b) {the ab-critical path contains
                  two vertices of a common missing color}
```

```
            end
        else
            if a component of G'[a, c] which is not a 2-cycle contains exactly one of the three a-edges then
                begin
                    interchange the colors of the component;
                    {the ab-critical path, if any, contains nine or more vertices}
                    if there exists no ab-critical path
                    then npath(x, y, a, b) else cpath(x, y, a, b)
                end
            else
                begin
                    {By a straightforward case analysis, it can be seen that edge (y, z1) must be colored c}
                    let S be the bc-alternating path or cycle containing z2, z3, z4, and z5; {We further divide this into 2 cases depending on the order of the appearance of z2, z3, z4, and z5 in S. The patient reader can find the details of these cases in the Appendix.}

                    Case 3.1 and Case 3.2

                end
    end
end
```

Case 4. The *ab*-critical path Q contains five vertices. Let $Q = (x, z_1, z_2, z_3, y)$. We can reduce this case to one of the preceding three cases by the following procedure *five*:

```
procedure five:
begin
  if Q < |E(H)|/((|V(H)| - 1/2) then
    begin
      {q < τ(G)}
      assign a new color q + 1 to edge (x, y);
      q := q + 1
    end
  else {there are two or more edges leaving H and colored with the same color}
    begin
      leave;
      assume WLOG that c ∈ M(x) and a c-edge leaves H;
```

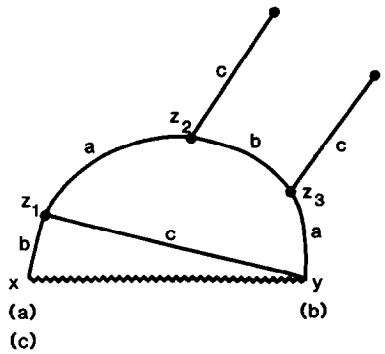

FIGURE 1

{since c is missing at none of the four vertices z_1, z_2, z_3, and y, there are two or four c-edges leaving H}
if there is no cb-critical path **then** *npath*(x, y, c, b)
else {there is a cb-critical path R}
  if R has two vertices of a common missing color
  then *cpath*
  else {R has at most seven vertices}
    if R has seven vertices then *seven*
    else
      if R has three vertices
        then
        {$R = (x, z_1, y)$. Two c-edges leave H at z_2 and z_3 for $c \in M(x)$. See Fig. 1.}
          begin
            interchange the colors of the bc-alternating path or cycle containing z_2 and z_3;
            {the new ab-critical path, if any, has seven or more vertices}
            recolor(x, y, a, b)
          end
        else {R has five vertices}
          if there are two vertices of a common missing color in $V(Q \cup R)$
          then *twopath*
          else
            begin
              let R be (x, z_1, w_2, w_3, y);
              {$w_2, w_3 \notin H$. (z_1, w_2) and (y, w_3) are two c-edges leaving H}
              let T be the subgraph of G' induced by $V(Q \cup R)$;

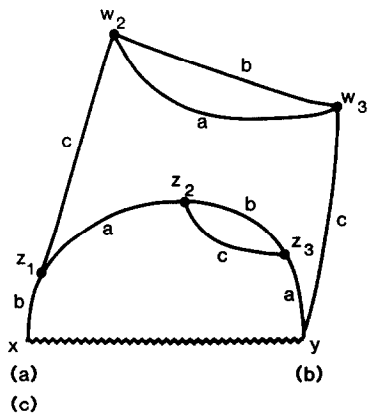

FIGURE 2

{T has exactly seven vertices}
if two c-edges leave H at z_2 and z_3 **then**
  begin
    interchange the colors of the bc-alternating path or cycle containing z_2 and z_3;
    {the ab-critical path, if any, has seven or more vertices}
    recolor(x, y, a, b)
  end
else {a c-edge joins z_2 and z_3}
  if $a \notin C(w_2, w_3)$ **then**
    begin
      {since a is missing at neither w_2 nor w_3, two a-edges leave T at w_2 and w_3}
      interchange the colors of the ab-alternating path or cycle containing w_2 and w_3;
      {the bc-critical path, if any, has seven or more vertices}
      recolor(x, y, b, c)
    end
  else {an a-edge joins w_2 and w_3. See Fig. 2.}
    if $q < |E(T)|/((|V(T) - 1)/2)$ **then**
      begin
        $\{q < \tau(G)\}$
        color (x, y) with a new color $q + 1$;
        $q := q + 1$
      end
    else
      {G' has two edges leaving T of the same color.

```
                    Separating four cases, we can find a q-coloring of
                    G'. Note that leave is not applicable since T is not
                    induced by the vertices of a single critical path. The
                    details are given in the Appendix for the extremely
                    patient reader.}
                    Cases 4.1 through 4.5
                end
    end
end
```

Case 5. The *ab*-critical path Q contains three vertices. Let $Q = (x, z, y)$, and let H be the subgraph of G' induced by the three vertices of Q. We can reduce this case to one of the preceding four cases as follows:

```
procedure three:
begin
  if q < |E(H)|/((|V(H)| - 1)/2) = |E(H)| then
    begin
      {q < τ(G)}
      assign a new color q + 1 to edge (x, y);
      q := q + 1
    end
  else {there are two or more edges leaving H and colored with the same
       color}
       begin
         leave;
         assume WLOG that c ∈ M(x) and two c-edges leave H at y
         and z;
         interchange the colors of the ac-alternating path or cycle contain-
         ing y and z;
         {the new ab-critical path, if any, has five or more vertices}
         recolor(x, y, a, b)
       end
end
```

4. Some Implementation Details

In this section we outline the details needed to show that the algorithm can be implemented in the claimed time and space bounds. The most critical of these details is the manner in which the partial coloring is stored. In addition to the standard adjacency list representation of the graph, a *same-color* list is stored for each color. Each *same-color* list contains all of

the edges assigned a particular color. In addition, there is a list of all of the uncolored edges. The *same-color* lists use $O(m)$ space.

An edge (u,v) assigned color c appears in the two adjacency lists for u and v, and also in the *same-color* list for c. These three elements are linked to each other by pointers so that each can be directly accessed from another. Each element of a *same-color* list also has a pointer to the beginning of the list. These devices clearly use $O(m)$ space in total.

It should be noted that our algorithm does not compute $\tau(G)$ explicitly, because $\tau(G)$ seemingly cannot be computed in $O(m(n + \Delta))$ time. To prove an $O(m(n + \Delta))$ bound for the execution time, it suffices to show that one execution of *recolor* can be done in $O(n + \Delta)$ time since *color* calls *recolor* exactly m times. In what follows we argue that the recoloring can be done in this time.

Let us consider some basic operations that are used repeatedly. First of all, note that the adjacency list representation of $G'[a, b]$ can be constructed in $O(n)$ time by using the same color lists. Since $G'[a, b]$ contains at most n edges, one can find the *ab*-alternating path from any given vertex in $O(n)$ time. The next fundamental operation is interchanging the colors of an alternating path. Once again, by maintaining pointers between corresponding edges in the adjacency list representations of $G'[a, b]$ and G', one can update the coloring of G' in $O(n)$ time. Furthermore, consider the computation of $M(x)$ for some vertex x. Since $\tau(G) \leq \lceil 3\Delta/2 \rceil$ there are $O(\Delta)$ colors in $M(x)$ and the time required to find $M(x)$ is $O(\Delta)$ by just scanning the adjacency list of x.

Consider now Case 1 in detail. The sets $M(x)$ and $M(y)$ can be found in $O(\Delta)$ time, and therefore one can know in $O(\Delta)$ time whether x and y have a common missing color. As discussed above, it takes $O(n)$ time to find the alternating path starting from vertex x. If it does not end at y, the colors can be interchanged in $O(n)$, and since the updating of the *same-color* lists can be done in $O(n + \Delta)$ time, the overall running time for Case 1 is $O(n + \Delta)$.

For Case 2, consider first the time to decide whether Q contains two vertices u and v having a common missing color, and then to find them if they exist. Once again, it requires $O(n)$ time to construct the $G'[a, b]$ subgraph. If Q has at most seven vertices, then the pair of vertices can be found in $O(\Delta)$ time. Otherwise, by Lemma 2(b), Q necessarily contains two vertices of a common missing color among the first 17 vertices. Therefore, this can be done in $O(\Delta)$ time. Next consider the time required for the **while** statement in *cpath*. Since one execution of the **while** statement decreases the distance between u and v by at least one, the statement is executed at most 15 times. One execution of the statement can be done in $O(n + \Delta)$ time using the basic operations discussed above. Therefore, the **while** statement requires $O(n + \Delta)$ time and *cpath* can be implemented to run in $O(n + \Delta)$ time overall.

For the remaining cases we consider a subgraph (H or T) induced by a constant number of vertices (at most seven) plus all of the edges colored in G' with some fixed number of colors (no more than six colors are ever considered in an iteration.) As a result, at most $O(n + \Delta)$ edges are ever considered in these cases. Furthermore, note that although *recolor* is used recursively in these cases, it is impossible for a particular subcase to recur. Therefore, the depth of recursion is bounded by a constant (which turns out to be five). Using many of the techniques discussed thus far, it is a routine exercise to show that these cases can in fact be implemented to run in $O(n + \Delta)$ time. Thus we have the following result.

THEOREM 1. *Given a graph G, the algorithm* color *edge colors G using at most* $\max\{\lfloor(9\Delta(G) + 6)/8\rfloor, \tau(G)\}$ *colors, in* $O(m(n + \Delta))$ *time and* $O(m)$ *space. Thus,* $R_{\text{color}} = \frac{4}{3}$ *and* $R^{\infty}_{\text{color}} = \frac{9}{8}$.

5. CONCLUSIONS

The algorithm presented here is considerably more complicated than the one presented in [9]. Algorithm *color*, and the earlier algorithms with asymptotic performance ratios of $\frac{5}{4}$ and $\frac{7}{6}$, can all be viewed in the following way: first show that if there is a critical path of length at least l, then there is an "easy" recoloring (for some l); then provide a recoloring routine that takes any critical path of length less than l, and either produces an augmented coloring or a longer critical path. (Once the longest critical path exceeds l then we used the "easy" recoloring mentioned above.) For the three algorithms the values of l are 5, 7, and 9, respectively.

Suppose that we have a critical path that contains all of the vertices of the graph to be colored. If this path contains two vertices with a common missing color and we are using $\Delta + 1$ colors, we may still apply procedure *cpath* to update the coloring. Otherwise, let $m_v = |M(v)|$ and d_v be the degree of v. Then if q colors are used in the current coloring,

$$\sum_{v \in V} m_v = 2 + \sum_{v \in V} (q - d_v) \leq q$$

and as result,

$$q \leq \frac{|E|}{(|V| - 1)/2} - \frac{2}{|V| - 1} < \chi'$$

by Fact 2. Therefore, we are free to use an additional color to color the uncolored edge. Therefore, we need only provide the polynomial-time procedure to either extend some critical path or augment the coloring in order to prove the following conjecture.

CONJECTURE. *There exists an approximation algorithm A for edge coloring a multigraph G that uses at most* $\max\{\chi', \Delta + 1\}$ *colors and runs in time that is polynomial in the size of G.*

This conjecture implies that from a computational point of view, the edge coloring of multigraphs is no harder than simple graphs.

APPENDIX

Case 3.1. The four vertices appear in S in one of the two following orders; z_2, z_3, z_5, z_4 or z_3, z_2, z_4, z_5. (See Figs. 3a and b.)

Either $c \notin C(z_2, z_4)$ or $c \notin C(z_3, z_5)$. Therefore, two c-edges leave H either at z_2 and z_4 or at z_3 and z_5. In this case we execute the following:

begin
  interchange the colors of S;
  interchange the colors of the ac-alternating cycle $(z_1, z_2, \ldots, z_5, y)$;
  if there is no cb-critical path **then** *npath*(x, y, c, b)
  else {the cb-critical path has at least nine vertices}
    cpath(x, y, c, b)
end

Case 3.2. The four vertices appear in S in one of the two following orders; z_2, z_3, z_4, z_5 or z_3, z_2, z_5, z_4. (See Figs. 3c and d.)

In this case we can show that two distinct c-edges, (z_2, z_2') and (z_5, z_5') leave H. Suppose to the contrary that $c \in C(z_2, z_5)$. (If the c-edges do not leave H, where could they go?) Then two c-edges leave H at z_3 and z_4, and (z_1, z_2, z_5, y) is an ac-alternating cycle. Therefore the component of $G'[a, c]$ containing the a-edge (z_3, z_4) is neither a 2-cycle nor includes the other two a-edges of Q. This contradicts the assumptions of our procedure. Therefore, we can execute the following procedure in this case:

begin
  let (z_2, z_2') and (z_5, z_5') be two c-edges leaving H;
  if $[a \notin C(z_2', z_5')]$ **or** $[b \in M(z_2') \cup M(z_5')]$ **or** $[b \notin C(z_2', z_5')]$
  then
    begin
      interchange the colors of S;
      {the new coloring of G' satisfies one of the following:
      (a) there is no ab-critical path (if we are lucky)
      (b) the new ab-critical path Q' has nine or more vertices (by $a \notin C(z_2', z_5')$)
      (c) vertex x shares a common missing color c with z_2' or z_5' (by $b \in M(z_2') \cup M(z_5')$)

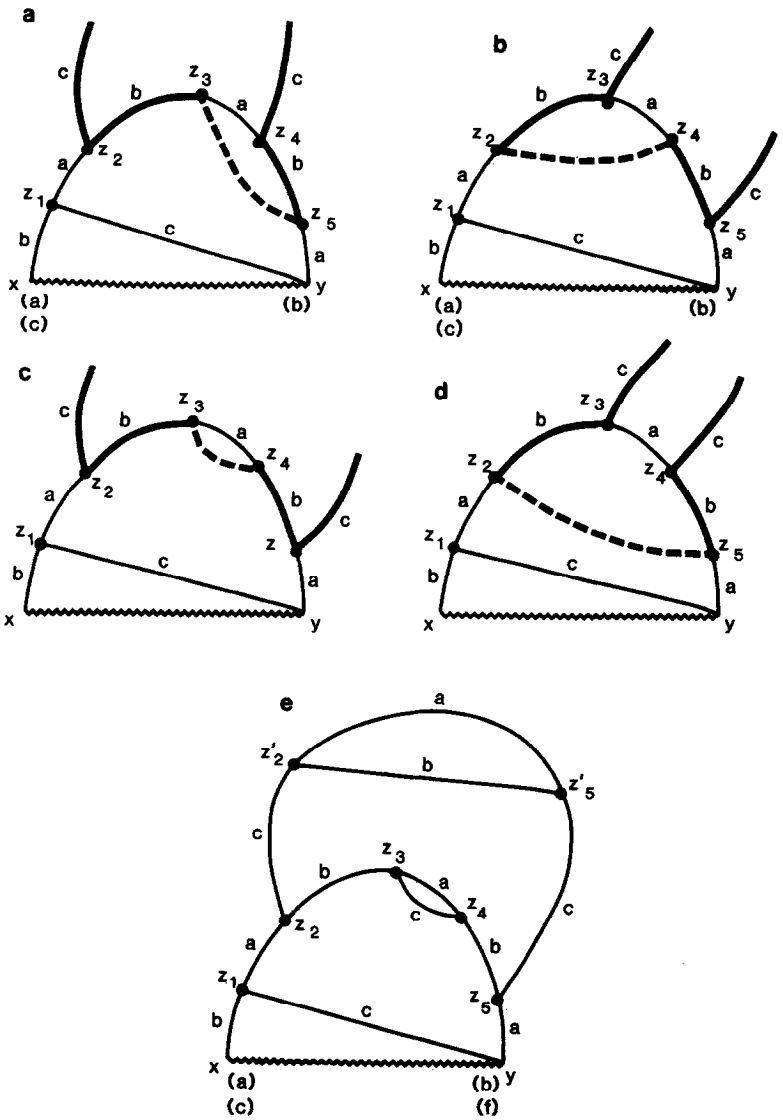

FIGURE 3

(d) the component of $G'[a, c]$ containing the a-edge (z_2', z_5') is neither a 2-cycle nor contains any of the other a-edges of the new ab-critical path $Q' = (x, z_1, z_2, z_2', z_5', z_5, y)$.
(by $a \in C(z_2', z_5')$, $b \in M(z_2') \cup M(z_5')$, and $b \notin C(z_2', z_5')$.)}
recolor(x, y, a, b)

end

else {$a, b \in C(z_2', z_5')$ and as a result of this and the 2-cycle condition used in the procedure it follows that $a, c \in C(z_3, z_4)$. See Fig. 3e.}
begin
let $f \neq b$ be any missing color of y;
if [there is no *af*- or *cf*-critical path] **or** [the *af*- or *cf*-critical path contains two vertices of a common missing color]
then *recolor*(x, y, h, f) {$h = a$ or c as appropriate from above}
else
if two of the four *ab*-, *af*-, *cb*-, and *cf*-critical paths contain two vertices of a common missing color
then *twopath*
else {since the *af*- and *cf*-critical paths do not go out of H (by Lemma 2(a)), an f-edge neither leaves H at z_5 nor joins z_5 with x or z_1; thus an f-edge joins z_5 with either z_2, z_3, or z_4.}
if an f-edge joins z_5 with z_2
then
begin
interchange the colors of the component of $G'[c, f]$ containing the f-edge (z_5, z_2);
if [there is no *af*-critical path] **or** [the *af*-critical path contains two vertices of a common missing color]
then *recolor*(x, y, a, f)
else {the *ab*- and *af*-critical paths contain nine or more vertices in total, so they contain two vertices of a common missing color}
twopath
end
else {an f-edge joins z_5 with z_3 or z_4}
begin
interchange the colors of S;
{the new *ab*-critical path $(x, z_1, z_2, z_2', z_5', z_5, y)$ and the *af*-critical path contain two vertices of a common missing color}
twopath
end
end
end

Case 4.1. An f-edge leaves T for $f \in M(y)$:

begin
if [there is no *af*- or *cf*-critical path] **or**
[the *af*- or *cf*-critical path has two vertices of a common missing color] **or** [the *af*- or *cf*-critical path contains seven vertices]

then *recolor*(x, y, h, f) {where h is either a or c, from above}
else
if two of the four *ab*-, *cb*-, *af*-, and *cf*-critical paths have two vertices of a common missing color
then *twopath*
else {The *af*- and *cf*-critical paths have at most five vertices and do not leave T. Thus, no f-edge leaves T at x, z_3, or w_3. Hence two f-edges leave T at z_1, z_2, or w_2}
begin
{The *af*-critical path does not contain the a-edge (z_1, z_2). Otherwise, the *af*-critical path would leave T.}
interchange the colors of the *af*-alternating path or cycle containing z_1 and z_2; {the *ab*-critical path, if any, has seven or more vertices}
recolor(x, y, a, b)
end
end

Case 4.2. An f-edge leaves T for $f \in M(z_1)$:

begin
erase the color b from edge (x, z_1);
color (x, y) with b;
let (x, z_1) be the new uncolored edge (x', y');
{$a, c \in M(x')$, $b \in M(y')$ and an f-edge leaves T for $f \in M(y')$. Thus Case 4.1 is applicable.}
recolor(x', y', a, b)
end

Case 4.3. An f-edge leaves T for $f \in M(x)$:

begin
if [there is no *fb*-critical path] **or**
[the *fb*-critical has two vertices of a common missing color] **or**
[the *fb*-criticalpath has seven vertices]
then *recolor*(x, y, f, b)
else {the *fb*-critical path has at most five vertices}
if two of the three *ab*-, *cb*- and *bf*-critical paths have two vertices of a common missing color
then *twopath*
else {Since the *fb*-critical path does not leave T, no f-edge leaves T at y or z_1. Hence two f-edges leave T at z_2, z_3, w_2, or w_3.}
begin
assume WLOG that an f-edge leaves T at z_2 or z_3;
interchange the colors of the *bf*-alternating path containing the f-edge leaving T;

```
        {the ab-critical path, if any, has seven or more vertices}
        recolor(x, y, a, b)
      end
end
```

Case 4.4. An f-edge leaves T where f is a missing color of z_2, z_3, w_2, or w_3:

```
begin
  assume WLOG that f ∈ M(w3)
  {Although it is clear that z3 and w3 are completely equivalent, it is not as
  simple to see that w2 is not fundamentally different from w3. However,
  this can be attained by uncoloring the b-edge (x, z1) and coloring (x, y)
  with b. This interchanges the roles of w3 and w2.}
  if [S = Apath(y, b, f) does not end at w3] or [S contains two vertices of
     a common missing color] or [S contains exactly seven vertices] or [S
     and R have two vertices of a common missing color]
  then
    begin
      erase the color c of (y, w3);
      color (x, y) with c;
      recolor(y, w3, b, f)
    end
  else {path S ends at w3 and contains no two vertices of a common
       missing color, and S and R do not have two vertices of a common
       missing color}
    if an f-edge leaves T at w2 or y
    then
      begin
        {|V(T ∪ S)| ≥ 9}
        T ∪ S contains two vertices u and v with a common missing color
        g, where u is in S − T and v is in T − S;
        {v = z2 or z3 since S and R have no two vertices of a common
        missing color. g ≠ a, b, c, f}
        let h ≠ b be a missing color of y; {h ≠ a, b, c, f, g}
        if Apath(y, h, g) does not end at v then
          begin
            interchange the colors of Apath(y, h, g);
            {two vertices v and y on Q have a common missing color g}
            cpath(x, y, a, b)
          end
        else {Apath(y, h, g) ends at v}
          begin
            interchange the colors of Apath(y, h, g);
```

```
            {vertices u and y have a common missing color g}
            erase color c of edge (y, w3);
            color (x, y) with c;
            cpath(y, w3, b, f)
          end
      end
    else
      if an f-edge leaves T at x or z1 then
        begin
          {the bf-alternating path S contains exactly three or five vertices,
          and contains neither x nor z1}
          interchange the colors of S;
          erase the color c of (y, w3);
          color (x, y) with c;
          {the fc-critical path, if any, contains seven or more vertices}
          recolor(y, w3, f, c)
        end
      else {an f-edge leaves T at z2 or z3}
        begin
          {Actually, two f-edges leave T at z2 and z3. An f-edge joins w2
          and either x, z1, or y. If f ∈ C(w2, y) then f ∈ C(x, z1).}
          interchange the colors of S;
          {the af-critical path, if any, contains seven or more vertices}
          recolor(x, y, a, f)
        end
end
```

Case 4.5. Three f-edges leave T where f is not missing at any vertex of T:

```
begin
  let (u, u') be an f-edge leaving T such that u ∈ T − {x, y} and u' ∉ T;
  let g be any missing color of u;
  {g ≠ a, b, c, f. No g-edge leaves T. (Otherwise do one of Cases 4.1–4.4.)}
  S := Apath(u, g, f);
  let (v, v') be the last f-edge of S such that v ∈ T and v' ∉ T as S is
  traversed from u;
  if v = u then
    begin
      interchange the colors of S;
      {Since f ∈ M(u) and an f-edge leaves T, one of Cases 4.1–4.4
      necessarily occurs.}
      recolor(x, y, a, b)
    end
```

else $\{v \neq u\}$
  if $v \neq x$ **then** $\{v \neq u, x\}$
    begin
      let h be any missing color of v such that $h \neq a, b, c$;
      assume WLOG that no h-edge leaves T; {otherwise Cases 4.1–4.4 are applicable}
      { $Apath(v, h, g)$ ends at u and does not leave T since g and h are missing only at u and v respectively, and there is no g- or h-edge leaving T. (Otherwise, one of Cases 4.1–4.4 apply immediately.)}
      interchange the colors of $Apath(v, h, g)$;
      $\{g \in M(v)\}$
      interchange the colors of $Apath(v, g, f)$;
      {Since $f \in M(v)$ and an f-edge leaves T at u, one of Cases 4.1–4.4 necessarily occurs.}
      recolor(x, y, a, b)
    end
else $\{v = x\}$
    if $Apath(x, a, f)$ or $Apath(x, c, f)$ contains a vertex of T other than x **then**
      begin
        assume WLOG that $Apath(x, a, f)$ contains a vertex of T other than x;
        let (r, r') be the last f-edge of $Apath(x, a, f)$ such that $r \in T$ and $r' \notin T$ as $Apath(x, a, f)$ is traversed from x;
        let h be any missing color of vertex r;
        {We can assume that no a- or h-edge leaves T since otherwise, Cases 4.1 to 4.4 are applicable.}
        interchange the colors a and h in T;
        $\{a \in M(r),\ h \in M(x)\}$
        interchange the colors of $Apath(r, a, f)$;
        {Since $f \in M(r)$ and an f-edge leaves the new $T = Apath(x, h, b) \cup Apath(x, c, b)$, one of Cases 4.1–4.4 must occur.}
        recolor(x, y, h, b)
      end
    else {neither $Apath(x, a, f)$ nor $Apath(x, c, f)$ contains a vertex of T other than x}
      if an f-edge, say (s, s'), leaves T at z_2, z_3, w_2, or w_3
      then
        begin
          assume WLOG that $s = w_2$ or w_3;
          {the situation for w_i and z_i is identical}
          interchange the colors of $Apath(x, a, f)$;

```
    if [there is no fb-critical path] or
        [the fb-critical path has seven or more vertices]
        then recolor(x, y, f, b)
        else
          begin
            {Since the fb-critical path contains at most five
            vertices, the path contains none of w2, w3, and s'}
            interchange the fb-alternating path or cycle contain-
            ing w2, w3, and s';
            {the new cb-critical path, if any, has at least seven
            vertices}
            recolor(x, y, c, b)
          end
  end
else
  begin
    {exactly three f-edges (x, x'), (y, y'), and (z1, z1') leave T}
    if b ∉ C(z1', y') then
      begin
        interchange the colors of Apath(x, a, f);
        {the bf-critical path, if any, contains seven or more
        vertices}
        recolor(x, y, b, f)
      end
    else
      begin
        let P be the fb-alternating path (x, z1, z1', y', y);
        {two of the nine vertices of P, Q or R have a common
        missing color}
        if P has two vertices of a common missing color then
          begin
            interchange the colors of Apath(x, a, f);
            recolor(x, y, f, b)
          end
        else
          begin
            assume WLOG that P and R have two vertices of a
            common missing color;
            interchange the colors of Apath(x, a, f);
            recolor(x, y, f, b)
          end
      end
  end
end
```

ACKNOWLEDGMENTS

We would like to thank Professor Nobuji Saito for stimulating discussions on the subject of this paper.

REFERENCES

1. R. COLE AND J. HOPCROFT, On edge coloring bipartite graphs, *SIAM J. Comput.* **11**, (1982), 540–546.
2. S. FIORINI AND R. J. WILSON, "Edge-colouring of Graphs," Pitman, London, 1977.
3. H. N. GABOW AND O. KARIV, Algorithms for edge coloring bipartite graphs and multigraphs, *SIAM J. Comput.* **11**, (1982), 117–129.
4. M. R. GAREY AND D. S. JOHNSON, "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman, San Francisco, 1979.
5. M. K. GOLDBERG, Edge-coloring of multigraphs; Recoloring technique, *J. Graph Theory* **8**, (1984), 121–137.
6. M. K. GOLDBERG, An approximate algorithm for the edge-coloring problem, *in* "Proceedings of the 15th Southeastern Conference on Graph Theory, Combinatorics, and Computing," to appear.
7. T. GONZALEZ AND S. SAHNI, Open shop scheduling to minimize finish time, *J. Assoc. Comput. Mach.* **23**, (1976), 665–679.
8. I. J. HOLYER, The NP-completeness of edge colourings, *SIAM J. Comput.* **10**, (1980), 718–720.
9. T. NISHIZEKI AND M. SATO, An approximation algorithm for edge-coloring multigraphs, preprint, 1983.
10. C. E. SHANNON, A theorem on colouring lines of a network, *J. Math. Phys.* **28**, (1949), 148–151.
11. V. G. VIZING, On an estimate of the chromatic class of a p-graph, *Diskret. Anal.* **3**, (1964), 25–30. [Russian]