ALGORITHMS FOR EDGE-COLORING GRAPHS

H.N. Gabow
T. Nishizeki
O. Kariv
D. Leven
O. Terada

TECHNICAL REPORT 41/85
September1985

Algorithms for Edge-Coloring Graphs

Harold N. Gabow [1]
Takao Nishizeki [2]
Oded Kariv [3]
Daniel Leven [4]
Osamu Terada [5]

ABSTRACT

In this paper we present some algorithms for edge-coloring simple graphs. Three algorithms for edge-coloring a general graph by $d+1$ (or d) colors have complexities of $O(|E||V|)$, $O(|E|.d.log|V|)$ and $O(|E|\sqrt{|V|log|V|})$ respectively (all of them use $O(|E|)$ space). The first algorithm can also find d-coloring for the following families of graphs: (1) all the planar graphs with $d \geq 8$; (2) all the series-parallel graphs with $d \geq 4$; and (3) almost all random graphs. We also show that every series-parallel graph except odd cycles has an edge-coloring with d colors and present some NP-completeness results related to the edge-coloring problem.

1. Computer Science Dept., University of Colorado, Boulder, Colorado 80309, U.S.A.
2. Department of Electrical Communications, Tohoku University, Sendai 980, Japan.
3. Partially affiliated with the Computer Science Dept, Tel-Aviv University, Israel. (part of this work was done while affiliated with the Comp. Sci. Dept. of the Technion, Haifa, Israel).
4. Computer Science Dept., School of Math. Sciences, Tel-Aviv University, Tel-Aviv, Israel.
5. Sapporo Power Office, The Hokkaido Electric Power Co. Inc. Sapporo, Hokkaido 065, Japan.

Algorithms for Edge-Coloring Graphs

Harold N. Gabow [1]

Takao Nishizeki [2]

Oded Kariv [3]

Daniel Leven [4]

Osamu Terada [5]

1. Introduction

The edge-coloring problem is simply stated: Color the edges of a given simple graph G using as few colors as possible, so that no two adjacent edges receive the same color. The problem arises in many applications, including permutation networks [LPV], preemptive scheduling of an open shop [Go],[GS], preemptive scheduling of unrelated parallel processors [LL] and the class-teacher timetable problem [Gt]. In view of the potential applications, it would be useful to have an efficient algorithm capable of coloring any graph G with this minimum number of colors (called the *chromatic index* of G and denoted by $q^*(G)$). Unfortunately no such efficient algorithm is currently known for the general case. Moreover, recent work has shown that the edge-coloring problem belongs to the class of "NP-complete" problems [H], therefore it seems unlikely that any such polynomial-time algorithm exists [AHU], [GJ].

1. Computer Science Dept., University of Colorado, Boulder, Colorado 80309, U.S.A.
2. Department of Electrical Communications, Tohoku University, Sendai 980, Japan.
3. Partially affiliated with the Computer Science Dept, Tel-Aviv University, Israel. (part of this work was done while affiliated with the Comp. Sci. Dept. of the Technion, Haifa, Israel).
4. Computer Science Dept., School of Math. Sciences, Tel-Aviv University, Tel-Aviv, Israel.
5. Sapporo Power Office, The Hokkaido Electric Power Co. Inc. Sapporo, Hokkaido 065, Japan.

Vizing proved that in simple graphs, either $q^*(G) = d$ or $q^*(G) = d+1$ where d is the maximum vertex degree of G ([FW],[V64]). Special cases which can be colored with d colors are bipartite graphs (for which efficient algorithms exist: [GK1], [GK2], [CH]), cubic bridgeless planar graphs (whose edge-coloring in three colors is equivalent to the four color problem) and planar graphs with $d \geq 8$ [FW]. For the case of planar graphs of degree $d = 8$ or 9, the algorithm presented in this paper is the first to be published.

Extension of the edge-coloring problem is the problem of edge-coloring multigraphs. Vizing's Theorem [V65a], [FW] gives bounds on the minimum number of colors which are required for that case and efficient algorithms are present in [GS], [GK2] and in [NS]. Another direction of generalization of the edge-coloring problem is presented in [HK].

In this paper we present several algorithms for edge-coloring simple graphs. In Section 2 we introduce some of the terminology and definitions used throughout the paper. A basic algorithm, COLOR, which is not new, but is an implementation of the standard proof of Vizing's Theorem, is introduced in Section 3. This algorithm colors the edges of an uncolored simple graph with $d+1$ (or d) colors and requires $O(|E||V|)$ time and $O(|E|)$ space. Although this algorithm is fairly well known, it is formulated and included in this paper because it serves as a subprocedure and is used to define some basic data structures used by all the (new) algorithms that we introduce in this paper. A new and efficient algorithm PARALLEL-COLOR that colors the edges of a simple graph with $d+1$ (or d) colors in $O(|E|.d.log|V|)$ time and $O(|E|)$ space is described in Section 4, and another new and efficient algorithm that does the same job in $O(|E|\sqrt{|V|log|V|})$ time and $O(|E|)$ space is described in Section 5. In Sections 6 and 7 we deal with special families of graphs for which d-coloring exists. First, an algorithm ALCOLOR that edge-colors an arbitrary graph G with $d+1$ (or d)

colors in $O(|E||V|)$ time and $O(|E|)$ space is introduced. Although in general this algorithm has worse complexity than the previous algorithms, yet it constitutes a significant contribution to the field: In Section 7, it is shown that ALCOLOR edge-colors with d-colors a large class of graphs, including: (1) all planar graphs with $d \geq 8$; (2) all series-parallel graphs with $d \geq 4$; and (3) almost all random graphs. (note that on planar graphs the complexity of that algorithm is $O(|V|^2)$). It is also shown that every series-parallel graph, except odd cycles, can be edge-colored with d colors (a generalization of Fiorini's result on outerplanar graphs [F]). Although for some of those classes of graphs (e.g. planar graphs of degree 8 or 9) it was already known for some time that they are edge-colorable by d colors, to the best of our knowledge no formal algorithm for that purpose was ever published. We conclude by presenting in Section 8 some NP-completeness results concerning edge-coloring of regular graphs and restricted edge-coloring problems. Some of the results presented in this paper were previously mentioned in [GK3].

We refer the reader to a comment appearing at the end of the paper in which a comperison of this paper with a similar paper by E. Arjomandi [A] is given.

2. Terminology and Definitions

In this section we introduce some of the terminology and definitions that we use throughout this paper.

Throughout the paper $G = G(V,E)$ denotes a given *simple graph* with vertex set V and edge set E, having no multiple or self-loop edges. We denote by $d(v)$ the degree of vertex $v \in V$. The maximum degree of G is denoted by $d(G)$ or simply d. A vertex adjacent with vertex v is called a *neighbor* of v. An edge joining vertices u and v is denoted by uv. The graph obtained from G by deleting (resp. adding) edge uv is denoted by $G-uv$ (resp. $G+uv$). For $S \subset V$, $G-S$

denotes the graph obtained from G by deleting all the vertices in S and the edges adjacent to them. An edge-coloring of G with at most k colors is called a *k-coloring* of G. Suppose that the edges of G are colored with a given set of colors. If color α in the set is not used for any of the edges incident with vertex v, then we say that α is *missing* at v. We denote by $M(v)$ the set of all the colors missing at v (later we shall define a specific data structure to implement $M(v)$). Denote by $G[\alpha,\beta]$ the subgraph of G induced by the edges colored with colors α and β. Clearly each component of $G[\alpha,\beta]$ is a path or cycle, in which edges are colored alternately with α and β. We call such a path (resp. cycle) an $\alpha\beta$-path (resp. $\alpha\beta$-cycle). A vertex v is an end-point of such an $\alpha\beta$-path if and only if either $\alpha\in M(v)$ or $\beta\in M(v)$. Interchanging colors α and β in a component P of $G[\alpha,\beta]$ yields another coloring of G with the same set of colors and is called a *flip* of P.

Suppose that all the edges of G except vw have been colored with a set of $d+1$ colors. Clearly both v and w have at least two missing colors, and each of the other vertices have at least one. We associate with each vertex v *one* of the missing colors of v, denoted by $m(v)$ and called *the* missing color of v (actually, $m(v)$ will be the first color in the list which implements $M(v)$). A *fan sequence* F at w starting with wv is a sequence of distinct edges $wv = wx_0, wx_1, \ldots, wx_s$ such that for each $1 \le i \le s$ wx_i is colored with $m(x_{i-1})$. We say that w is the *center* of the fan and that the x_i's are its leaves. If F is a maximal fan sequence at w, then one of the following must occur:

Case (1): the missing color of x_s is also missing at w, that is, $m(x_s)\in M(w)$; or

Case (2): an edge wx_{j+1} $(0 \le j < s-1)$ of F is colored with $m(x_s)$ (thus $m(x_j) = m(x_s)$).

Note that F consists of a single uncolored edge wv if $m(v)\in M(w)$.

A *shift* of a fan F from x_i means to circularly shift the colors of the edges $wx_0, wx_1, \ldots\ldots, wx_i$. That is, for $0 \leq j < i$ wx_j gets color $m(x_j)$ and wx_i becomes uncolored. This gives another k-coloring of G.

3. Algorithm COLOR

On the basis of a constructive proof of Vizing's theorem ([Bo], [FW]) it is rather easy to give a polynomial-time (say, $O(|E|^2)$) algorithm for coloring a graph with d or $d+1$ colors. However it is less trivial to give more efficient algorithms. In this section we present an algorithm COLOR whose time complexity is $O(|E||V|)$ and whose main significance is in the fact that it serves as a basic subprocedure in all the algorithms presented in this paper and it is used to define some basic data structures that are applied by all the (new) algorithms which are introduced in this work.

Procedure COLOR (G) :

{G is a graph of maximum degree d; The edges of G may all be uncolored or some of them may be colored. The algorithm completes the coloring of the edges of G in $d+1$ colors}

begin

while there exists an uncolored edge vw in G *do*

RECOLOR(vw)

end COLOR;

The procedure RECOLOR(vw) colors an uncolored edge of G in one of the $d+1$ colors.

Procedure RECOLOR(vw)

begin

1. let $F=[wv = wx_0, wx_1, \ldots, wx_s]$ be a maximal fan sequence F at w starting with the uncolored edge wv; Let $\alpha = m(w)$ and $\beta = m(x_s)$.

2. *if* $\beta \in M(w)$

then begin {Case(1)}

3. shift F from x_s; {now x_s is uncolored}

4. color wx_s by β;

end

else begin {Case (2): $\beta \notin M(w)$}

5. let P be the $\alpha\beta$-path that starts at x_s; {P may be empty}

6. *if* P does not reach (and thus terminate at) w

then begin

7. shift F from x_s; {now wx_s is uncolored}

8. flip P; {now $m(x_s) = \alpha$}

9. color wx_s by α;

end

else begin

10. let x_t , $0 \le t < s-1$ be the vertex for which $m(x_t) = \beta = m(x_s)$; {namely, wx_{t+1} has color β and it is the last edge of P}

11. let P' be the $\alpha\beta$-path that starts at x_t;

12. shift F from x_t; {now wx_t is uncolored}

13. flip P'; {now $m(x_t) = \alpha$}

14. color wx_t by α;

end

end

end RECOLOR.

We have the following theorem on COLOR.

Theorem 3.1 Algorithm COLOR edge-colors an arbitrary uncolored graph $G=(V,E)$ with d or $d+1$ colors in $O(|E||V|)$ time, using $O(|E|)$ space.

Proof:

(a) Correctness: omitted since it follows the proof of Vizing's theorem ([Bo], [FW]).

(b) Space: We use the following main data structures.

(1) **Adjacency Lists:** G is represented by adjacency lists, each containing the edges incident with a certain vertex v. These use $O(|E|)$ space.

(2) **An Array of Missing Colors:** The function $m(.)$ is represented by an array $m[.]$ of length $|V|$.

(3) **Color Lists:** Each list is a doubly-linked list and contains the edges colored with the same color. It is convenient also to keep one more list of the uncolored edges. Thus these $d+2$ lists use in total $O(|E|)$ space.

(4) **An Array of Pointers:** This is an array of $|E|$ entries, each of them consists of 3 pointers: The entry which corresponds to the edge $e = uv$ of G contains 2 pointers to the entries in the adjacency lists of u and v where edge e resides, and a third pointer to the appropriate entry in the color lists.

An edge uv colored with α appears in the adjacency lists for u and v and also in the color list for α. These three elements are linked to each other by the uv-th entry in the array of pointers so that each can be directly accessed from another. Clearly these devices use $O(|E|)$ space in total.

(c) Time: Clearly one can initialize the color lists and the array m[.] for a given graph (including an uncolored one) in $O(|E|)$ time. Thus, it suffices to show that one execution of RECOLOR can be done in $O(|V|)$ time, since (for an initially uncolored graph) COLOR repeats RECOLOR $|E|$ times.

We first show that the fan sequence F at w can be found and shifted in $O(d(w))$ time as follows: Before entering RECOLOR(vw) construct an array W of length $d+1$ such that for each color γ incident at w, $W[\gamma]$ contains the edge

incident at w and has color γ; and if $\gamma \in M(w)$ then $W[\gamma]$ is undefined (+). Exploring the adjacency list of w, one can construct W in $O(d(w))$ time. Then one can decide in $O(1)$ time whether a given color γ is missing at w, and also find the edge colored with γ if $\gamma \notin M(w)$. Using arrays $m[.]$ and $W[.]$, one can easily find F and also shift it in $O(d(w))$ time. Note, that Arjomandi [A] uses for this purpose a vertex-color incidence matrix which results in an $O(|V|d)$ term both in the time complexity of the algorithm and in the space it needs (see comment at the end of the paper).

Using the color lists, one can easily constructs the $\alpha\beta$-subgraph $G[\alpha,\beta]$ in $O(|V|)$ time, since it contains at most $|V|$ edges. Furthermore, one can also construct in $O(|V|)$ time an $\alpha\beta$-path P and flip it.

Thus we have shown that one execution of RECOLOR can be done in $O(|V|)$ time. and the whole procedure COLOR (on an uncolored graph) requires $O(|E||V|)$ time.

Q.E.D.

Since RECOLOR is a basic procedure that serves as a subprocedure in all the algorithms presented in this paper, we emphasize the following fact.

Corollary 3.1: RECOLOR(vw) colors an edge vw in $O(|V|)$ time, using $O(|E|)$ space (provided the color lists and m[.] are initialized).

4. Algorithm PARALLEL-COLOR

In this section we present the first of the two new efficient algorithms which color the edges of a (general) graph by $d+1$ colors. This algorithm, PARALLEL-

(+) If at the beginning of COLOR we define the vector W and assign to all $d+1$ entries of W a value indicating that their corresponding color is missing, and each time we return from RECOLOR(vw) we reassign this special value to all $d(w)$ entries which were used in W, then $W[\gamma]$ would also indicate if γ is missing at w. Otherwise, the value of $W[\gamma]$ may be undefined and thus one more check is necessary to find whether color γ is missing at w or not.

COLOR (which is an extension of COLOR), colors edges in "parallel" and has complexity of $O(|E|.d.log|V|)$. The second algorithm EULER-COLOR is presented in the next section.

We start by defining a new type of fan sequence, an "uncolored fan" (or "u-fan"). In order to distinguish a u-fan from the fan which was defined and used in the previous sections, we shall refer in this section to the latter as "colored fan" (or "c-fan"). For convenience we restate here the definition of a c-fan:

(a) An *$\alpha\beta$-c-fan* is a sequence of edges $wx_0, wx_1, \ldots, wx_s$, such that wx_0 is uncolored and each wx_i (for $1 \le i \le s$) has color $m(x_{i-1})$. We also assume that $m(w) = \alpha$ while for each x_i $(0 \le i \le s)$ $\alpha \notin M(x_i)$. Also, $m(x_s) = \beta$ and either $\beta \in M(w)$ (that is, case (1) in RECOLOR) or for some t $(0 \le t < s-1)$ $\beta \in M(x_t)$ (that is, case (2) in RECOLOR). We refer to w as the *center* of the fan and to the x_i's as the fan's *leaves*. A *shift* of a c-fan from x_i means that every edge wx_j (for $1 \le j < i$) gets color $m(x_j)$ while wx_i itself becomes uncolored.

(b) An *$\alpha\beta$-u-fan* consists of a central vertex (root) w and peripheral vertices (leaves) $x_1, x_2, \ldots, x_s$, where $s \ge 2$, such that all edges wx_i $(1 \le i \le s)$ are uncolored and for each x_i $(1 \le i \le s)$ $\alpha \in M(x_i)$ but $\alpha \notin M(w)$ (that is, an edge colored α is incident to w). β is an arbitrary color of the (at least) $s+1$ colors missing at w: $\beta \in \{\gamma_1, \gamma_2, \ldots, \gamma_{s+1}\} \subseteq M(w)$.

The idea behind the definition of the $\alpha\beta$-u-fan is to use RECOLOR in order to augment a large number of $\alpha\beta$-c-fans "in parallel". Namely, to compute a large number of $\alpha\beta$-c-fans and then to augment all of them. For this to work, the fans must be vertex disjoint, for otherwise the augmenting of one fan can destroy (an arbitrary number of) other fans, forcing fans to be recomputed and thus losing the advantage of parallelism (see in this context the comment at the end of the paper regarding Arjomandi's paper [A]). So we need a mechanism to allow the

creation of a large number of vertex disjoint c-fans: This is the u-fan. We see below (procedure MAKE-S) that when c-fans intersect, they can be converted into a u-fan that allows the uncolored edges to be augmented as follows:

Procedure U-AUGMENT(U,$\alpha\beta$)

{U is an $\alpha\beta$-u-fan $wx_1, wx_2, \ldots, wx_s$ where β is some color missing at w}.

begin

1. let P be the $\alpha\beta$-path that starts at w. {P starts with an edge of color α}.
2. flip P;
3. *if* P does not end at x_1
4. *then* color edge wx_1 with α
5. *else* color edge wx_2 with α.

end U-AUGMENT;

We now outline the algorithm which consists of *stages* each of which is characterized by a certain color α. Each stage is further divided into $\alpha\beta$-*substages* , one substage for each pair of colors $\alpha\beta$. In an $\alpha\beta$-substage the algorithm simultaneously augments as many $\alpha\beta$-fans (c-fans and u-fans) as possible. The algorithm executes a stage for each color and then repeats itself until all edges are colored.

Procedure PARALLEL-COLOR

{this algorithm finds a $(d+1)$-coloring on a graph}.

begin

1. *while* there are uncolored edges *do*
2. *for* each color α *do*

begin {lines 3-6 below are α's stage}

3. color by α a maximal number of edges missing α at both ends;
4. *MAKE−S*; {MAKE-S constructs a collection S of vertex-

disjoint c-fans and u-fans of type $\alpha\beta$ (where β ranges over all colors $\beta \neq \alpha$). S contains an uncolored edge incident to each vertex in G that misses α and is on an uncolored edge. Notice that the coloring may change while constructing S}.

5. *for* each color β ($\beta \neq \alpha$) *do*

{line 6 below is $\alpha\beta$'s substage}.

6. augment as many $\alpha\beta$-fans of S as possible;

end

end PARALLEL-COLOR;

We show that PARALLEL-COLOR can be implemented in time $O(|E|.d.log\,|V|)$. For this, we implement a "set" of stages (lines 2-6) to color $\geq \frac{1}{6}$ of the remaining uncolored edges, with one stage (lines 3-6) taking $O(|E|)$ time. This implies that there are $O(log\,|E|)$ "sets" of stages and since each such set contains $d+1$ stages (one stage per color) the total time is $O(|E|.d.log\,|V|)$.

The basic data-structures that we use for PARALLEL-COLOR are the same as those used in COLOR, namely, color lists and an array $m[.]$ of missing colors such that for each $v \in V$, $m(v)$ is one of the colors missing at v. As mentioned we also maintain a list of all the uncolored edges (this list may be regarded as one of the color lists except that its size is $O(|E|)$ rather than $O(|V|)$). More data-structures will be described later.

Using the list of edges colored by α, we can find in $O(|V|)$ time all the vertices missing that color. Thus, step 3 in PARALLEL-COLOR can be done in $O(|E|)$ time (including the updating of the data structures).

The crucial steps are, of course, lines 4 and 6 which we now discuss: Step 4 is a procedure MAKE-S that constructs the set S of vertex-disjoint fans. It forms c-fans one by one, converting intersecting c-fans into u-fans, so that fans stay

vertex-disjoint (see comment at the end of the paper regarding the algorithm in [A]).

Procedure MAKE-S

begin

4.1 $S \leftarrow \Phi$;

4.2 *for* each vertex w, such that w does not belong to any fan in S, w misses α and w is incident to an uncolored edge e *do*

begin

4.3 let F be an $\alpha\beta$-c-fan with $wx_0=e$ (where β is any color that comes up during the construction of the fan);

4.4 *if* no leaf of F is in a fan of S

4.5 *then* $S \leftarrow S \cup \{F\}$;

else begin

4.6 let x_i be the first leaf of F which belongs to another fan F' in S whose center is w';

4.7 *if* F' is an $\alpha\gamma$-c-fan

4.8 *then begin* {by the definition of an $\alpha\gamma$-c-fan x_i must be a leaf of F'}

4.9 shift F from x_i;

4.10 shift F' from x_i;

4.11 let T be a new u-fan with center x_i and leaves w and w';

$\{\alpha \notin M(x_i)\ ,\ \alpha \in M(w)\ ,\ \alpha \in M(w')\}$;

4.12 $S \leftarrow S \cup \{T\} - \{F\} - \{F'\}$;

end

4.13 *else begin* {F' is an $\alpha\gamma$-u-fan and by its definition x_i must be its center}.

4.14 shift F from x_i;

4.15 $S \leftarrow S - \{F\}$;

4.16 enlarge F' by including $x_i w$ in it

(with w as a leaf)

end

end

end

end MAKE-S;

We now prove that MAKE-S fulfills the specifications of line 4 of PARALLEL-COLOR:

Lemma 4.1: After the loop of line 4.2 is executed for $w = w_i$, $1 \leq i \leq k$ for some k, S is a collection of vertex-disjoint $\alpha\beta$-fans (for β ranging over colors other than α), containing an uncolored edge incident to each w_i.

Proof:

By induction (note that because of line 3 of PARALLEL-COLOR, the only possible intersections between F and another fan F' of S are those described in lines 4.8 and 4.13 of MAKE-S).

Q.E.D.

Before we proceed to prove that MAKE-S has time complexity $O(|E|)$, we describe more data-structures needed to implement MAKE-S. Each fan in S can be maintained as an ordered list of its edges plus its center and its type (c- or u-fan). The ordered list is implemented as a doubly linked list so that it can be traversed forward and backwards and so that insertions and deletions from it take $O(1)$ time. Since the fans are vertex-disjoint it is convenient to maintain a vector of length $|V|$, such that its v-th entry, which corresponds to a vertex v, indicates the following: (a) Whether the vertex belongs to a fan in S or not, (b) If the vertex belongs to a fan in S then whether it is a center or a leaf, (c) If it is a

leaf then which vertex is the center of the fan, and (d) If it is a center of the fan then it points to the ordered list in which the fan is maintained. Finally, S itself is maintained as a doubly linked list of the centers of its fans, such that insertions and deletions can be done in $O(1)$ time.

Lemma 4.2: MAKE-S has time complexity $O(|E|)$ and requires $O(|E|)$ space.

Proof:

Step 4.1 (the initialization of the data structure) requires $O(|V|)$ time. We already saw that Step 4.2 can be done in $O(|E|)$ time (see implementation of line 3 of PARALLEL-COLOR). Step 4.3 can be implemented in $O(d(w))$ time for each vertex w (see the implementation of RECOLOR), and thus the total time of Step 4.3 in MAKE-S is $O(|E|)$. The construction of a c-fan in Step 4.3 need not be completed but can be stopped the first time we reach a leaf x_i which belongs to another fan F' in S. Thus, Steps 4.4, 4.5 and 4.6 take $O(1)$ time per fan or $O(|V|)$ time in total. The shifts of Steps 4.9, 4.10 and 4.14 take $O(d(w))$ time per fan whose center is w. Since such a shift can be done around a certain vertex w only once (for after the shift w becomes a leaf of a u-fan) the total time for those shifts during the whole execution of MAKE-S is $O(|E|)$. Steps 4.12 and 4.15 may also require $O(d(w))$ time per fan, that is a total of $O(|E|)$ time. The other steps of MAKE-S (lines 4.7, 4.8, 4.11, 4.13 and 4.16) require $O(1)$ time per fan or $O(|V|)$ time in total.

The space bound follows from the discussion of the data-structures above.

Q.E.D.

We now turn to lines 5-6 of PARALLEL-COLOR: In line 6 (which we call "a substage" for $\alpha\beta$) we want to augment "in parallel" as many $\alpha\beta$-fans as possible. However, in order to reach the desired complexity for PARALLEL-COLOR lines 5-6 must be done in $O(|E|)$ time. Unfortunately, we have not found a way to augment in line 5 **all** $\alpha\beta$-fans "in parallel". The problem is that when we augment an

$\alpha\beta$-fan F, we usually flip the colors of some $\alpha\beta$-path which starts at F. However, the other end-point of this path might belong to another fan F', and thus when we later augment F', we shall traverse that same path again, thus violating our time constraints. Moreover, the flip of the $\alpha\beta$-path and the coloring of an uncolored edge, can cause 3 $\alpha\beta$-paths to be concatenated into one $\alpha\beta$-path which thus may be traversed many times (if the process of concatenation of that path would repeat itself). So we must give up the augmentation of **all** $\alpha\beta$-fans, and we settle instead for a fixed fraction ($\frac{1}{6}$). The approach is to repeatedly augment a fan F and remove from S all other fans that are invalidated by the change in coloring and the flip of an $\alpha\beta$-path that starts at F.

Now, before we describe the implementation of an $\alpha\beta$-substage (line 6), we have to prepare some data structures to enable an efficient implementation. What we need are d lists, such that for each color β ($\beta \neq \alpha$) we shall have a list which contains all $\alpha\beta$-fans in S. There is no problem to build such lists for all $\alpha\beta$-c-fans in S. However, since each u-fan may belong to more than one such list, more careful treatment is required in order not to violate the $O(|E|)$ complexity.

Procedure PRE-SUBSTAGE

begin

5.1 *for* each vertex $v \in V$ *do*

5.2 let ALPHA(v) be the edge of color α incident at v if such an edge exists;

5.3 Arrange all colors β such that $\beta \neq \alpha$ in some arbitrary order: $\beta_1, \beta_2 \ldots \beta_d$.

5.4 *for* each vertex w, which is the center of some u-fan in S *do*

open an empty list CLR(w); {CLR(w) will contain all colors $\neq \alpha$ incident at w arranged according to the order of step 5.3}

5.5 *for* $i=1$ *step* 1 *until* d *do*

begin

5.6 *for* each edge (u,v) of color β_i *do*

5.7 *if* u [resp. v] is the central vertex of a u-fan in S

5.8 *then* add β_i at the end of CLR(u) [resp. CLR(v)];

end

5.9 *for* each vertex w, which is the center of some u-fan in S *do*

begin

5.10 let $d'(w)$ be the number of uncolored edges incident at w.

5.11 open an empty list MISS(w); {MISS(w) will contain the first $d'(w)$ colors missing at w}

5.12 *for* $i=1$ *step* 1 *until* d *while* MISS(w) has less than $d'(w)$ colors *do*

begin

5.13 *if* β_i is the first color in CLR(w)

5.14 *then* CLR(w) $\leftarrow$ CLR(w) - $\{\beta_i\}$

5.15 *else* MISS(w) $\leftarrow$ MISS(w) $\cup$ $\{\beta_i\}$

{β_i is inserted at the end of MISS(w)}

end

end

5.16 *for* $i=1$ *step* 1 *until* d *do* open an empty list $S_{\alpha\beta_i}$;

{$S_{\alpha\beta_i}$ will contain the $\alpha\beta_i$-fans of S}

5.17 *for* each fan F in S *do*

begin

5.18 *if* F is an $\alpha\beta$-c-fan

5.19 *then* $S_{\alpha\beta} \leftarrow S_{\alpha\beta} \cup \{F\}$

else begin

5.20 let w be the center of F;

5.21 let β be the first color of MISS(w);

5.22 $S_{\alpha\beta} \leftarrow S_{\alpha\beta} \cup \{F\}$;

5.23 MISS(w) $\leftarrow$ MISS(w) - $\{\beta\}$;

end

end

end PRE-SUBSTAGE;

Lemma 4.3: PRE-SUBSTAGE requires $O(|E|)$ space and has time complexity $O(|E|)$.

Proof:

For each vertex w such that w is a central vertex of some u-fan in S, $|CLR(w)| + |MISS(w)| = d(w)$, and thus the total space that all lists $CLR(.)$ and $MISS(.)$ require is $O(|E|)$. The total space that the lists $S_{\alpha\beta}$ require is $O(|V|)$. The vector $ALPHA[.]$ also needs $O(|V|)$ space. Thus, PRE-SUBSTAGE requires a total of $O(|E|)$ space.

Steps 5.1 and 5.2 can be accomplished in $O(|V|)$ time using the list of edges colored α. Step 5.3 requires $O(d)$ time and Step 5.4 requires $O(|V|)$ time. The loop of Steps 5.5 - 5.8 can be done in $O(|E|)$ time (using the color lists mentioned in the previous section). For each w the execution of the loop in Steps 5.10 - 5.15 requires $O(d(w))$ time, and thus the whole execution of Steps 5.9 - 5.15 requires $O(|E|)$ time. Step 5.16 takes $O(d)$ time and finally the loop of Steps 5.17 - 5.23 requires $O(|V|)$ time. Thus, the total time complexity of PRE-SUBSTAGE is $O(|E|)$.

Q.E.D.

In view of PRE-SUBSTAGE we now reformulate PARALLEL-COLOR as follows:

Procedure PARALLEL-COLOR;

{this algorithm finds a $(d+1)$-coloring on a graph}.

begin

1. *while* there are uncolored edges *do*

2. *for* each color α *do*

begin {lines 3-6 below are α's stage}

3. color by α a maximal number of edges missing α at both ends;

4. MAKE-S;

5.a PRE-SUBSTAGE;

5.b *for* $i = 1$ *step* 1 *until* d *do*

6. SUBSTAGE($\alpha\beta_i$);

{SUBSTAGE($\alpha\beta_i$) augments as many $\alpha\beta_i$ fans of S as possible and updates the necessary data structures}

end

end PARALLEL-COLOR;

Before we give a detailed description of SUBSTAGE we define (for each given pair of colors α,β_i) the following subgraphs: Let H_F be the subgraph which consists of all edges and vertices of fans in $S_{\alpha\beta_i}$. Let $H_{\alpha\beta_i}$ be the subgraph which consists of all edges (and their endpoints) that lie on $\alpha\beta_i$-paths which start at vertices of H_F (however, we exclude from $H_{\alpha\beta_i}$ those edges which belong to H_F itself). $H_{\alpha\beta_i}$ is in fact a union of disjoint $\alpha\beta_i$-paths. We can further partition $H_{\alpha\beta_i}$ into two subgraphs: Let H_α be the set of all $\alpha\beta_i$-paths in $H_{\alpha\beta_i}$ that consists of one single edge of color α, and let $H_{\beta_i} = H_{\alpha\beta_i} - H_\alpha$ (i.e., H_{β_i} is the set of all $\alpha\beta_i$-paths in $H_{\alpha\beta_i}$ that contain at least one edge of color β_i).

As we shall see, only the subgraph of G which consists of H_F, H_α and H_{β_i} is needed for the execution of SUBSTAGE. However there is no need to explicitly

construct all those three subgraphs but only the explicit construction of H_{β_i} is required.

Procedure SUBSTAGE($\alpha\beta_i$)

begin

6.0 construct the subgraph H_{β_i};

6.1 *for* each fan $F \in S_{\alpha\beta_i}$ *do*

begin

6.2 $S_{\alpha\beta_i} \leftarrow S_{\alpha\beta_i} - F$;

6.3 *if* F is a u-fan and has more than 2 edges

then begin

6.4 let w be the center of F;

6.5 let β_j be the first color of MISS(w);

6.6 MISS(w) $\leftarrow$ MISS(w) - $\{\beta_j\}$;

6.7 $S_{\alpha\beta_j} \leftarrow S_{\alpha\beta_j} \cup \{F\}$

end;

6.8 *if* F is a c-fan

6.9 *then* RECOLOR(F); {ALPHA[v] must be updated}

6.10 *else* U-AUGMENT(F); {ALPHA[v] must be updated}

6.11 let $e = (w,x)$ be the edge of F that was colored by RECOLOR (after the shift was done) or by U-AUGMENT; {except for case (1) of RECOLOR, e has now color α}

6.12 let P_1 and P_2 be the $\alpha\beta_i$-paths that start at w and x (where if one of those paths ends at a leaf of an $\alpha\beta_i$-u-fan then denote that path by P_1).

6.13 REMOVE(P_1,F,$\alpha\beta_i$);

6.14 REMOVE(P_2,F,$\alpha\beta_i$);

{REMOVE($P,F,\alpha\beta_i$) removes or amends the fan that lies on the other end of the $\alpha\beta_i$-path P}

end

end SUBSTAGE;

REMOVE($P,F,\alpha\beta_i$) is the following procedure:

Procedure REMOVE($P,F,\alpha\beta_i$)

begin

R.1 *if* both end-points of P belong to F

R.2 *then* return.

else begin

R.3 let v be the end-point of P which does not belong to F;

R.4 *if* v does not belong to any fan in $S_{\alpha\beta_k}$ (for some k)

{$k \geq i$, since $S_{\alpha\beta_j}$ for $j < i$ are already empty}

R.5 *then* return

else begin

R.6 let F' be the fan in $S_{\alpha\beta_k}$ to which v belongs;

R.7 *if* F' is a u-fan having exactly two edges or

F' is a c-fan

R.8 *then* $S_{\alpha\beta_k} \leftarrow S_{\alpha\beta_k} - \{F'\}$ {see comment below}

R.9 *else begin*

{F' is a u-fan which has more than 2 edges}

R.10 *if* v is a leaf of F'

R.11 *then* remove from F' the edge wv

{where w is the center of F'}

R.12 *else begin*

{v is the center of F' and in this case $k = i$}

R.13 $S_{\alpha\beta_i} \leftarrow S_{\alpha\beta_i} - \{F'\}$;

R.14 let β_j be the first color of

MISS(v);

R.15 MISS(v) $\leftarrow$ MISS(v) - $\{\beta_j\}$;

R.16 $S_{\alpha\beta_j} \leftarrow S_{\alpha\beta_j} \cup \{F'\}$;

R.17 let P' be the $\alpha\beta_i$-path that

starts at the first leaf x_1 of F';

{by the notation of 6.12 v

itself is not on P'}

R.18 *if* the edge of P which is

incident at v has color β_i

R.19 *then* color vx_1 by α and

update ALPHA[.]

else begin

R.20 flip P' and

update ALPHA[.];

R.21 color vx_1 by β_i

end

R.22 REMOVE(P',F',$\alpha\beta_i$)

end

end

end

end REMOVE;

{COMMENT: Actually in line R.8 of REMOVE(P,F,$\alpha\beta_i$) it is not always necessary to remove F' from $S_{\alpha\beta_k}$. In fact, if F' is a c-fan it is only necessary to remove F' from $S_{\alpha\beta_k}$ in the following two cases: (i) when v is the center of F' and either the

last edge of P has color α or $k = i$; (ii) when v is a leaf of F', $i = k$ and either the last edge of P has color β_i or v is x_t or x_s. Also, when F' is a u-fan and v is the center of F' it is possible to apply Steps R.12-R.22 of REMOVE rather than execute R.8}.

Lemma 4.4: The total time SUBSTAGE requires for the execution of one stage (that is, the execution of lines 3-6 of PARALLEL-COLOR) is $O(|E|)$. The space required is also $O(|E|)$.

Proof:

The only additional data-structure which is required for the execution of SUBSTAGE (and REMOVE) is the data-structure which is needed for the construction of H_{β_i}. Clearly, this data-structure requires at most $O(|E|)$ space.

In order to analyze the time complexity of SUBSTAGE($\alpha\beta_i$) we distinguish between operations done on edges of the fans (edges of H_F) and operations done on $\alpha\beta_i$-paths.

Consider, first, the operations done on the edges of fans (edges of H_F) and on the fans themselves: First we note that whenever an $\alpha\beta_i$-c-fan is referred to in the procedures SUBSTAGE and REMOVE, it is immediately removed from $S_{\alpha\beta_i}$ (lines 6.2 and R.8). Thus, the operation of shifting a c-fan (which may be done in line 6.9) is executed at most once on each edge. The other operations on edges of c-fans (for example, coloring an uncolored edge) are also done at most once per edge and thus the total complexity of these operations during one stage of PARALLEL-COLOR is $O(|E|)$.

U-fans may be referred to more than once during the execution of SUBSTAGE($\alpha\beta_i$) in one stage of PARALLEL-COLOR (line R.11). Moreover, a u-fan which is removed from $S_{\alpha\beta_i}$ may be inserted into $S_{\alpha\beta_j}$ (lines 6.7 and R.16). However, each time a u-fan is approached in SUBSTAGE and in REMOVE it loses at

least one of its uncolored edges (lines 6.10, R.8, R.11, R.19, R.21). Thus, during one stage of PARALLEL-COLOR, the total time of operations done on u-fans (except for operations done on $\alpha\beta_i$-paths) is $O(|E|)$.

In order to compute the time required for operations on $\alpha\beta_i$-paths we first observe that any $\alpha\beta_i$-path that may be used in line 6.9 (execution of RECOLOR) or in line 6.10 (execution of U-AUGMENT) is later computed explicitly in line 6.12. Thus, at most two $\alpha\beta_i$-paths are constructed during SUBSTAGE for each fan of $S_{\alpha\beta_i}$ and another $\alpha\beta_i$-path P' may also be constructed in line R.17 (but we attribute the construction of P' to fan F' rather than to F). It follows from the discussion in the previous paragraphs that during the whole execution of one stage of PARALLEL-COLOR at most $O(|E|)$ $\alpha\beta_i$-paths are computed.

Now, each of those $\alpha\beta_i$-paths should be constructed (lines 6.9, 6.10, 6.12, R.17), should be traversed (lines 6.9, 6.10, R.1, R.3, R.18) and may be flipped (lines 6.9, 6.10, R.20). Once H_{β_i} is constructed, and with the help of the vector ALPHA[.], each of those operations requires time proportional to the length of the path. An $\alpha\beta_i$-path may also cause a call to REMOVE, where (except for traversing the path in steps R.1, R.3 and R.18) only fixed number of additional operations are done (recall that lines R.17 and R.20 are attributed to P' rather than to P). Thus the total number of operations done on the $\alpha\beta_i$-paths is proportional to their total length.

To proceed we shall now distinguish between $\alpha\beta_i$-paths that belong to H_α and those that belong to H_{β_i}: The $\alpha\beta_i$-paths that belong to H_α are of length 1 and since there are at most $O(|E|)$ of them, their total length, and therefore the total complexity of operations done on them during one stage of PARALLEL-COLOR is $O(|E|)$.

To compute the time required for operations on $\alpha\beta_i$-paths that belong to H_{β_i}, let us denote by c_{β_i} the number of edges having color β_i when MAKE-S ends.

Clearly, this number does not change until the beginning of SUBSTAGE($\alpha\beta_i$) (although the edges themselves may be changed whenever a shift of a c-fan is done). Now, using the list of edges of color β_i, and the vector ALPHA[.], the subgraph H_{β_i} can be constructed in step 6.0 in $O(c_{\beta_i})$ time and it does not change during the whole execution of SUBSTAGE($\alpha\beta_i$). Moreover, once a path of H_{β_i} is traversed, the fan at its first end-point is removed (in line 6.2 for P_1 and P_2 of line 6.12, and in line R.13 for P' of line R.17) while if there exists a fan on its other end-point it is either removed (in line 6.2 for the case of line R.1, in line R.8 for the case of line R.7, and in line R.13 for the case of line R.12) or the edge at its other end-point which belongs to the fan is removed from the fan (in line R.11 for the case of line R.10). Therefore, no path of H_{β_i} (or any part of it) is processed more than once during SUBSTAGE($\alpha\beta_i$), and the total number of operations done on $\alpha\beta_i$-paths of H_{β_i} is proportional to the total length of H_{β_i}, namely $O(c_{\beta_i})$.

Summing up over all colors β_i $(1 \leq i \leq d)$ we obtain

$$\sum_{i=1}^{d} O(c_{\beta_i}) = O(\sum_{i=1}^{d} c_{\beta_i}) = O(|E|)$$

which concludes the proof of lemma 4.4

Q.E.D.

Theorem 4.1: PARALLEL-COLOR colors all the edges of $G(V,E)$ in $O(|E|.d.log\,|V|)$ time and $O(|E|)$ space.

Proof:

The space bound follows immediately from Lemmas 4.2, 4.3 and 4.4.

The execution of PARALLEL-COLOR (lines 3-6) takes $O(|E|)$ time (Lemmas 4.2, 4.3, 4.4). If we could color in one stage α all uncolored edges that are incident at vertices which miss color α (these are exactly all the uncolored

edges that belong to those fans which are constructed by MAKE-S) then we would color all edges of $G(V,E)$ during one execution of lines 2-6 of PARALLEL-COLOR, that is in time $O(|E|.d)$. However, not all the uncolored edges of the fans in S are colored in steps 2-6. In fact, each time a fan F is treated in SUBSTAGE($\alpha\beta_i$) (in line 6.1 or R.12), one uncolored edge is colored (in lines 6.9, 6.10, and in lines R.19, or R.21) but up to 5 uncolored edges may be removed from the fans of S without getting colored: One edge in line 6.2 in the case when F is a u-fan having only two edges, and two edges for each of the $\alpha\beta_i$-paths P_1 and P_2 in line R.8 when F' is a u-fan of exactly two edges. Thus, in the worst case only $\frac{1}{6}$ of the uncolored edges of fans in S are colored in one execution of lines 2-6 of PARALLEL-COLOR, and we must repeat the loop $log\,|E| = log\,|V|$ times.

To this we must add two remarks: First, if an $\alpha\beta_i$-u-fan has more than two edges then when it is removed from $S_{\alpha\beta_i}$ it is inserted into some $S_{\alpha\beta_j}$ (lines 6.7, R.16), and its uncolored edges still belong to the fans of S. Second, a shift of the edges of a c-fan does not cause any uncolored edge to be removed from the fans of S: Let x_i be a leaf of a c-fan and assume that there are k uncolored edges incident to x_i. If $\gamma_0 = m(x_i)$ then there are other k colors $\gamma_1,\gamma_2,\ldots,\gamma_k$ missing at x_i. When the c-fan to which x_i belongs is shifted, only $m(x_i)$ changes but not $\gamma_1.....\gamma_k$. Thus, the k uncolored edges of x_i will get colors in the stages that correspond to $\gamma_1.....\gamma_k$ and the possibility that color γ_0 would be changed to some other color $\overline{\gamma_0}$ before the γ_0-stage is executed but after the $\overline{\gamma_0}$-stage has already been executed is therefore irrelevant to the uncolored edges incident at x_i because they would be treated at the γ_j-stages ($1 \le j \le k$).

Thus, lines 2-6 of PARALLEL-COLOR are repeated at most log|V| times and the total complexity of PARALLEL-COLOR is $O(|E|.d.log\,|V|)$.

Q.E.D.

5. Algorithm EULER-COLOR

In this section we present another efficient algorithm for coloring general graphs with $d+1$ colors. This algorithm uses a divide-and-conquer method that combines COLOR and PARALLEL-COLOR and has time complexity of $O(|E|\sqrt{|V|\log|V|})$ (a similar algorithm is presented in [A]; see comment at the end of the paper).

In [GK1] the following algorithm for edge coloring by d colors a bipartite graph whose maximum degree is d was presented:

Procedure EULER-COLOR(G);

begin

1. let d be the maximum degree in G;
2. *if* $d = 1$
3. *then* color all edges of G with a new color

   else begin

   4. use EULER-PARTITION to divide G into two edge-disjoint subgraphs G_1 and G_2 with maximum degrees d_1 and d_2 such that $\lfloor \frac{d}{2} \rfloor \le d_1, d_2 \le \lceil \frac{d}{2} \rceil$;
   5. EULER-COLOR(G_1);
   6. EULER-COLOR(G_2);
   7. *if* G is $(d+1)$-colored

      then begin

      8. let γ be the color with fewest edges in G and let E_γ be the set of all edges colored by γ; $\{ \; |E_\gamma| \le \frac{|E|}{d} \; \}$
      9. *if* $d \ge \sqrt{\frac{|V|}{\log|V|}}$
      10. *then for* each edge $e \in E_\gamma$ *do* AUGMENT(e)

11. *else* TYPED-RECOLOR(G, E_γ)

end

end

end EULER-COLOR;

Where:

(1) EULER-PARTITION is a procedure that partitions the edges of *any* graph (bipartite or not) into (possibly non-simple) open and closed paths such that: (i) no closed path intersects another (closed or open) path. (ii) a vertex of odd (resp. even) degree in G is the end-point of exactly one (resp. zero) open path. EULER-PARTITION runs in $O(|E|)$ time and it can be used to divide G into two subgraphs G_1 and G_2 with maximum degrees d_1 and d_2 where $d_1 = \left\lfloor \frac{d+1}{2} \right\rfloor$ and for bipartite graphs $d_2 = \left\lceil \frac{d}{2} \right\rceil$ while for other graphs $d_2 = \left\lceil \frac{d+1}{2} \right\rceil$. This is done by placing alternate edges of each path in alternate subgraphs, always starting with G_2 (see [B], [G]).

(2) AUGMENT(e) is a procedure that colors an uncolored edge e in one of the possible d colors in $O(|V|)$ time.

(3) TYPED-RECOLOR(G, E_γ) is a procedure that colors a set E_γ of edges in d colors, where E_γ constitutes a matching of G.

EULER-COLOR(G) is based on the fact that for a bipartite graph G, in Step 5 (resp. 6), the (bipartite) subgraphs G_1 (resp. G_2) is recursively colored by d_1 (resp. d_2) colors, and thus at the end of Step 6, the whole graph G is colored by $d_1 + d_2$ colors, namely, by d or $d+1$ colors. In the latter case, the color γ with the fewest edges in G is uncolored, and we obtain a set (matching) E_γ of at most $\frac{|E|}{d}$ uncolored edges in G. If we denote $d_0 = \sqrt{\frac{|V|}{\log |V|}}$ then for all recursive calls such that $d \geq d_0$, AUGMENT is repeatedly used to color the edges of E_γ and

it is shown in [GK1] that the total time AUGMENT requires is $O(\frac{|E||V|}{d_0}) = O(|E|\sqrt{|V|log|V|})$. For all recursive calls such that $d < d_0$ TYPED-RECOLOR is used to color the edges of E_γ and the total time it consumes is $O(|E|.d_0.log|V|) = O(|E|\sqrt{|V|log|V|})$ (see Theorm 1 in [GK1]). Thus, the time EULER-COLOR needs is $O(|E|\sqrt{|V|log|V|})$.

When we turn to the non-bipartite case we can clearly replace AUGMENT by RECOLOR (both color an edge in time $O(|V|)$) and we can replace TYPED-RECOLOR by PARALLEL-COLOR (for both color the edges of E_γ in time $O(|E|.d.log|V|)$. The only problem is that when G is not a bipartite graph, the recursive call for G_1 (resp. G_2) returns d_1+1 (resp. d_2+1) coloring, and thus at the end of Step 6 the whole graph G might be colored by $(d_1+1) + (d_2+1) \leq d+3$ colors. Therefore, in Steps 8-11 edges of *two* colors (rather than one) should be recolored. This however constitutes no problem, since the total number of those edges is $\leq \frac{2|E|}{d}$, that is of the same order of magnitude as in the bipartite case. Also, PARALLEL-COLOR would color those edges in the same time (that is, $O(|E|.d.log|V|)$), and in fact, when $d \leq \sqrt{\frac{|V|}{log|V|}}$ no recursive calls are required at all but PARALLEL-COLOR can be used directly to color G_1 (resp. G_2) in d_1+1 (resp. d_2+1) colors.

Thus, EULER-COLOR for the non-bipartite case is as follows:

Procedure EULER-COLOR(G);

{this procedure colors by $d+1$ colors the edges of a (general) graph G whose maximum degree is d}

begin

1. let d be the maximum degree of G;

2. *if* $d \leq \sqrt{\frac{|V|}{log|V|}}$

3. *then* PARALLEL-COLOR(G)

else begin

4. use EULER-PARTITION to divide G into two edge-disjoint subgraphs G_1 of degree $d_1 \leq \lfloor \frac{d+1}{2} \rfloor$ and G_2 of degree $d_2 = \lceil \frac{d+1}{2} \rceil$.

5. EULER-COLOR(G_1);

6. EULER-COLOR(G_2);

7. *if* G has more than $d+1$ colors

then begin

8. let γ_1 be the color with fewest edges in G and let E_γ be the set of all edges colored by γ_1.

9. *if* G has more than $d+2$ colors

10. *then begin* {G has $d+3$ colors}

11. let γ_2 be the color with fewest edges in $G-E_\gamma$ and let E_{γ_2} be the set of all edges colored by γ_2;

12. $E_\gamma \leftarrow E_\gamma \cup E_{\gamma_2}$;

end

13. *for* each edge $e \in E_\gamma$ *do* RECOLOR(e)

end

end

end EULER-COLOR;

Theorm 5.1: EULER-COLOR(G) colors the edges of a general graph G by $d+1$ colors in $O(|E|\sqrt{|V| log |V|})$ time and uses $O(|E|+|V|)$ space.

Proof:

The detailed proof that EULER-COLOR(G) runs in $O(|E|\sqrt{|V|\log|V|})$ time follows the proof in [GK1].

6. Algorithm ALCOLOR

In this section we give algorithm ALCOLOR which edge-colors with d colors a large class of graphs (and with $d+1$ colors all other graphs).

ALCOLOR is based on the proof of Vizing's "Adjacency Lemma" ([FW], [V65a], [V65b]) and on the following definition: Denote by $d^*(w)$ the number of vertices adjacent with w and having degree d. Then an edge vw is defined to be *eliminatable* if w has at most $d - d(v)$ adjacent vertices of degree d other than v: i.e. edge vw is eliminatable when

$$\begin{aligned} d(v) + d^*(w) &\le d \quad (\text{if } d(v) < d) \\ d^*(w) &= 1 \qquad\quad (\text{if } d(v)=d) \end{aligned}$$

(notice that the definition is not symmetric with v and w)

In other words, the edges that are not permitted in a "critical" graph by the adjacency lemma are eliminatable.

ALCOLOR is outlined as follows. It repeatedly deletes eliminatable edges from G until they all disappear or the maximum degree of G decreases. An edge that was not eliminatable in the original graph G may become eliminatable when some edges are deleted. On the other hand, once an edge becomes eliminatable, it remains so thereafter. Let G' be the resulting graph. There are two cases: $d(G') = d(G)-1$; or $d(G') = d(G)$. In the lucky case, namely when $d(G') = d(G)-1$, we first color G' with $d(=d(G')+1)$ colors by algorithm COLOR. We then update the d-coloring of G' to a d-coloring of G' plus the last deleted edge, using a procedure called ALRECOLOR and we repeat this procedure for each of the deleted edges in *reverse* order to their deletion until a d-coloring of G is obtained. In the second case, when $d(G') = d(G)$, we simply color G' itself with $d+1$ colors by COLOR.

Procedure ALCOLOR (G):

{ALCOLOR finds a d- or $d+1$- coloring based on eliminatable edges}

begin

1. $d \leftarrow d(G)$;
2. $G' \leftarrow G$;
3. *while* G' has an eliminatable edge vw *and* $d(G')=d$ *do*

begin

4. $G' \leftarrow G'-vw$; {delete vw from G}
5. push vw on the top of a stack S;

end

6. *if* $d(G')= d - 1$

then begin {the lucky case}

7. COLOR (G'); {G' is now colored with d colors}
8. *while* stack S is not empty *do*

begin

9. pop up an edge, say vw, from S;
10. ALRECOLOR (G',vw);

{augments the d-coloring of G' to $G' + vw$}

11. $G' \leftarrow G'+ vw$

end

end

12. *else* {unlucky case: G' has no eliminatable edges and $d(G') = d$}

COLOR (G)

end ALCOLOR;

We now explain procedure ALRECOLOR. Suppose that edge vw is eliminatable in G and that $G-vw$ is colored with a set of d colors. Each vertex $u(\neq v,w)$ has at least one missing color $m(u)$ if $d(u)<d$ and has no missing color if

$d(u)=d$. In the d-coloring of $G-vw$, vertex v has $d-d(v)+1(\geq 1)$ missing colors; For each color γ of the $d-d(v)+1$ missing colors of v, there exists a maximal fan sequence $F=[wv = wx_0, wx_1, \ldots, wx_s]$ at w in which edge wx_1 is colored with γ if $s \geq 1$. One of the following must occur:

(a) $m(x_s) \in M(w)$;

(b) an edge of F is colored with $m(x_s)$;

(c) $d(x_s)=d$ (and hence x_s has no missing color).

Since vw is eliminatable, w has at most $d-d(v)$ adjacent vertices of degree d other than v. Therefore one of the following must occur:

Case (1): there exists a maximal fan sequence $F=[wx_0, wx_1, \ldots, wx_s]$ such that $m(x_s) \in M(w)$;

Case (2): there exists a maximal fan sequence $F=[wx_0, wx_1, \ldots, wx_s]$ such that an edge of F is colored with $m(x_s)$;

Case (3): there exist two fan sequences, not necessarily maximal, $F1=[wx_0, wx_1, \ldots, wx_s]$ and $F2=[wy_0, wy_1, \ldots, wy_t]$ which meet exactly at $v(=x_0=y_0)$ and at $x_s(=y_t)$. (If neither of the $d - d(v) + 1$ fans that start at wv belong to cases (1) or (2), then at least two of the fans must coincide from certain edge $wx_s = wy_t$ on; we may assume without loss of generality that $s \geq t$ and thus $s \geq 2$).

We are now ready to present ALRECOLOR.

Procedure ALRECOLOR (G', vw) :

{ALRECOLOR augments the d coloring of G' to $G' + vw$}

begin {vw is eliminatable in $G'+vw$}

1. *if* Case (1) or case (2) occurs
2. *then* extend a d-coloring of G' into a d-coloring of $G'+vw$ in a similar way to Case (1) or (2) of RECOLOR

3. *else begin* {Case (3) occurs}

4. let $\alpha = m(w)$ and $\beta = m(x_{s-1}) = m(y_{t-1})$;

5. let P be an $\alpha\beta$ path which starts at w;

6. *if* P does not end at x_{s-1}

then begin

7. shift $F1$ from x_{s-1};{now wx_{s-1} is uncolored}

8. flip P;{now $\beta \in m(w)$}

9. color wx_{s-1} with β;

end

else begin { P does not end at y_{t-1}}

10. shift $F2$ from y_{t-1};{now wy_{t-1} is uncolored}

11. flip P;{now $\beta \in m(w)$}

12. color wy_{t-1} with β;

end

end

end ALRECOLOR;

We have the following lemma on ALRECOLOR:

Lemma 6.1 Suppose that all the edges of $G=G'+vw$ except vw are colored with d colors and that vw is eliminatable in G. Then ALRECOLOR edge-colors G in $O(|V|)$ time, using $O(|E|)$ space.

Proof:

One can easily establish the correctness of ALRECOLOR in a similar way to the proof of Vizing's adjacency lemma [FW, pp.72-74]. The claims on time and space are also easily verified.

Now we have the following theorem.

Theorem 6.1: Algorithm ALCOLOR edge-colors an arbitrary graph G with d or $d+1$ colors in $O(|E||V|)$ time, using $O(|E|)$ space.

Proof:

Since one can easily prove the correctness of ALCOLOR, we shall establish the claims on time and space. As for the space, in addition to the data used by COLOR, ALCOLOR needs a stack S to store the deleted edges, which clearly uses $O(|E|)$ space and two arrays $d[.]$ and $d^*[.]$, each of length $|V|$, representing $d(u)$ or $d^*(u)$ for $u \in V$. We also need an $O(|E|)$ space for a (temporary) list of eliminatable edges which were found but not yet deleted. Thus ALCOLOR uses $O(|E|)$ space.

By lemma 6.1 one execution of ALRECOLOR can be done in $O(|V|)$ time, and since ALRECOLOR is executed at most $|E|$ times the total cost of ALRECOLOR is $O(|E||V|)$ time. Since one can delete an edge in $O(1)$ time using our data structure, the deletion of edges costs $O(|E|)$ time. Thus, what remains to be proved is that the total cost of finding eliminatable edges is $O(|E||V|)$ time. Since one can compute $d(u)$ and $d^*(u)$ for all $u \in V$ in $O(|E|)$ time, one can find all the edges eliminatable in the original graph G in $O(|E|)$ time. Hence, it suffices to show that the newly eliminatable edges can be found in $O(|E||V|)$ time in total.

Suppose that edge xy is deleted in graph G', then we must update both $d[.]$ and $d^*[.]$. We update $d[.]$ by decreasing both $d[x]$ and $d[y]$ by one. Some of the edges incident with x or y may become eliminatable. Therefore, we check for each of the edges incident with x or y whether it becomes eliminatable, which clearly can be done in $O(|V|)$ time. Since ALCOLOR deletes at most $|E|$ edges, this checking can be done in $O(|E||V|)$ time in total.

However, the procedure above does not find all the edges that become newly eliminatable. Suppose that $d(x) = d$ or $d(y) = d$, say $d(x) = d$. Then $d^*(z)$ decreases for each neighbor z of x, and so an edge incident with z may also become eliminatable. Therefore, we must check for every edge incident

with z whether it becomes eliminatable. This check can be done in $O(d(x)+\Sigma d(z))=O(|E|)$ time, where z runs in the summation over all the neighbors of x. Of course, the arrays $d[.]$ and $d^*[.]$ can be updated in this time. The case in which an end of a deleted edge has degree d (the maximum degree of the graph) occurs at most $|V|$ times, because the deletion of edges ends as soon as the maximum degree decreases. Hence the checking above can be done in $O(|E||V|)$ time in total, This completes the proof showing that the newly eliminatable edges can be found in $O(|E||V|)$ time in total.

Q.E.D.

7. Special Graphs which are d-Colorable

7.1. Planar Graphs

Vizing [V65b] has proved that $q^*(G) = d$ if G is a planar graph with $d \geq 8$. There exists a planar graph G with $q^*(G) = d+1$ for each d, $3 \leq d \leq 5$. It is conjectured that $q^*(G) = d$ for a planar graph G with $d = 6$ or 7 [FW].

We shall show that ALCOLOR colors a planar graph of degree $d \geq 8$ by d colors. A direct implementation of Vizing's proofs yields algorithms for the case $d \geq 10$. But for the cases $d = 8$ or 9 ALCOLOR is the first algorithm to be published.

We first have a lemma.

Lemma 7.1: Any planar graph whose maximum degree is $d \geq 8$ has an eliminatable edge.

Proof:

Vizing showed that for $d=8$ and for $d \geq 10$ this condition is required by the planarity of the graph (see [FW, pp. 106-108] or [Y, p. 295]). We show in the Appendix that the condition must hold whenever $d \geq 8$.

The following is an immediate consequence:

Theorem 7.1: Algorithm ALCOLOR edge-colors a planar graph G with d colors if $d \geq 8$.

7.2. Series-Parallel Graphs

A *simple* graph G is said to be *series-parallel* if G contains no K_4 as a subcontraction, that is, K_4 cannot be obtained from G by repeating the deletion or contraction of edges (see [D] or [TNS] for constructive definitions of a series-parallel graph). The class of series-parallel graphs is a subclass of planar graphs, but large enough to include the class of outerplanar graphs. We have the following lemma.

Lemma 7.2: Any series-parallel graph G whose maximum degree $d \geq 4$ has an eliminatable edge.

Proof:

Since G is series-parallel, G has a vertex of degree at most two [O]. Let S be the set of such vertices. Clearly $G' = G - S$ is also series-parallel. Therefore G' has a vertex w of degree at most two. Since the degree of w was at least three in G, a vertex $v \in S$ was adjacent with w in G. But in G $d^*(w) \leq 2$, so $d(v) + d^*(w) \leq 4$. Hence edge vw is eliminatable.

Q.E.D.

The following is an immediate consequence:

Theorem 7.2: Algorithm ALCOLOR edge-colors a series-parallel graph G with d colors if $d \geq 4$.

A series-parallel graph G does not always contain an eliminatable edge if $d = 3$. Therefore the direct application of ALCOLOR does not always produce a $q^*(G)$-coloring for the case $d = 3$. However we have the following lemma:

Lemma 7.3: If a series-parallel graph G with $d = 3$ has no eliminatable edge, then G has a triangle uvw such that $d(v)=2$ and $d(u)=d(w)=3$.

Proof:

G may not have a vertex of degree one (for the edge adjacent to such a vertex is eliminatable). Thus by [0] G must have a vertex of degree two. Let v be any vertex of degree two, and let u and w be the neighbors of v. Since none of the edges vu,vw,uv, and uv is eliminatable, we have $d(v)+d^*(u) \geq 4$, $d(v)+d^*(w) \geq 4$, or $d^*(u) \geq 2$, $d^*(w) \geq 2$. Hence $d(u)=d(w)=3$ and $d^*(u)=d^*(w)=2$. We shall show that u,v and w constitute a triangle, that is, u is adjacent with w. Suppose, contrary to the claim, that u is not adjacent with w for every vertex v of degree two. Then contract one of the two edges incident with v for every vertex v of degree two. The resulting graph G' has neither multiple edges nor vertices of degree two, so is a simple graph with maximum degree three. Since G is series-parallel, G' is also series-parallel and hence G' must contain a vertex of degree at most two, a contradiction.

Q.E.D.

Let G' be the graph obtained from G by contracting the triangle specified by Lemma 7.2 into a single vertex. G' is also series-parallel and has two vertices fewer than G. Clearly any 3-coloring of G' can be extended into a 3-coloring of G. Thus we have shown that $q^*(G)=3$ for any series- parallel graph G with $d=3$. Obviously $q^*(G)=d$ if $d \leq 2$ and G is not an odd cycle. These facts together with Theorem 7.2 imply the following theorem:

Theorem 7.3: If a series-parallel graph G is not a simple odd cycle, then $q^*(G)=d$.

Using Lemma 7.3, one can easily modify algorithm ALCOLOR so that it would color any series-parallel graph G with $q^*(G)$ colors. Theorem 7.3 is a generalization of Fiorini's result that every outerplanar graph except odd cycles has an

edge-coloring with d colors [F].

7.3. Random Graphs

Start with $|V|$ distinguished (labeled) vertices, and choose every edge with a fixed probability p, $0<p<1$, independently of the choices of the other edges. The resulting graph is called a *random* graph. Almost every random graph has exactly one vertex w of maximum degree (see [Bo, Theorem 9, pp. 135-136]). Let v be any neighbor of w, then $d(v)+d^*(w) \leq d$. Thus vw is eliminatable, and moreover $G-vw$ has maximum degree one less than d. Hence, algorithm ALCOLOR colors almost every random graph G with d colors.

8. Some NP-Completeness Results (on Matching and Regular Graphs)

As stated in the introduction, the problem of finding a minimum edge-coloring in a general graph is NP-complete [H]. Moreover, even the problem of finding a minimum edge-coloring in a general (regular) graph of degree k is NP-complete for any given k [LG]. We conclude this paper by indicating two (additional) sets of NP-completeness results which are implied by the NP-complete nature of the general Edge-Coloring problem. No detailed proofs are given but only their outlines are sketched.

(1) The first set of results deals with regular graphs:

Claim 8.1: The general edge-coloring problem (on a non-regular graph) reduces to the edge coloring problem on a regular graph of the same degree.

Outlines of proof:

Given a graph G of maximum degree d. For each vertex v in G add to G a subgraph $G_v = K_{d,d-1}$. Denote the d vertices of degree $d-1$ in G_v by $v_1,v_2,\ldots,v_d$, then add the following edges: For each edge vu in the original graph G, add an edge which connects some vertex v_i (which still has degree $d-1$) to some vertex u_j of G_u. Let $d(v)$ be the degree of v. Then connect each of the $d - d(v)$

vertices of G_v which still have degree $d-1$ to the vertex v itself.

The resulting graph is a regular graph that has a d-coloring of its edges if and only if the original graph G has such a coloring.

Corollary 8.1: The general edge-coloring problem reduces to the problem of finding whether a regular graph G of degree d with even number of vertices and no cut-vertex has a d-coloring of its edges (follows from [FW, Corollary 6.3 and Exercise 6b]).

Corollary 8.2: The problem of finding whether a general graph G has k disjoint perfect matchings for some integer k is NP-complete (follows from Claim 8.1 when G is regular and $k = d$).

Comment: The problem of Corollary 8.2 reduces to the case where G has maximum degree $k+2$ (for $k=1$ the result was first observed by Adi Shamir : Each vertex v in the original graph is replaced by a path of length $2(d(v)-1)$ and the edges incident to v are made incident to alternate vertices of that path).

(2) The second set of results deals with finding a restricted edge-coloring on a (bipartite) graph.

Theorem 8.1: The following problem is NP-complete: Given a (bipartite) graph G with an even number of vertices and an integer s, find whether there exist in G two disjoint matchings M_0 which is perfect ($|M_0| = \frac{|V|}{2}$) and M_1 of cardinality s.

Comment: The Theorem remains correct even when the degree of G is 3.

Outlines of proof:

In [EIS] the following problem, denoted by RTT (Restricted Time-Table) was proved to be NP-complete: Given a bipartite graph G with vertex sets T and C, where the maximum degree of the graph is 3 and each vertex $t \in T$ has degree 3 or 2, and where in the latter case t has a "forbidden-color" i, $1 \leq i \leq 3$. Find

whether there exists an edge-coloring of G with colors i, $1 \leq i \leq 3$ such that i misses every vertex with a "forbidden-color" i.

Let RRTT (Regular Restricted Time-Table) be the problem RTT where each color i is forbidden the same number of times and each vertex $c \in C$ has degree 3. RTT reduces to RRTT as follows: Let G be an instance of RRT. Make 3 copies of G. Cyclically permute the "forbidden-colors" on each copy, so that each color is forbidden the same number of times. For a vertex $c \in C$ with degree 2 add a new vertex joined to all 3 copies of c. Also, identify all 3 copies of a vertex $c \in C$ with degree 1.

Now RRTT reduces to the problem of Theorem 8.1 as follows: Let $t \in T$ have "forbidden-color" i. If $i = 2$ add two edges tc_1 and c_1t_2. If $i = 3$ add an edge tc_3 (where c_1, t_2, c_3 are new vertices). Finally, take $s = |T|$.

The new graph is bipartite and both vertex sets have the same number of vertices. A perfect matching M_0 includes all edges tc_3 and c_1t_2 and so can be used for color no. 3. Similarly, M_1 can be used for color no. 2.

Corollary 8.3: The following problem is NP-complete: Given a (bipartite) graph G, find two disjoint matchings M_0 and M_1 such that the pair $(|M_0|, |M_1|)$ is lexicographically maximum (this is the Lexicographical Matching Problem).

Corollary 8.4: The following problem is NP-complete: Given a (bipartite) graph G and integers c_i, $i = 1,2, \ldots, d$. Is there a d-coloring of the edges of G with exactly c_i edges colored i? (Theorem 8.1 with G of degree 3 is a special case of the problem of Corollary 8.4).

Comment: Comparison with the paper of Arjomandi [A].

Before submitting the revised version of our paper for publication, we have learned about the paper of Eshrat Arjomandi which was independently published in INFORMATION [A]. Comparison of the two papers shows that they are very much in similarity. In fact, the algorithm EULER-COLOR presented in [A] is as much the same as the algorithm EULER-COLOR presented in Section 5 of this article (both algorithms are based on the procedure EULER-COLOR which was presented in [GK1]); also algorithm RECOLOR-ONE of Arjomandi is in fact the same as algorithm COLOR which appears in Section 3 of our paper - both algorithms consist of running over the uncolored edges procedures which are direct implementations of Vizing's proof [V64], that is, PAINT and AUGMENT of Arjomandi and RECOLOR (cases 1 and 2 respectively) in our work.

Moreover, both EULER-COLOR of Arjomandi and our EULER-COLOR contain as subprocedure an algorithm which colors in parallel as many edges as possible, using $\alpha\beta$-paths: These are RECOLOR-TWO in [A] and PARALLEL-COLOR in Section 4 of this work. However, at this point the similarity breaks. Our PARALLEL-COLOR algorithm employs a special routine, MAKE-S, which takes care that the basic elements of Vizing's proof (i.e. the colored "fans") on which our procedure SUBSTAGE works will be vertex-disjoint, and for that purpose special data structure (the u-fans of Section 4) are defined. Contrary to this, the procedure RECOLOR-TWO of Arjomandi lacks a similar mechanism and thus the fans created by PAINT(e) in Step 5.2 may share common leaves. In particular, it may happen that the colors of the last fan created by PAINT are shifted (if that fan has parameter $t = 0$) and thus destroy all fans that share with it a common leaf. It seems to us that such a case would prevent COLOR-ALL (the analog in [A] of our SUBSTAGE) from completing its job in the expected time, or that it would force the loop of Step 3 of RECOLOR-TWO to be executed more than $O(\log |V|)$ times.

Another (minor) difference between the papers of Arjomandi and ours is that in [A] a vertex-color incidence matrix is maintained, resulting in an additional term of $O(|V|d)$ both in the time complexity of the algorithm and in the space it needs, whereas in our algorithm we use instead an array W (see Section 3) and thus avoid that term.

Acknowledgement.

We thank Nobuji Saito and Norishige Chiba for their stimulating suggestions.

References

[A] E. Arjomandi, "An Efficient Algorithm for Coloring the Edges of a Graph with $d+1$ Colors", INFORMATION, 20, 2 (1982), pp. 82-101.

[AHU] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley, Reading, Mass., 1974.

[B] C.Berge, "Graphs and Hypergraphs", North-Holland, Amsterdam, 1973.

[Bo] B. Bollobás, "Graph Theory", An Introductory Course, Springer-Verlag, Berlin, 1979.

[CH] R.Cole and J.Hopcroft, "On Edge Coloring Bipartite Graphs", SIAM J. on Comp., 11 (Aug. 1982), pp. 540-546.

[D] R.J. Duffin, "Topology of Series-Parallel Networks", J. Math. Applic., 10 (1965), pp. 303-318.

[EIS] S.Even, A.Itai, and A.Shamir, "On the Complexity of Timetable and Multicomodity Flow", Siam J. on Computing, 5 (Dec. 1976), pp. 691-703.

[F] S. Fiorini, "On the Chromatic Index of Outerplanar Graphs", J. Combinatorial Theory (Ser. B), 18 (1975), pp. 35-38.

[FW] S. Fiorini and R.J. Wilson, "Edge-Coloring of Graphs", Pitman, London, 1977.

[G] H.Gabow, "Using Euler Partitions to Edge-Color Bipartite Multigraphs", Internl. J. of Computer and Information Sciences, 5 (Dec. 1976), pp. 345-355.

[GJ] M.R.Garey and D.S.Johnson, "Computers and Intractability: A Guide to the Theory of NP-completeness", W.H.Freeman and Co., San-Francisco, Calif., 1978.

[GK1] H.Gabow and O.Kariv, "Algorithms for Edge-Coloring Bipartite Graphs", Proc. 10th Annual ACM Symp. on Theory of Computation (STOC), San-

Diego, Calif., 1978, pp. 184-192.

[GK2] H.Gabow and O.Kariv, "Algorithms for Edge-Coloring Bipartite Graphs and Multigraphs", SIAM J. on Comp., 11 (Feb. 1982), pp. 117-129.

[GK3] H.Gabow and O.Kariv, "On the Edge-Coloring Problem for General Graphs", Unpublished Extended Abstract, 1978.

[Go] T.Gonzalez, "A Note on Open Shop Preemptive Schedules", IEEE Trans. Comp., C-28 (1979), pp. 782-786.

[GS] T.Gonzalez and S.Sahni, "Open Shop Scheduling to Minimize Finish Time", J. ACM, 23 (Oct. 1976), pp. 665-679.

[Gt] C.C.Gotlieb, "The Construction of Class-Teacher Timetable", Proc. IFIP Congress 62, Munich, North-Holland, Amsterdam, 1963, pp. 73-77.

[H] I.J. Holyer, "The NP-Completeness of Coloring", SIAM J. on Computing, 10 (1981), pp. 718-720.

[HK] S.L.Hakimi and O.Kariv, "On a Generalization of Edge-Coloring in Graphs", Technical Report, EECS Dept., Northwestern Univ., Ill., Sept. 1983, to be published in the J. of Graph Theory.

[LG] D.Leven and Z.Galil, "NP Completeness of Finding the Chromatic Index of Regular Graphs", J. of Algorithms, 4 (1983), pp. 35-44.

[LL] E.L.Lawler and J.Lebetoulle, "On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming", J. ACM, 25 (1978), pp. 612-619.

[LVP] G.Lev, N.Pippenger and G.Valiant, "A Fast Algorithm for Routing in Permutation Networks", IEEE Trans. Comput., C-30 (1981), pp. 93-110.

[NS] T.Nishizeki and M.Sato, "An Approximation Algorithm for Edge-Coloring Multigraphs", Technical Report TRECIS-83003, Tohoku University, Japan, July 1983.

[O] O. Ore, "Theory of Graphs", Amer. Math. Soc., Colloq. Publ., 38, Providence, R.I., 1962.

[TNS] K. Takamizawa, T. Nishizeki, N. Saito, "Linear-Time Computability of Combinatorial Problems on Series-Parallel Graphs", J. ACM, 29, 3 (July 1982), pp. 623-641.

[V64] V.G. Vizing, "On an Estimate of the Chromatic Class of a p-Graph" (in Russian), Diskret. Analiz., 3 (1964), pp. 23-30.

[V65a] V.G. Vizing, "The Chromatic Class of a Multigraph", Cybernetics, 3 (1965), pp. 32-41 [Kibernetika 1 (1965) pp. 29-39].

[V65b] V.G. Vizing, "Critical Graphs with a Given Chromatic Class" (in Russian), Diskret. Analiz., 5 (1965), pp. 9-17.

[Y] H.P. Yap, "On Graphs Critical with Respect to Edge-Colorings", Discrete Math., 37 (1981), pp. 289-296.

Appendix: Proof of Lemma 7.1

Lemma 7.1: Any planar graph whose maximum degree is $d \geq 8$ has an eliminatable edge.

Proof:

Suppose that a planar graph G with $d \geq 8$ has no eliminatable edges. Let n_i be the number of vertices of degree i in G. Clearly $n_1=0$. Since G is planar, we have from Euler's equation.

$$12 + n_7 + 2n_8 + \cdots + (d-6)n_d \leq 4n_2 + 3n_3 + 2n_4 + n_5. \qquad (1)$$

Let $n_d(i_2,i_3, \ldots, i_7)$ be the number of vertices of degree d which have i_2 neighbors of degree 2, i_3 neighbors of degree 3, $\ldots$, i_7 of degree 7. Since each edge vw of G is not eliminatable,

$$\begin{array}{ll} d(v) + d^*(w) \geq d+1 & \text{if } d(v) < d; \\ d^*(w) \geq 2 & \text{if } d(v) = d \end{array}$$

Thus, $d^*(w) \geq 2$ for every $w \in V$. Let j be $2 \leq j \leq 7 \leq d-1$. Counting the number of edges with one end of degree j and the other end of degree d, we have

$$2n_j \leq \Sigma\, i_j\, n_d(i_2,i_3, \ldots, i_7), \qquad (2)$$

where the summation is over all possible $i_2,i_3, \ldots, i_7$.

Equation (2) can be further refined if $j = 3$ or 4 in particular. First let v be any vertex of degree 3, and consider, in detail, the degrees of neighbors of v. For any neighbor w of v, we have $d^*(w) \geq d-2$ and hence $d(w)= d$ or $d-1$. Since $d^*(v) \geq 2$, the following must occur:

(a) One of the neighbors of v has degree $d-1$ and the other two have degree d; or

(b) The three neighbors have degree d.

Let r be the number of vertices of degree 3 satisfying (a). Then equation (2) is

refined for $j=3$ as follows:

$$2r + 3(n_3 - r) \leq \Sigma i_3\, n_d(i_2, i_3, \ldots, i_7). \tag{3}$$

Next let v be any vertex of degree 4. Let w_1, w_2, w_3, and w_4 be the neighbors of v. Since vw_i is not eliminatable, $d^*(w_i) \geq d-3$ and hence $d(w_i) \geq d-2$ for $i=1,2,3,4$. If $d(w_i) = d-2$ for some i, then $d^*(v) \geq 3$, and hence $d(w_j)=d$ for all $j \neq i$. Therefore, the following must occur (note that in any case $d^*(v) \geq 2$):

(a) One of the neighbors of v has degree $d-2$, and the other three have degree d;

(b) One has degree $d-1$ and the other three have degree d;

(c) Two have degree $d-1$ and the other two have degree d; or

(d) All the four have degree d.

Let s, t, and u be the numbers of vertices of degree 4 satisfying (a), (b) and (c), respectively. Then Equation (2) is refined for $j=4$ as follows:

$$3s + 3t + 2u + 4(n_4 - s - t - u) \leq \Sigma\, i_4\, n_d(i_2, i_3, \ldots, i_7). \tag{4}$$

Thus, from (2), (3) and (4) we have

$$2n_2 + \{2r + 3(n_3 - r)\}/2 + \{3s + 3t + 2u + 4(n_4 - s - t - u)\}/3$$

$$+ 2n_5/4 + 2n_6/5 + 2n_7/6$$

$$\leq \sum_{j=2}^{7} \Sigma\, i_j\, n_d(i_2, i_3, \ldots, i_7)/(j-1)$$

$$= \Sigma\, n_d(i_2, i_3, \ldots, i_7) \sum_{j=2}^{7} i_j / (j-1). \tag{5}$$

We next show that if $n_d(i_2, i_3, \ldots, i_7) \neq 0$ then

$$\sum_{j=2}^{7} i_j / (j-1) \leq 1. \tag{6}$$

Let w be any vertex of degree d which has i_2 neighbors of degree 2, i_3 neighbors of degree 3, ... , i_7 neighbors of degree 7. Let t be the minimum degree of the neighbors of w, then $i_j \neq 0$ implies $t \leq j$. Let v be a neighbor of w with $d(v)=t$, then $d^*(w) \geq d-t+1$ since edge vw is not eliminatable. Therefore, we have

$$\sum_{j=2}^{7} i_j \leq t - 1,$$

which implies (6).

By the definition we have

$$n_d = \Sigma\, n_d(i_2, i_3, \ldots, i_7). \tag{7}$$

Combining (5), (6) and (7), we have

$$n_d \geq 2n_2 + \{2r + 3(n_3 - r)\}/2 + \{3s + 3t + 2u + 4(n_4 - s - t - u)\}/3$$

$$+ 2n_5/4 + 2n_6/5 + 2n_7/6,$$

which immediately yields the following.

$$(d-7)n_{d-1} + 2n_d \geq 4n_2 + 3n_3 + 2n_4 + n_5$$

$$+ 2(n_4 - s - t)/3 + (n_{d-1} - r - u)$$

$$+ \{2n_7/3 + (d-8)n_{d-1} - u/3\}. \tag{8}$$

The definition of s and t implies that

$$n_4 - s - t \geq 0. \tag{9}$$

If a vertex w of degree $d-1$ is adjacent with a vertex v of degree 3, then all the neighbors of w except v have degree d. Therefore, among n_{d-1} vertices of degree $d-1$ at most $n_{d-1} - r$ are adjacent with a vertex of degree 4, and furthermore each of these vertices is adjacent with at most two vertices of degree 4. Therefore, we have $t + 2u \leq 2(n_{d-1} - r)$, which implies,

$$n_{d-1} - r - u \geq 0. \tag{10}$$

We now show that

$$2n_7/3 + (d-8)n_{d-1} - u/3 \geq 0. \qquad (11)$$

Suppose first that $d=8$. Then, noting that $n_7=n_{d-1}$ and using (10), one can easily verify (11). Suppose next that $d \geq 9$. Then from (10), we easily obtain $(d-8)n_{d-1} - u/3 \geq 0$, implying (11).

Thus, from (8) - (11) we have

$$(d-7)n_{d-1} + (d-6)n_d \geq 4n_2 + 3n_3 + 2n_4 + n_5,$$

which contradicts (1).

Q.E.D

Technical Reports

Number	Author	Title
1/84	M. Sharir A. Schorr	On Shortest Paths in Polyhedral Spaces
2/84	S. Hart M. Sharir	Probabilistic Propositional Temporal Logics
3/84	J.E. Hopcroft J.T. Schwrtz M. Sharir	On The Complexity of Motion Planning For Multiple Independent Objects; Pspace Hardness Of The "Warehouseman's Problem"
4/84	B.A. Trakhtenbrot	A Survey of Russian Approaches To "PEREBOR": Brute Force Search
5/84	N.Alon M. Tarsi	Covering Multigraphs By Simple Circuits
6/84	J.Y. Halpern A.R. Meyer B.A. Trakhtenbrot	The Semantics of Local Storage, Or What Makes The Free List - Free ?
7/84	B.A. Trakhtenbrot J.Y. Halpern A.R. Meyer	From Denotatonal To Operational And Axiomatic Semantics For Algol- Like Languages: An Overview
8/84	I. Bar-On U. Vishkin	Optimal Parallel Generation of A Computation Tree Form
9/84	M. Atallah U. Vishkin	Finding Euler Tours In Parallel
10/84	N. Rishe	Semantics of Universal Languages and Information Structures In Data Bases
11/84	S. Hart M. Sharir	Nonlinearity of Davenport-Schinzel Sequences And Of Generalized Path Compression Schemes
12/84	R.E. Tarjan U. Vishkin	An Efficient Parallel Biconnectivity Algorithm
13/84	U. Vishkin	Optimal Parallel Pattern Matching In Strings
14/84	D. Leven M. Sharir	An Efficient And Simple Motion Planning Algorithm For A Ladder Moving In Two-Dimensional Space Amidst Polygonal Barriers

* The reports are available upon request.
Please write to Mrs. Dorit Barak,Eskenasy Institute of Computer Science
School of Math. Sci. Tel-Aviv University,
Ramat-Aviv, ISRAEL 69978.

Number	Author	Title
15/84	J. Gal-Ezer G. Zwas	Convergence Acceleration As A Computational Assignment
16/84	M. Jeger O. Kariv	Algorithms For Finding P-Centers On A Weighted Tree
17/84	E. Gabber A. Yehudai	Deducing Type Information From Context In Ada Based PDLs (see report number 35/85)
18/84	D. Levin	Multidimensional Reconstruction By Set-Valued Approximations
19/84	D. Gottlieb E. Tadmor	Recovering Pointwise Values Of Discontinuous Data Within Spectral Accuracy
20/84	U. Vishkin	An Optimal Parallel Algorithm For Selection
21/84	Y. Maon	On The Equivalence Problem Of Composition Of Morphisms And Inverse Morphisms On Context-free Languages
22/84	Y. Maon	On The Equivalence Of Some Transductions Involving Letter To Letter Morphisms On Regular Languages
23/85	S. Abarbanel D. Gottlieb	Information Content In Spectral Calculations
24/85	K. Kedem M. Sharir	An Efficient Algorithm For Planning Collision-free Translational Motion of a Convex Polygonal Object in 2-dimensional Space Amidst Polygonal Obstacles
25/85	A. Tamir	On The Solution Value Of The Continuous p-Center Location Problem On A Graph
26/85	H. Tal-Ezer	Spectral Methods In Time For Parabolic Problems
27/85	I.M. Longman	The Summation Of Power Series And Fourier Series
28/85	D. Leven M. Sharir	On Voronoi Diagrams for a Set of Discs
29/85	M. Sharir	Almost Linear Upper Bounds on The Length of General Davenport-Schinzel Sequences

Number	Author	Title
30/85	Y. Maon A. Yehudai	Balance of Many-valued Transductions and Equivalence Problems
31/85	G.M. Landau U. Vishkin	Efficient String Matching With k Mismatches
32/85	Y. Maon	Decision Problems Concerning Equivalence Of Transductions On Languages
33/85	J. Gal-Ezer G. Zwas	The Computational Potential Of Rational Approximations
34/85	D. Leven M. Sharir	Planning A Purely Translational Motion For A Convex Object In Two-Dimensional Space Using Generalized Voronoi Diagrams
35/85	E. Gabber A. Yehudai	Deducing Type Information from Context in Ada Based PDLs-A Revised version
36/85	G.M. Landau U. Vishkin	Efficient String Matching With k Differences
37/85	G.M. Landau U. Vishkin R. Nussinov	An Efficient String Matching Algorithm with k Differences for nucleotide and Amino Acid Sequences
38/85	E. Gabber	The Implementation Of The Algorithm for Deducing Type Information From Context In Ada Based PDLs
39/85	J. Reif M. Sharir	Motion Planning In The Presence Of Moving Obstacles
40/85	S. Sifrony M. Sharir	A New Efficient Motion-planning algorithm For A Rod In Tow-dimensional Polygonal Space
41/85	H.N. Gabow T. Nishizeki O. Kariv D. Leven O. Terada	Algorithms for Edge-Coloring Graphs