JOURNAL OF COMPUTER AND SYSTEM SCIENCES 38, 36–67 (1989)

# Embedding Planar Graphs in Four Pages

MIHALIS YANNAKAKIS

*AT & T Bell Laboratories, Murray Hill, New Jersey 07974*

Received September 23, 1986; revised August 25, 1987

An algorithm is presented which embeds any planar graph in a book of four pages. The algorithm runs in linear time. © 1989 Academic Press, Inc.

## 1. INTRODUCTION

A *book embedding* of a graph consists of an embedding of its nodes along the spine of a book (i.e., a linear ordering of the nodes), and an embedding of its edges on the pages so that edges embedded on the same page do not intersect. The objective is to minimize the number of pages used. The minimum number of pages in which a graph can be embedded is called the *pagenumber* of the graph.

The book embedding problem arises in connection with an approach to fault-tolerant VLSI design [R, CLR1, CLR2]. The graph models a desired interconnection pattern among a set of processors. The processors are arranged (physically or logically) on a line and are tested to determine which ones are good and which are faulty. The good processors are interconnected via "bundles" of wires running parallel to the line. Each bundle functions like a stack: Scan the line from left to right and suppose that a good processor $u$ wants to connect to some processor $v$ to its right. At $u$, the connection $(u, v)$ is pushed into one of the stack-bundles; that is, $(u, v)$ occupies the bottom wire of the bundle, while the other connections that are currently in this bundle are shifted up one place. At $v$, the connection $(u, v)$ is popped from the stack. The problem is to realize the desired interconnection graph using the minimum number of stack-bundles. This is equivalent to the problem of embedding the graph in a book with the minimum number of pages. The pages correspond to the bundles. The constraint that edges on the same page do not intersect corresponds to the LIFO property of stacks.

Computationally, the book embedding problem is hard: it is NP-complete to tell if a planar graph can be embedded in two pages [W, CLR2]. Note that in this case the crux of the problem is in the node-embedding part, as once this is fixed, it is easy to tell whether the edges can be embedded in two pages. The subproblem of embedding optimally the edges for a fixed node-embedding had been studied earlier in connection with other problems such as sorting permutations with stacks (see [EI; T; GO, Chap. 11]); this problem is also NP-complete [GJMP].



0022-0000/89 $3.00

Graphs with pagenumber one are exactly the outerplanar graphs. Graphs with pagenumber two are the subhamiltonian planar graphs: these are the subgraphs of planar Hamiltonian graphs. As there are triangulated (maximal) planar graphs which are not Hamiltonian, this implies that there are planar graphs which require at least three pages [BK]. Berhart and Kainen conjectured that planar graphs have unbounded pagenumber, but this was disproved in [BS] and [H]. Buss and Shor gave an algorithm, based on Whitney's theorem, which embeds all planar graphs in nine pages [BS]. Heath used a method of "peeling" the graph into levels to reduce the number to seven [H]. Recently, Istrail found a six-page algorithm [I].

In this paper we show that all planar graphs can be embedded in four pages. The proof is constructive; that is, we actually present an algorithm which constructs the embedding. The algorithm runs in linear time. Four pages are also necessary for planar graphs: as we show in another paper [Y], there are planar graphs that cannot be embedded in three pages.

The rest of the paper is organized as follows. In preparation for the algorithm, we analyze in Section 2 a simple class of planar graphs. We prove several results and present a book-embedding algorithm for these graphs. In Section 3 we describe the four-page algorithm for general planar graphs. In Section 4 we prove its correctness, and in Section 5 we analyze its time complexity.

## 2. Two-Level Graphs

We assume familiarity with basic terminology and results from graph theory. Let $G=(N, E)$ be a graph with $n$ nodes. A (linear) *layout* $L$ of the nodes of $G$ is an ordering of $N$; i.e., a one-to-one function from $N$ to $\{1, ..., n\}$. We say that two edges $(a, b)$ and $(c, d)$ *conflict* in the layout $L$ if $L(a)<L(c)<L(b)<L(d)$ or $L(c)<L(a)<L(d)<L(b)$; that is, if we place the nodes on a horizontal line ordered by $L$ and draw the edges above this line, two edges conflict if they intersect. A *book embedding* of $G$ in $k$ pages consists of (1) a linear layout $L$ of its nodes, and (2) a coloring of its edges with $k$ colors (the "pages") so that conflicting edges in $L$ receive different colors.

Clearly, if a graph can be embedded in a given number of pages, the same is true of its subgraphs. Therefore, we can assume without loss of generality that our planar graph is triangulated (all faces are triangles). We will assume from now on a fixed embedding of the graph on the plane; the embedding is inherited by its subgraphs. To avoid confusion, we will reserve the term "embedding" for the embedding of the graph on the plane, and use the term "layout" for the book embedding.

The nodes of a planar graph can be partitioned into *levels* according to their "distance" from the outer face: Nodes on the outer face are at level 0. Delete these nodes; the nodes that lie now on the outer face are at level 1. In general, level $t$ consists of the nodes that lie on the outer face after deleting the nodes at levels less than $t$. The edges are partitioned into *level* edges, edges that connect nodes at the

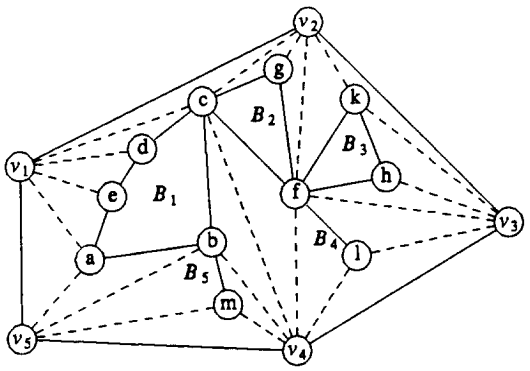


FIGURE 1

same level, and *binding* edges, edges that connect nodes at different levels. The fact that a level $i$ node is not on the outer face after deleting the first $i-2$ levels implies that every level $i$ node lies in the interior of some cycle composed of level $i-1$ nodes. This means in particular that a level $i$ node cannot be adjacent to a level $j$ node with $j < i-1$; i.e., binding edges connect only consecutive levels.

Our algorithm follows the "peeling" into levels to lay out the graph, working from the outside in. First it lays out level 0, the boundary of the outer face. Then it lays out level 1, coloring also the binding edges between levels 0 and 1. Now the rest of the graph is in the interior of level 1 cycles. The algorithm takes each level 1 cycle in turn and lays out its interior in a similar way.

Before going into more detail, we will analyze in this section a simple class of graphs having only two levels. We shall show that three pages suffice for these graphs. (It is easy to see that three pages are also necessary.) More importantly, we shall show several results and layout techniques which will be used in the general algorithm that is described in the next section.

For the rest of this section we fix $H$ to be a planar (embedded) graph with two levels, all of whose faces are triangles, except for the outer face which is bounded by a cycle $K$ of arbitrary length. Furthermore, we assume that $K$ has no chords; i.e., there are no edges connecting two nonconsecutive nodes of $K$.[1] We call the nodes of $K$ the *outer* nodes and enumerate them around the cycle in clockwise order as $v_1, v_2, \ldots, v_k$. The rest of the nodes of $H$ are at level 1; we call them the *inner* nodes. We denote by $I$ the subgraph induced by the inner nodes, and call it the inner graph. By the definition of levels, $I$ is an outerplanar graph. Figure 1 shows an example of a graph $H$. In the figure we have only included the level 1 edges that are on the boundary (on the outer face) of $I$. We will be interested in the part of the

[1] Of course, this is not the most general case of a two level graph. The general case is treated at the end of this section.

graph that is interior to the cycle $K$ and exterior to the inner graph $I$. The interior of $I$ will not be important for our discussion; in particular, it does not matter if it is not triangulated.

We start with some simple observations.

• Every outer node is adjacent to some inner node. To see this, note that every edge of the cycle $K$ belongs to two faces; the outer face, and a triangular inner face. The third node of the latter face cannot be a node of $K$, because $K$ has no chords. Therefore it must be an inner node.

• Every inner node is adjacent to some outer node. Since every inner node $u$ is on the boundary of $I$, it belongs to at least one face of $G$ that is exterior to $I$. This face cannot consist only of inner nodes, because then it would be interior to $I$. Therefore, it must contain an outer node which is adjacent to $u$, because the face is a triangle.

• The inner graph $I$ is connected. This was shown in Lemma 4.1 of [H]. Another way to see it is to use the first observation: Suppose that $M$ is a connected component of $I$ and let $H - M$ be the subgraph of $H$ obtained by deleting all nodes of $M$; this subgraph inherits a planar embedding from $H$. Since $M$ is connected, all nodes of $M$ must lie in the same face $F$ of $H - M$. Insertion of $M$ results in the triangulation of $F$. As in the first observation, every node of $F$ must be adjacent to some node of $M$. Since $M$ is a connected component of $I$, this implies that $F$ has only outer nodes. Since the cycle $K$ has no chords, this means that $F$ is bounded by the cycle $K$, and therefore there are no other inner nodes besides those of $M$.

Decompose the inner graph into its biconnected components (or blocks). From now on "blocks" will always refer to the blocks of $I$. In the example of Fig. 1 there are five blocks: $B_1, ..., B_5$. A block may be either trivial—consist of a single edge—or nontrivial—consist of a cycle possibly with some internal chords. (Recall that $I$ is outerplanar.) Every edge on the boundary of $I$ belongs to two faces. If the edge is a nontrivial block, then one face is in the interior of $I$ and one in the exterior. If the edge is in a trivial block (i.e., is a block by itself), then both faces are exterior to $I$.

We say that a node $x$ *sees* an edge $(y, z)$ if $x$ is adjacent to $y$ and $z$, and furthermore the triangle $(x, y, z)$ is a face. We say that an outer node *sees* a block (of $I$) if it sees an edge of the block. For example, in Fig. 1, node $v_1$ sees the edge $(a, e)$, and therefore block $B_1$.

Consider the inner face (a triangle) which contains the edge $(v_1, v_k)$ that connects the first and last nodes of the cycle $K$. The third node of this face is called the *first inner* node and is denoted by $a$. It is convenient to assume that the first inner node $a$ belongs to only one block (i.e., it is not a cutpoint of the inner graph $I$), and furthermore, that this block is seen by $v_1$. If this is not the case, then add a new node $a'$ inside the triangle $(v_1, a, v_k)$ and connect it to the three nodes of the triangle. The first inner node of the resulting graph is the new node $a'$; it belongs to only one block, namely the trivial block $(a, a')$, which is seen by $v_1$.

Recall some properties of the decomposition of graphs into blocks. (These properties hold for arbitrary graphs, not only for planar graphs.) If $a$ is a node and $B$ a block that does not contain $a$, then all paths from $a$ to the block $B$ enter $B$ through the same node of $B$; also, on all the $a-B$ paths, the last edges that are not in $B$ belong to the same block.

Consider the inner graph $I$ as being "rooted" at the first inner node $a$. This induces a rooted tree structure $T$ on the blocks. The root is the block $B_1$ that contains $a$. For any other block $B$, the parent of $B$ is the (unique) block that contains the last edge outside $B$ of any path from $a$ to $B$. We define the *leader* of a block $B$ to be the first node of $B$ that is encountered in any path from $a$ to $B$. Thus, the leader of the root block is node $a$; for any other block $B$, the leader of $B$ is the node of $B$ that belongs also to its parent in the tree $T$. In Fig. 2 we show the tree $T$ of blocks for the example of Fig. 1.

An inner node (in particular a cutpoint of $I$) may belong to more than one block. We *assign* each inner node $u$ to a unique block as follows: node $u$ is assigned to the highest block (in the tree $T$) that contains it. Thus, the root block of $T$ is assigned all the nodes that it contains. Any other block is assigned all its nodes except its leader.

We view the outer nodes as ordered from $v_1$ to $v_k$. If $S$, $U$ are two subsets of outer nodes where each member of $S$ precedes or is equal to each member of $U$ (in this ordering), we write $S \leqslant U$. If $u$ is an inner node, we denote by $\Gamma(u)$ the set of outer nodes adjacent to $u$.

LEMMA 1. *Let $B$ be a block, $u$ its leader. Let $u, u_1, \ldots, u_t$ be the order in which the nodes of $B$ are encountered in a clockwise traversal of its boundary starting with $u$. (If $B$ is trivial, then $t=1$.)*

(1) $\Gamma(u_1) \leqslant \Gamma(u_2) \leqslant \cdots \leqslant \Gamma(u_t)$.

(2) *Let $v_f$ be the first outer node adjacent to $u_1$, and $v_l$ the last outer node adjacent to $u_t$. Then $v_f$ and $v_l$ are distinct, $v_f$ sees the edge $(u, u_1)$, and $v_l$ sees the edge $(u_t, u)$. The leader $u$ is not adjacent to any outer node strictly between $v_f$ and $v_l$.*

*Proof.* Consider the block $B$ together with a path from the first inner node $a$ to $u$ (the trivial path if $B$ is the root block in which case $a=u$), the edges from $a$ to $v_1$ and $v_k$, and an arbitrary binding edge $(u_i, v_m)$. The interior of the cycle $K$ is par-

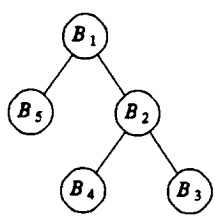


FIGURE 2

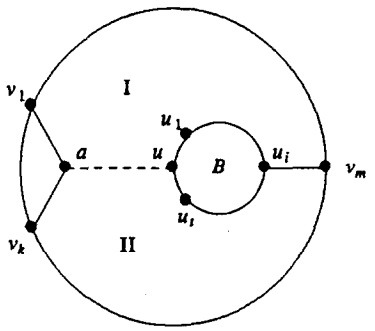


FIGURE 3

titioned into the face $(v_1, v_k, a)$, the interior of $B$ (if $B$ is a nontrivial block), and the two regions indicated as regions I and II in Fig. 3. Every binding edge must lie in region I or II. If $u_j$ is another node of $B$ with $j > i$, then $u_j$ is only in region II, and therefore can only reach outer nodes between $v_m$ and $v_k$. This establishes part (1).

For the second part, suppose first that $B$ is trivial, i.e., the edge $(u, u_1)$. Then this edge is seen by two outer nodes. Thus, $u_1$ is adjacent to at least two outer nodes and $v_f \neq v_l$. For an inner node to reach an outer node strictly between $v_f$ and $v_l$, the inner node must lie in the interior of the cycle formed by the edges $(u_1, v_f)$, $(u_1, v_l)$ and the arc of the cycle $K$ from $v_f$ to $v_l$. Since $u$ is outside this region, we conclude that $u$ cannot be adjacent to any such node. It follows that the only outer nodes that could possibly see $(u, u_1)$ are $v_f$ and $v_l$.

Suppose that $B$ is nontrivial, and let $u_i$ be $u_1$ and $v_m$ be $v_f$ in Fig. 3. Let $v_j$ be the outer node that sees the edge $(u, u_1)$. Since $v_f$ is the first outer node adjacent to $u_1$, we have $v_j \geqslant v_f$. On the other hand, for the triangle $(v_j, u, u_1)$ to be a face, the edge $(u, v_j)$ must be in region I; otherwise it would contain $u_t$ in its interior. Thus, $v_j \leqslant v_f$, and we conclude that $v_f$ sees the edge $(u, u_1)$.

Now even if $u_t$ is adjacent to $v_f$, the triangle $(u, u_t, v_f)$ could not be a face because it would contain $u_1$ in its interior. Therefore, the node that sees the edge $(u, u_t)$ is distinct from $v_f$. By a symmetric argument this node must be $v_l$. The fact that $u$ is not adjacent to any outer node (strictly) between $v_f$ and $v_l$ is similar to the case when $B$ is trivial. ∎

We say that an outer node $v_i$ is *adjacent* to a block $B$ if $v_i$ is adjacent to some node assigned to $B$. (Recall that, unless $B$ is the root, the leader of $B$ is not assigned to $B$.) We denote by $\Gamma(B)$ the set of outer nodes adjacent to $B$. By Lemma 1, there are at least two such nodes.

LEMMA 2. *Let $B$, $B'$ be two blocks. Let $v_f$, $v_l$ be respectively the first and last outer nodes adjacent to $B$.*

(1) *If $B'$ is an ancestor of $B$ in the tree $T$, then $\Gamma(B')$ has both a node smaller or equal to $v_f$ and a node larger or equal to $v_l$.*

(2) *If $B'$ is not a descendant of $B$, then $B'$ is not adjacent to any outer node strictly between $v_f$ and $v_l$.*

(3) *If $B$ and $B'$ are not related in the tree $T$ (i.e., neither is a descendant of the other), then $\Gamma(B) \leqslant \Gamma(B')$ or $\Gamma(B') \leqslant \Gamma(B)$.*

*Proof.* (1) Suppose that $B'$ is the parent of $B$. Then the leader $u$ of $B$ is assigned to $B'$. From Lemma 1, $v_f$ and $v_l$ see edges incident to $u$, and therefore are both adjacent to $u$. Thus, the claim holds for the parent of $B$, and follows for all the ancestors by transitivity.

(2) Let $u$ be the leader of $B$, $u_1$ the first node in a clockwise ordering of the boundary of $B$, and $u_t$ the last node. Consider the block $B$, a path from the first inner node $a$ to $u$, the edges from $v_f$ to $u$ and $u_1$, from $v_l$ to $u$ and $u_t$, and from $a$ to $v_1$ and $v_k$. The interior of the cycle $K$ is partitioned into regions as shown in Fig. 4. Outer nodes strictly between $v_f$ and $v_l$ are reachable only from region III. For node $a$ to reach this region, it must go through block $B$. That is, all blocks with nodes in region III are descendants of $B$.

(3) All the nodes (and edges) on the path from $a$ to $u$ are assigned to blocks that are ancestors of $B$ in the tree. Since $B'$ is not related to $B$, and the nodes assigned to $B'$ induce a connected subgraph, they must all be either inside region I or inside region II. In the first case $\Gamma(B') \leqslant \Gamma(B)$, and in the second case $\Gamma(B) \leqslant \Gamma(B')$. ∎

We call the first outer node adjacent to a block $B$ (node $v_f$ of Lemma 1), the *dominator* of the block, and denote it as dom($B$). In the example of Fig. 1, the dominator of $B_1$ is $v_1$, and the dominator of $B_2$ and $B_3$ is $v_2$.

LEMMA 3. *The blocks dominated by the same outer node $v_i$ form a directed path in the tree $T$.*

*Proof.* Two unrelated blocks cannot have the same dominator because by Lemma 2, the dominator of one block is greater or equal to the last node adjacent

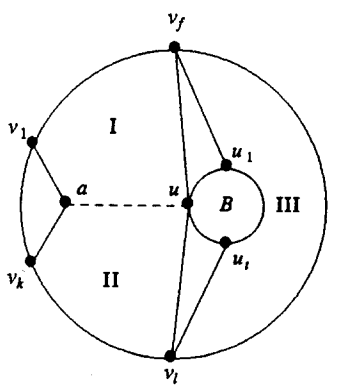


FIGURE 4

to the other block, which in turn is strictly greater than its dominator. Thus, all blocks dominated by $v_i$ are related.

From part (1) of Lemma 2, the dominator is monotonically nondecreasing along any path down the tree $T$. It follows that the blocks dominated by $v_i$ form a (directed) path in $T$. ∎

We are ready now to describe a three-page layout for our two-level graph $H$.

*Node Layout*

The outer nodes are laid out in the order $v_1, ..., v_k$. The inner nodes assigned to each block $B$ are placed right after the outer node $v_i$ that dominates $B$ (i.e., between $v_i$ and $v_{i+1}$). The nodes assigned to the blocks dominated by a common outer node $v_i$ are ordered according to either of the following two methods.

*Nested Method.* The blocks dominated by $v_i$ form a directed path in the tree $T$. Starting from the leader of the highest block, traverse the boundary of the subgraph induced by these blocks in counterclockwise order, and list the nodes assigned to them as they are first encountered. For example, with this method, the nodes assigned to blocks $B_2$, $B_3$ dominated by $v_2$ in Fig. 1 will be ordered as follows: $f, h, k, g$.

*Consecutive Method.* The blocks dominated by $v_i$ are listed one after the other in top-down order of $T$; i.e., first the nodes assigned to the highest block, then the nodes assigned to its child, and so on. To order the nodes assigned to block $B$, starting at the leader of $B$ traverse its boundary in counterclockwise order, and list the nodes assigned to $B$ as they are encountered. With this method, the nodes assigned to the blocks $B_2$, $B_3$ will be ordered as follows: $f, g, h, k$.

The choice of which method to use for the blocks dominated by each outer node is arbitrary; that is, we may use the nested method for some outer nodes and the consecutive method for the rest.

*Edge Coloring*

Let the three colors be 1, 2, 3. All level 0 edges (i.e., edges connecting outer nodes) receive color 1. The rest of the edges are either binding or level 1 edges. Every level 1 edge belongs to exactly one block. We also assign each binding edge $(v_i, u)$ to a unique block, namely, the block to which the inner node $u$ of the binding edge has been assigned. To color the rest of the edges, we associate a color 2 or 3 with each block according to the following rules.

*Rule* 1. The root block gets arbitrarily color 2 or 3.

*Rule* 2. Let $B$ be a nonroot block and $v_i$ its dominator. If the parent of $B$ has a different dominator, then $B$ gets a color opposite to that of its father. If the parent of $B$ has the same dominator $v_i$, and if the blocks dominated by $v_i$ are ordered by the nested method (resp. the consecutive method), then $B$ gets the same (resp. opposite) color as its father.

Every level 1 edge gets the color of its block. To color the binding edges we partition them into *back* and *forward*. A binding edge $(v_i, u)$ is a back edge if $v_i$ comes before $u$ in the layout; otherwise, it is forward. All back edges get color 1. A forward edge assigned to a block $B$ gets color 2 or 3, opposite to that of $B$.

In Fig. 5 we show the layout for the example of Fig. 1. In this layout, the blocks $B_2$ and $B_3$ which are dominated by $v_2$ have been ordered by the consecutive method. The solid edges in the upper halfplane have color 2, while those in the lower halfplane have color 3. The dashed edges have color 1.

We shall argue now that this indeed a legal layout, i.e., that no two conflicting edges have the same color. First some observations about the node layout. From Lemma 1, each block is adjacent to at least two nodes. Thus, no block has $v_k$ (the last node of the cycle $K$) as its dominator. This means that all inner nodes are laid out between $v_1$ and $v_k$. A second observation is that, if $v_i$ is the dominator of $B$, then no outer node before $v_i$ is adjament to $B$ (by definition); therefore, if $(v_i, u)$ is a back edge, then $u$ is between $v_i$ and $v_{i+1}$. Consequently, there is no conflict between any two edges with color 1.

Let $B$ be any block. The leader of $B$ precedes in the layout all other nodes of $B$. This is obvious if $B$ is the root block; the first inner node $a$ appears first among all inner nodes in the layout. If $B$ is not the root block, then the leader of $B$ is assigned to the parent of $B$ in the tree. By Lemma 2, the parent has either the same or a smaller dominator. In either case, the leader precedes in the layout all other nodes of $B$, regardless of whether the nested method or the consecutive method is used. Thus, the nodes of $B$ are laid out in counterclockwise order of its boundary, starting with the leader.

LEMMA 4. *Let $B$ a block, $u$ its leader, and $u_1, ..., u_t$ the rest of the nodes of $B$ in clockwise order. Let $v_f$ be the dominator of $B$, and $u_q$ the node of $B$ with highest index $q$ that is adjacent to $v_f$ (see Fig. 6). If $q > 1$, then the nodes $u_1, ..., u_{q-1}$ are adjacent to $v_f$ but to no other outer node. All level 1 edges incident to $u_1, ..., u_{q-1}$ belong to block $B$. If $B$ has a child $B'$ which is also dominated by $v_f$, then the leader of $B'$ is $u_q$.*

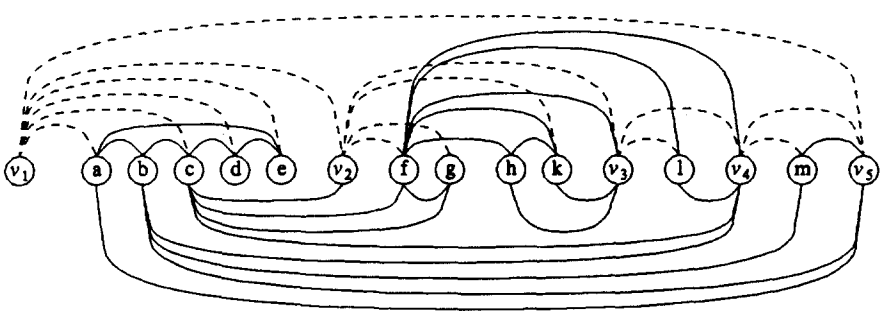


FIGURE 5

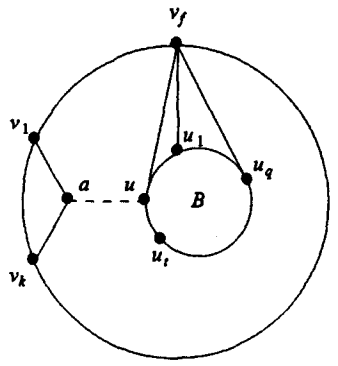


FIGURE 6

*Proof.* The first part follows from Lemma 1: $v_f \leqslant \Gamma(u_1) \leqslant \cdots \leqslant \Gamma(u_{q-1}) \leqslant v_f$, and therefore, $\Gamma(u_1) = \cdots = \Gamma(u_{q-1}) = \{v_f\}$. For the second part, note that if a node $u_i$ assigned to $B$ belongs also to another block $B'$, then $u_i$ must be the leader of $B'$. But the leader of any block must be adjacent to at least two outer nodes, namely the first and last outer nodes adjacent to the block. Therefore, $u_1, \ldots, u_{q-1}$ do not belong to any other block.

Finally, suppose that $B$ has a child $B'$ which is dominated by $v_f$. Then the leader of $B'$ must be adjacent to $v_f$. The leader of $B'$ cannot be $u$, because then $B'$ would not be a child of $B$. Also, as we just showed, it cannot be one of $u_1, \ldots, u_{q-1}$. Thus, it has to be $u_q$. ∎

The nodes of a block $B$ are not necessarily consecutive in the layout. If the consecutive method is used to lay out $B$, then all the nodes of $B$ except for the leader are consecutive. If the nested method is used to lay out $B$, and the dominator of $B$, call it $v_f$, dominates also some descendants of $B$, then there is one more interruption. In particular, if $u$ is the leader of $B$, and $u_1, \ldots, u_t$ the rest of its nodes in clockwise order (notation as in Lemma 4), the portion of the node layout containing the nodes of $B$ is as follows: $u$, possibly some other nodes, $u_t, \ldots, u_q$, posibly some other nodes, $u_{q-1}, \ldots, u_1$. The dots represent nodes of $B$ in order of decreasing index. The other nodes after $u_q$ are the nodes assigned to descendants of $B$ which are dominated by the same node $v_f$ as $B$. Node $u_q$, the leader of the child of $B$ that is dominated by $v_f$, is as in Lemma 4. It is possible that $q = t$ or that $q = 1$ (or both in case $t = 1$ and $B$ is a trivial block). From Lemma 4, no forward binding edge is incident to a node $u_i$, $i < q$, from the last part of $B$. Also, all level 1 edges incident to a node $u_i$ from the last part of $B$ belong to block $B$.

An implication of these observations is, that if we restrict attention to the inner nodes that are incident to foward binding edges, then the nested and the consecutive method lay out these nodes in exactly the same way. The same is true if we just look at the nodes that belong to at least two blocks.

LEMMA 5. *There is no conflict between any pair of forward binding edges.*

*Proof.* Let $(x, v_i)$, $(y, v_j)$ be two forward binding edges, and assume without loss of generality that $x$ precedes $y$ in the layout. We want to show that either $v_i \leqslant y$ or $v_i \geqslant v_j$.

Suppose first that $x$ and $y$ are assigned to the same block $B$. Assume that one of the two nodes is the first inner node $a$. Then $a = x$, $B$ is the root block, and both $a$ and $y$ are laid out between $v_1$ and $v_2$. From Lemma 1, $a$ is not adjacent to any outer node strictly between $v_1$ (the first outer node that sees the root block) and $v_j$ (an outer node adjacent to some other node of the block). Thus, $v_i \geqslant v_j$. If $a \neq x$, then $x$ and $y$ are two nodes of $B$ other than its leader. The claim follows from part (1) of Lemma 1.

Suppose now that $x$ and $y$ are assigned to different blocks $B_x$ and $B_y$, respectively. Since $x$ precedes $y$ in the layout, $\mathrm{dom}(B_x) \leqslant \mathrm{dom}(B_y)$. In case of equality, $B_x$ must be an ancestor of $B_y$, regardless of whether the nested method or the consecutive method was used to lay out these blocks. This is obvious if the consecutive method was used and follows from our previous observations in case the nested method was used, because $x$ and $y$ are incident to forward binding edges.

If $B_x$ and $B_y$ are unrelated, then $\mathrm{dom}(B_x) < \mathrm{dom}(B_y)$, and by Lemma 2, $\Gamma(B_x) \leqslant \Gamma(B_y)$. Therefore, $v_i$ precedes (or is equal to) the dominator of $B_y$, which precedes $y$ in the layout.

Suppose that $B_x$ and $B_y$ are related. Then $B_x$ is an ancestor of $B_y$, because the dominator is monotonically nondecreasing along any path down the tree $T$. From part (2) of Lemma 2, $B_x$ is not adjacent to any outer node strictly between $\mathrm{dom}(B_y)$ and the last outer node aldjacent to $B_y$. Thus, either $v_i \leqslant \mathrm{dom}(B_y)$ and $v_i$ precedes $y$, or $v_i \geqslant v_j$. ∎

Thus, the only possible conflicts are between two level 1 edges, or a forward binding and a level 1 edge. Since the nodes of each block are laid out in circular (counterclockwise) order of its boundary, no two level 1 edges of the same block conflict.

LEMMA 6. *Let $(w, x)$, $(y, x)$ be two level 1 or forward binding edges which conflict. Suppose (without loss of generality) that in the layout $L$ these nodes are ordered as follows: $L(w) < L(y) < L(x) < L(z)$. Then $(w, x)$ is a level 1 edge. Let $B$ be its block.*

*If the consecutive method was used to lay out $B$ (i.e., the blocks dominated by $\mathrm{dom}(B)$), then $y$ is an inner node assigned to $B$, and $(y, z)$ is either a (forward) binding edge, or is a level 1 edge that belongs to a child $B'$ of $B$.*

*If the nested method was used to lay out $B$, then $y$ is an inner node assigned to a block $B_y$, with the same dominator as $B$. Either $(y, z)$ is a (forward) binding edge, or it is a level 1 edge that belongs to a child $B'$ of $B_y$ with a different dominator.*

*Proof.* We shall show first that $(w,x)$ cannot be a forward binding edge. If this were the case then, by Lemma 5, $(y, z)$ should be a level 1 edge. Let $B$ be its block. The outer node $x$ is laid out beteen the two inner nodes $y$ and $z$. This implies that $y$ is the leader of $B$. The last node $v_l$ adjacent to $B$ is laid out after $z$ and is adjacent to

the leader $y$. This means that two binding edges, namely $(w, x)$ and $(y, v)$ conflict, contradicting Lemma 5.

We conclude that $(w, x)$ is a level 1 edge, and let $B$ be its block. Since $x$ is not the first node of $B$ in the layout, it is not the leader of $B$. We claim that $\text{dom}(B)$ is to the left of $y$. Clearly, it has to be left of $x$. If it was between $y$ and $x$ then $w$ would have to be the leader of $B$, and we would have a conflict between the binding edge $(w, \text{dom}(B))$ and the edge $(y, z)$, which we just ruled out.

Clearly, $y$ is an inner node; otherwise $(y, z)$ would not be a forward or a level 1 edge. Let $B_y$ be the block to which $y$ is assigned. Since $y$ is between $\text{dom}(B)$ and $x$, and $x$ is assigned to $B$, we conclude that $\text{dom}(B_y) = \text{dom}(B)$. We distinguish now two cases depending on which method was used to lay out the blocks dominated by $\text{dom}(B)$.

Suppose that the consecutive method was used. The fact that $w$ precedes $y$ which precedes $x$ implies that $B$ is both an ancestor and a descendant of $B_y$; that is, $B = B_y$. The edge $(y, z)$ is either binding or a level 1 edge. In the latter case it must belong to another block $B'$. Since $y$, a node of $B'$, is assigned to a different block, namely $B$, node $y$ is the leader of $B'$, and $B'$ is a child of $B$.

Suppose that the nested method was used. Since $y$ is between two nodes of $B$, the block $B_y$ is a descendant (not necessarily proper) of $B$. The edge $(y, z)$ is either binding or a level 1 edge. In the latter case let $B'$ be its block. Suppose that $\text{dom}(B') = \text{dom}(B)$. Since $y$ ( a node of $B'$) is between $w$ and $x$ (two nodes of $B$), $B'$ must be a descendant of $B$. Similarly, since $x$ is between $y$ and $z$, $B$ must be a descendant of $B'$. It follows that $B = B'$, and two edges of $B$ conflict, which is impossible. Thus, $\text{dom}(B')$ is different (in fact greater) than $\text{dom}(B)$. It follows that $y$ is the leader of $B'$, and $B'$ is a child of $B_y$. ▮

THEOREM 1. *No two edges with the same color conflict; i.e., the layout is legal.*

*Proof.* We already argued that no two edges with color 1 conflict. The theorem follows immediately from Lemma 6 and the rules of the edge coloring. ▮

The algorithm we described for laying out the graph $H$ in three pages can be extended to all two-level graphs. A typical two-level graph may not be biconnected. By a result of [BK], the pagenumber of any graph is equal to the maximum pagenumber of its biconnected components; furthermore, one may easily combine $k$-page layouts of the biconnected components to produce a $k$-page layout for the whole graph. Thus, it suffices to look at biconnected two-level graphs. A typical such graph is bounded by a cycle $K$. The cycle may in general have chords, and its interior may not be triangulated. We can color the chords of $K$ with color 1, and ignore them (remove them from the graph). After that, we can add edges to triangulate the interior of the cycle $K$, without introducing any chords and without destroying the two-level property; it is easy to see that this is possible. At this point we have a two-level graph $H$ which is bounded by a chordless cycle, all of whose inner faces are triangles, and we may apply the algorithm.

The algorithm can be also easily extended to lay out all planar graphs in 5 pages. Rather than giving the details, we proceed to the four-page algorithm.

## 3. The Four-Page Algorithm

For the purposes of our description of the algorithm, it will be more convenient to regard a node layout as a mapping from the nodes into distinct points of the real line. We will identify a node with its image on the real line. Thus, if $u$ and $v$ are nodes, we will say the interval $[u, v]$ instead of "the (closed) interval of the real line between the images of $u$ and $v$."

The heart of the algorithm is a recursive procedure EXPAND($C$), where $C$ is a cycle of the graph. The purpose of the procedure is to add to an already constructed partial layout the nodes and edges that lie in the interior of $C$. Besides $C$, there are several more parameters, which we will describe as we go along.

The procedure EXPAND is called only on cycles $C$ that have no external chords, i.e., no edges which lie in the exterior of $C$ and connect two nonconsecutive nodes of $C$. When EXPAND($C$) is called, the algorithm has already constructed a layout $L$ of a subgraph. This layout has several properties, which we will now state. It will be helpful for the reader to recall the layout of the previous section and think of $C$ as the cycle that bounds a nontrivial block of the inner graph.

There are three properties P1–P3 concerning the node layout part of $L$ and three properties P4–P6 concerning the edge coloring part of $L$.

P1. The layout $L$ includes all nodes of $C$, listed in circular (clockwise or counterclockwise) order. It does not indclude any nodes inside $C$ (in the planar embedding).

To avoid repeated subscripts, we denote the nodes of $C$ by 1, 2, ..., $p$ in the order in which they apper in $L$. One of the extreme nodes of $C$, node 1 or node $p$, is specified as the *leader* of $C$. For concreteness, we assume without loss of generality that the leader is the first node 1; otherwise, reverse the layout. It follows immediately from the definition of conflict, that two edges conflict in a layout if and only if they conflict in the reverse layout. Thus, by reversing a legal layout we get another legal layout with the same edge coloring.

In Fig. 7 we present a schematic picture of the layout which illustrates the various properties. The solid and dashed lines of the figure represent edges with two different colors, whose other endpoints lie somewhere outside the interval $[2, p]$.

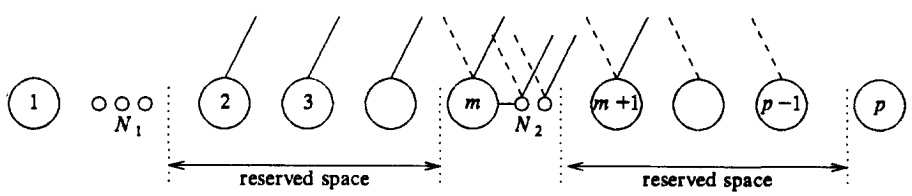


FIGURE 7

P2. There is a node $m$ of $C$, other than its leader (i.e., $2 \leqslant m \leqslant p$) such that, for all $i \neq m$ with $2 \leqslant i \leqslant p-1$, node $i+1$ of $C$ follows immediately node $i$ in the layout $L$.

That is, all intervals $(i, i+1)$ between two consecutive nodes of $C$ are empty (contain no nodes), except possibly for the interval $(1, 2)$ next to the leader, and for the interval $(m, m+1)$, if $m < p$. We call $m$ the *turning point* of $C$, and the interval $(m, m+1)$ the *border*. Referring to the previous section, if $C$ is the cycle that bounds a nontrivial block, then $m$ is node $u_q$ in the notation of Lemma 4; that is, $m$ is the first node in the layout (other than the leader) which is adjacent to the dominator of the block. Let $N_1, N_2$ be respectively the sets of nodes in the intervals $(1, 2)$ and $(m, m+1)$.

P3. Nodes of $N_2$ are not adjacent to any nodes of $C$ except to only one of the border nodes ($m$ or $m+1$).

Of course, if $m = p$ (in which case there is no border), property P3 is vacuous. If $m < p$, we call that border node which is adjacent to $N_2$ the *attachment point*, and the other border node the *free point*. In Fig. 7 the attachment point is $m$. If $N_2 = \varnothing$ (or $N_2$ is not adjacent to any node of $C$), the choice of attachment and free point is arbitrary.

We come now to the properties satisfied by the edge coloring part of the layout $L$. Of course, first of all, $L$ is a legal 4-page layout. Only edges connecting nodes that are included in $L$ have been colored, but not all such edges have received yet a color.

P4. Among the edges connecting nodes of the cycle $C$, only the edges incident to the leader are colored in $L$. All of them, except possibly for the edge $(1, p)$ have the same color, called the *leader's color*, and denoted by $c_0$.

We say that an edge $e$ *exits* an interval if one node of $e$ is in the interval and one outside. In the following property, $m$ is the turning point of $C$, as in P2.

P5. There are colors $c_1$ and $c_2$, with $c_1 \neq c_2$, $c_1 \neq c_0$, such that the following are true of the edges that are colored in $L$ and are not incident to the first and last nodes of $C$ (nodes 1 and $p$).

(a) All edges exiting the interval $[2, m-1]$ have color $c_1$;

(b) all edges exiting the interval $[m+2, p-1]$ have color $c_2$;

(c) all edges exiting the interval $[m, m+1]$ have color $c_1$ or $c_2$.

We call $c_1$ the *first* color, and $c_2$ the *second* color of $C$. The second color $c_2$ may be the same as the leader's color $c_0$. If an interval is not well-defined in property P5, then the corresponding clause is vacuous. Thus, (a) is vacuous if $2 = m$, (b) if $m \geqslant p-2$, and (c) if $m = p$. If $m = p-1$, clause (c) constraints only edges incident to $p-1$ and $N_2$, but not the edges incident to $p$.

By P2 and P4, if $m > 2$, the edges of clause (a) are exactly those colored edges that are incident to nodes $2, \ldots, m-1$ but not the leader. That is, (a) can be written

equivalently as: all (colored) edges which are incident to nodes $2, ..., m-1$ but not to the leader, have color $c_1$. Similarly, clause (b) constrains all colored edges that are incident to nodes $m+2, ..., p-1$ but not to the leader. In words, property P5 says the following. As we move along the layout from node 2 to node $p-1$, we see for a while only one color, $c_1$, up to node $m$. At the border we may see edges exiting that have a second color $c_2$, but then from node $m+2$ up to $p-1$ we only see that second color. Referring to the previous section, if $C$ is the cycle that bounds a nontrivial block, then $m$ is the first node in the layout (other than the leader) that is adjacent to the dominator of the block; note that nodes of the block that precede $m$ in the layout are not incident to any back edges, while nodes $m+1, ..., p$ are only incident to back edges.

The final property is:

P6. There is no edge of color $c_0$ connecting node 2 to a node of $N_1$.

This property follows from P5, except when $c_0 = c_2$ and $m = 2$.

The two output constraints are as follows.

O1. EXPAND($C$) will lay out the nodes inside $C$ in, what we have called in Fig. 7, the *reserved space* of $C$. This is the interval $(1, p)$ minus the subinterval spanned by the leader node 1 and $N_1$, and the subinterval spanned by the attachment poin and $N_2$; that is, nodes inside $C$ that are laid out in the interval $(1, 2)$ will go next to node 2, and nodes that are laid out in the border will go next to the free point.

O2. EXPAND will color the edges of $C$ that are not incident to the leader and the edges that lie in the interior of $C$, so that all (new) edges incident to the leader receive color $c_0$.

To lay out an arbitrary planar graph, we first add edges to triangulate it. Let $G$ be the resulting graph, and let $C$ be the cycle that bounds the outer face. We lay out the nodes of $C$, say in clockwise order, pick the first node as the leader, and color its incident edges with an arbitrary color $c_0$. We let the turning point $m$ be the last node, pick arbitrarily colors $c_1, c_2$, and call EXPAND($C$) to lay out the rest of the graph.

We will now describe how EXPAND($C$) works. Assume without loss of generality that the nodes of $C$ are laid out in clockwise order; otherwise, look at the plane from the back (exchange clockwise and counterclockwise in what follows). We call the edge $(1, p)$ that connects the first and last nodes of $C$, the *long* edge of $C$; the rest are the *short* edges of $C$.

In general, $C$ may have some chords in its interior. In Fig. 8a we show a cycle $C = (1, ..., 10)$ with five chords. Figure 8b shows the nodes of $C$ as they are ordered in the layout, and depicts also the edges and chords of $C$. Let $G_c$ be the graph formed by $C$ and its internal chords (see Fig. 8a). The chords partition the interior of $C$ into a number of inner faces of $G_C$, each one bounded by a cycle. The long edge $(1, p)$ of $C$ belongs to exactly one inner face of $G_C$. Let $K$ be the cycle that bounds

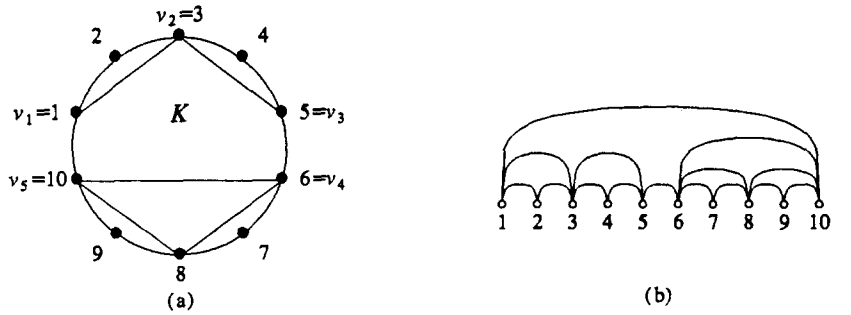


FIGURE 8

this inner face. Let the nodes of $K$ in clockwise order be $v_1 = 1, v_2, ..., v_k = p$. Suppose that a short edge $(v_j, v_{j+1})$ of $K$ is a chord of $C$. Then this edge together with some short edges of $C$ closes a cycle $K_j$; $K_j$ is the cycle $(v_j, v_j + 1, ..., v_{j+1})$. Note that $K$ and these cycles $K_j$ partition the interior of $C$. The algorithm lays out first the interior of $K$, and then the interior of each cycle $K_j$ in turn. In other words, EXPAND can be written in the following recursive form.

*Step* 1. Let $K$ be the cycle bounding the inner face of $G_C$ that contains the long edge $(1, p)$ of $C$; denote the nodes of $K$ as $v_1 = 1, v_2, ..., v_k = p$. Lay out the interior of $K$.

*Step* 2. For each short edge $(v_j, v_{j+1})$ of $K$ that is a chord of $C$ call EXPAND($K_j$), where $K_j$ is the cycle composed of short edges of $C$ and the chord $(v_j, v_{j+1})$.

In the example of Fig. 8, we will lay out the interior of the cycle $K = (1, 3, 5, 6, 10)$, and then expand recursively the cycles $(1, 2, 3)$, $(3, 4, 5)$, $(6, 7, 8, 9, 10)$. Note that the latter cycles occupy disjoint intervals (except for their endpoints) in the layout.

Consider now the subgraph consisting of $K$ and its interior. Level 0 of this subgraph consists of the nodes of $K$. Let $H$ be the subgraph induced by the level 0 (nodes of $K$) and level 1 nodes. This graph $H$ is a two-level graph, bounded by a chordless cycle $K$. All its inner faces are triangles, except possibly for some faces that are interior to the inner graph. The interior of the inner graph did not play any role in the previous section. Thus, we will use the results and the terminology of Section 2. We elaborate Step 1 of EXPAND as follows.

*Step* 1a. Lay out the level 1 nodes of $H$ and color the short edges of $K$ and the binding edges between levels 0 and 1. Prepare for the recursive calls of Steps 1b and 2.

*Step* 1b. Expand recursively the cycles that bound nontrivial blocks of the inner graph of $H$.

In Step 1a, we have to choose a leader for each cycle that is going to be expanded

subsequently and color the edges incident to the leader. Aside from recursive calls, Step 1a does all the work of EXPAND.

Let $m$ be the turning point of $C$ as in property P2. If $m \neq p$, then there is a unique short edge $(v_s, v_{s+1})$ of $K$ that crosses over the border, i.e., with $v_s \leqslant m < m+1 \leqslant v_{s+1}$; $s=1$ is a possibility. If $m=p$, let $v_s$ be the last node of $K$ (i.e., node $v_k=p$) in what follows.

Before we describe Step 1a, let us give a rough sketch of the main problems involved and a preview of how we overcome them. We would like to use an algorithm like that of the previous section for Step 1a. Since the edges exiting the border use colors $c_1$ and $c_2$, all edges that cross over the border (i.e., connect a node to the left to a node to the right of the border) may only use the other two colors. Let $c_3, c_4$ be the other two colors besides $c_1$ and $c_2$. What we have in mind, is to let $c_3$ and $c_4$ play the roles of colors 2 and 3 of the previous section. The main complications are the following. First, the leader's color $c_0$ may be one of $c_3$ and $c_4$. To satisfy the output constraint O2, all back edges to the leader must have color $c_0$. This leaves only one color among $c_3$, $c_4$ available for forward edges from blocks dominated by the leader $v_1$. The solution is to use the nested method for the blocks dominated by $v_1$.

The edge $(v_s, v_{s+1})$ of $K$ that crosses the border must receive color $c_3$ or $c_4$. A problem arises if there are any blocks dominated by $v_s$. Let us say for example that $v_s=m$, $v_{s+1}=m+1$, and that $v_s$ dominates exactly one block $B$ which is adjacent to some nodes right of the border. The leader of $B$ will be laid out (by the algorithm of the last section) before $v_s$, and the nodes assigned to $B$ will be laid out between $v_s$ and $v_{s+1}$. Then, there are three pairwise conflicting edges which may only use the two colors $c_3, c_4$: the edge $(v_s, v_{s+1})$, the edge from the leader of $B$ to its last node, and an edge from $B$ to a node right of the border. The solution is to avoid, if at all possible, placing any blocks between $v_s$ and $v_{s+1}$. In particular, we modify the node layout algorithm so that the impossible situation that we described does not happen: blocks that are placed between $v_s$ and $v_{s+1}$ have edges either only to nodes left of the border or only to nodes right of the border, but not both.

We proceed now with the description of Step 1a. We distinguish three cases.

*Case* 1. $s=1$ or $s \neq 1$ but node $v_s$ does not dominate any blocks. Included here is also the case that there are no nodes in the interior of $K$ ($v_s$ does not dominate any blocks, because there are none), and the case that $m=p$ (the last outer node does not dominate any blocks).

The algorithm in this case is very similar to that of the previous section. As in the previous section, we assume that the first inner node (the node that forms a face with the long edge $(1, p)$) belongs to exactly one block which is seen by node $v_1=1$. Every inner node is assigned to a unique block. The nodes assigned to each block $B$ are placed in the first reserved space of $C$ after the node $v_f$ of $K$ that dominates $B$; i.e., right before $v_f+1$. There is such reserved space after $v_f$ since $f<k$. Thus, for example, the nodes assigned to blocks dominated by $v_1=1$ are placed right before node 2.

For $f \neq 1$, we may choose either the nested method or the consecutive method to order the nodes assigned to the blocks dominated by $v_f$. For $v_1$ we use the nested method if $c_0 \neq c_2$ (i.e., $c_0 \in \{c_3, c_4\}$), and the consecutive method if $c_0 = c_2$.

We come now to the edge coloring part. We color the short edges of $K$ as follows. The edge $(v_1, v_2)$ has already been colored $c_0$. All other edges before the border, i.e., edges $(v_j, v_{j+1})$ with $j < s$ receive color $c_2$. Edges $(v_j, v_{j+1})$ after the border (i.e., $j > s$) receive color $c_1$. If $s \neq 1$, the edge $(v_s, v_{s+1})$ that crosses the border receives color $c_3$.

We also have to choose a leader for each cycle $K_j$ and color its incident edges. If $s \neq 1$ and the edge $(v_s, v_{s+1})$ is a chord of $C$, then one of $v_s$, $v_{s+1}$ is not a border node. We let the leader of $K_s$ be this node. That is, if $v_s < m$ we choose $v_s$ to be the leader of $K_s$, otherwise, we choose $v_{s+1}$. In all other cases (i.e., $j \neq s$ or $j = s = 1$) we let the leader of the cycle $K_j$ be $v_j$ (the first node of the cycle). For all $K_j$ the leader's color is the color of the edge $(v_j, v_{j+1})$.

We come now to the binding edges and the level 1 edges (in $H$). As before, we partition the binding edges into back and forward. Back edges to the leader $v_1$ receive its color $c_0$; all other back edges get color $c_1$. We associate a color $c_3$ or $c_4$ with every block as in the previous section. The only difference is that the choice for the color of the root block may not be arbitrary now. Rule 1 becomes: If $c_0 \in \{c_3, c_4\}$ (i.e., $c_0 \neq c_2$), then the root block gets color $c_0$; otherwise the choice of the color for the root block is arbitrary ($c_3$ or $c_4$). Rule 2 is the same as before. Forward binding edges incident to an inner node $u$ which is assigned to a block with associated color $c_3$ (resp. $c_4$) receive the opposite color $c_4$ (resp. $c_3$).

The leader of a cycle bounding a nontrivial block is the leader of the block. We color the edges of a (trivial or nontrivial) block $B$ incident to its leader as follows. If $B$ is dominated by $v_1$ and $c_0 \in \{c_3, c_4\}$, then we give them color $c_1$; in all other cases (i.e., $B$ not dominated by $v_1$, or $B$ dominated by $v_1$ but $c_0 \notin \{c_3, c_4\}$), we give them the color of the block. This concludes the description of Step 1a for Case 1.

Suppose now that $s \neq 1$ and $v_s$ dominates some blocks. Let $R$ be the highest block (in the tree $T$ of blocks) dominated by $v_s$. Let $u$ be the leader of $R$, and $v_l$ the last node (highest index) of $K$ that sees $R$. Note that since $s \neq 1$, $R$ is not the root block. For an example, suppose that Fig. 1 depicts the graph $H$ for the example of Fig. 8. If the border in Fig. 8 is (3, 4), then $v_s = v_2 = 3$. The highest block dominated by $v_2$ in Fig. 1 is $B_2$. Thus, $R = B_2$, its leader $u$ is node $c$, and $v_l$ is $v_4$. If the border is (5, 6), then $v_s = v_3 = 5$, $R = B_4$, $u = f$, and $v_l = v_4$.

Let $T_R$ be the subtree of $T$ rooted at $R$, and let $T' = T - T_R$ be the tree obtained from $T$ by removing $T_R$. From Lemma 2, all nodes assigned to blocks of $T_R$ (i.e., descendants of $R$) are only adjacent to outer nodes between $v_s$ and $v_l$, while the blocks of $T'$ are not adjacent to any outer node strictly between $v_s$ and $v_l$. See also Fig. 9: The edges $(u, v_s)$, $(u, v_l)$ partition the interior of $K$ into two regions I and II. Each block is contained in one of the two regions. The blocks of $T'$ are in region I, and the blocks of $T_R$ are in region II.

Since $R$ is the highest block dominated by $v_s$, every block of $T'$ is dominated by

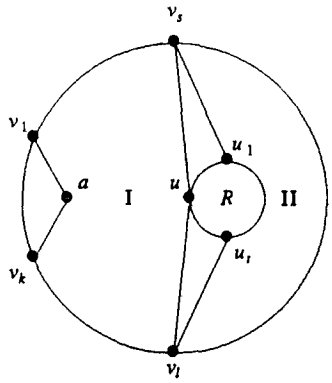


FIGURE 9

some outer node $v_i$ which strictly precedes $v_s$ $(i<s)$ or which succeeds $v_l$ $(i \geqslant l)$. On the other hand, every block of $T_R$ is dominated by a node $v_i$ with $s \leqslant i < l$. Thus, if we used the previous algorithm, all nodes assigned to blocks of $T_R$ would be laid out in the interval $(v_s, v_l)$ while nodes assigned to the blocks of $T'$ would be laid out outside this interval.

The tree $T'$ is processed exactly as in Case 1. The part of the layout outside the interval $(v_s, v_l)$ remains he same. Thus, all nodes assigned to blocks of $T'$ (including the leader $u$ of $R$) are laid out in the intervals $(v_1, v_s)$, $(v_l, v_k)$. Also, all edges with both endpoints outside the interval $(v_s, v_l)$ are colored as in Case 1. The nodes assigned to blocks of $T_R$ will be laid out in the interval $(v_s, v_l)$ in a different way. We distinguish two cases, depending on whether $l = s+1$ or $l > s+1$.

*Case* 2. $l = s+1$. The blocks of $T_R$ are adjacent to at least two outer nodes. Since they can only be adjacent to outer nodes between $v_s$ and $v_l = v_{s+1}$, we conclude that they are all adjacent to $v_s$ and $v_{s+1}$. Thus, they are all dominated by $v_s$, and they form a directed path. We use the nested method to lay out the nodes assigned to these blocks. The nodes are placed right next to $v_s$ or $v_{s+1}$ according to one of the following three subcases:

(2a) $v_s = m$ and $v_{s+1} = m+1$; i.e., $(v_s, v_{s+1})$ is the border of $C$.

If (2a) does not hold, then by the planarity of the graph, either

(2b) there is no edge from $v_s$ to another node $j \neq v_{s+1}$ of $C$ that crosses the border $(j \geqslant m+1)$, or

(2c) there is no edge from $v_{s+1}$ to another node $j \neq v_s$ of $C$ that crosses the border $(j \leqslant m)$.

Note that in Case 2b we have $v_s \neq m$, and in Case 2c, $v_{s+1} \neq m+1$. In Case 2a place the nodes assigned to the blocks of $T_R$ next to the free point of the border; in Case 2b they are placed right after $v_s$, and in Case 2c they are placed right before $v_{s+1}$.

From the processing of $T'$, a color has been given to the parent of $R$ in the tree

$T$. Associate the opposite color ($c_3$ or $c_4$) with all the blocks of $T_R$ (as in Rule 2 of the coloring of blocks). Binding edges from the inner nodes assigned to the blocks of $T_R$ to the furthest of $v_s$, $v_{s+1}$ (e.g., the attachment point in Case 2a) receive color $c_3$ or $c_4$ opposite to that of $R$, and binding edges to the closer one receive color $c_1$. For each block of $T_R$, the leader of the cycle bounding the block is the leader of the block, and edges of the block incident to the leader get the color of the block.

The edge $(v_s, v_{s+1})$ receives color $c_3$ or $c_4$ opposite to that assigned to block $R$. If $(v_s, v_{s+1})$ is a chord of $C$, the leader of the corresponding cycle $K_s$ is $v_s$ in Case 2b ($v_{s+1}$ in Case 2c) with color $c_2$ (resp. $c_1$).

*Case* 3. $l > s+1$. To lay out region II, ignore region I (remove from the graph the nodes inside region I). Imagine that there is an edge $(v_s, v_l)$ drawn on the plane in region I, and that we want to expand the cycle $C' = (v_s, v_s+1, ..., v_l)$ which is formed by this edge and edges of $C$. Regard $v_l$ (the last node of $C'$) as the leader of $C'$ and let $c_1$ be the leader's color. By planarity, node $v_l$ may only be adjacent to other nodes $i$ of $C'$ between $v_{l-1}$ and $v_l$; i.e., $m+1 \leqslant v_{s+1} \leqslant v_{l-1} \leqslant i < v_l$. Give color $c_1$ to these edges $(v_l, i)$. We could color the fictitious edge $(v_s, v_l)$ with the color assigned to the parent of $R$ ($c_3$ or $c_4$), but this is not really necessary since this edge does not exist. It is easy to see that $C'$ satisfies the properties P1–P6. The cycle $C'$ has the same border and free point as $C$; only, its turning point is $m+1$ because its leader is $v_l$ (the last node). The first color of $C'$ is $c_2$ and the second color is $c_1$.

To expand $C'$ we will first add to it the chords. Let $K'$ be the boundary of the (inner) face that contains the long edge $(v_l, v_s)$ of $C'$. The cycle $K'$ is $(v_s, v_{s+1}, ..., v_l)$ and its interior is region II. The first inner node for $K'$ is $u$ which belongs to exactly one block (of region II), namely $R$. Note that the leader $v_l$ sees the block $R$. The tree of blocks for the interior of $K'$ is the subtree $T_R$ of $T$ rooted at $R$.

The edge of $K'$ that crosses the border is $(v_s, v_{s+1})$, which is the last edge of the cycle as seen from the leader's $v_l$ viewpoint (i.e., from right to left). Thus, if we apply the algorithm to $C'$ we will arrive at Case 1 or Case 2, which we have already described. The only differences are that (i) we omit the first inner node $u$ (since it has been already laid out) and (ii) in Rule 1, we give to the root block $R$ color opposite to that of its parent in $T$. The node layout, coloring of edges, and choice of leaders and their colors for the various cycles follow from our description for Cases 1 and 2; remember only that now we proceed right-to-left since the leader is the last node $v_l$. Thus, for example, we take now the dominator of a block $B$ in region II (a block of $T_R$) to be the *last* node $v_i$ of $K$ (highest $i$) that sees $B$; for $i > s+1$ the nodes assigned to $B$ are inserted right *before* $v_i$. If the last node of $K$ that sees a block $B$ of region II is $v_{s+1}$, then $B$ is seen only by nodes $v_s$ and $v_{s+1}$; all such blocks are placed next to $v_s$ or $v_{s+1}$ as explained in Case 2.

This concludes the description of Step 1a, and also the description of the algorithm.

EXAMPLE. Suppose that Fig. 1 depicts the graph $H$ for the cycle of Fig. 8. Let node 4 be the turning point. Then, $v_s = v_2 = 3$. Node $v_2$ dominates some blocks. The

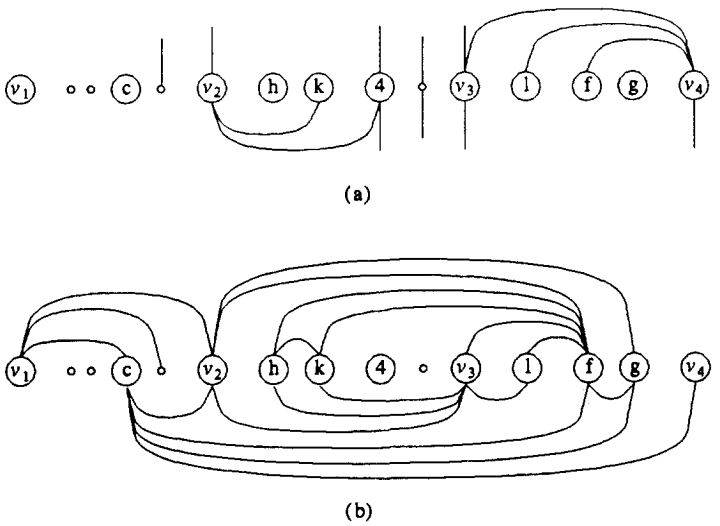


FIGURE 10

highest such block $R$ is $B_2$. The leader of $B_2$ is node $c$, and the last node that sees $B_2$ is $v_4 = 6$. The tree rooted at $B_2$ contains the blocks $B_2$, $B_3$, $B_4$. The rest of the blocks $B_1$, $B_5$ are in $T'$. These blocks are laid out as in Case 1. The nodes of $B_1$, $\{a, b, c, d, e\}$, are placed before node 2, and the node assigned to $B_5$, node $m$, is placed after $v_4$ (node 6).

Since $l = 4 \neq s + 1 = 3$, Case 3 applies. We take the dominator of each remaining block to be the last node that sees the block. Thus, the dominator of $B_2$ and $B_4$ is $v_4$, and the dominator of $B_3$ is $v_3$. Since the leader's color for $C'$ is the same as its second color, we lay out the nodes assigned to $B_2$ and $B_4$ right before $v_4$ using the consecutive method. The nodes assigned to $B_3$ are to be placed between $v_2$ and $v_3$. Since $v_3$ (node 5) has an edge that crosses the border (namely, the edge (5, 4)), we lay out $B_3$ next to $v_2$ (node 3).

In Fig. 10 we show the layout. For clarity, we have omitted most of the layout outside the interval $[v_2, v_4]$; this portion is very similar to Fig. 5. In the figure we show also the coloring of the edges. The edges of Fig. 10a have colors $c_1$ (upper halfplane) and $c_2$ (lower halfplane), and those of Fig. 10b have colors $c_3$ (upper halfplane) and $c_4$ (lower halfplane). We assume that the leader's color is $c_3$. Then the root block $B_1$ gets color $c_3$, its child $B_2$ color $c_4$, and $B_3$ and $B_4$ get color $c_3$.

## 4. PROOF OF CORRECTNESS

We shall show that EXPAND works correctly: if properties P1–P6 hold at the beginning, then upon termination of the call the layout is legal and satisfies the output constraints O1 and O2. We shall use induction,. Thus, we assume that the

expansions of the cycles in Steps 1b and 2 satisfy the claim, and will show it for $C$. To use the induction hypothesis, we will have to argue also that, when the expansion of a cycle is initiated in Step 1b or Step 2, the layout at that time satisfies the properties P1–P6 for that cycle.

We will refer to the cycles that are expanded in Steps 1b and 2 as the *new* cycles. We note at first that $K$ and the cycles $K_j$ partition the interior of $C$. Thus, every node inside $C$ (in the planar embedding) is either inside $K$ or inside one of the $K_j$'s. A node inside $K$ is either at level 1 of the subgraph induced by $K$ and its interior, in which case it is laid out in Step 1a, or it is at a higher level, in which case it is in the interior of the inner graph, i.e., in the interior of a level 1 cycle. Also note that the new cycles have disjoint interiors. Thus, every node inside $C$ is either laid out in Step 1a, or is in the interior of exactly one new cycle and thus is laid out when this cycle is expanded. A similar observation holds for the edges. Note that the only shared edges among the new cycles are short edges of $K$ which are colored in Step 1a.

We also observe that EXPAND is only called on cycles with no external chords. In fact, it is easy to show inductively that all the cycles that are expanded consist of nodes at the same level in the graph $G$. This is obviously true for the initial cycle, the boundary of the outer face of $G$. Also, if $C$ has nodes at level $i$ of $G$, then the cycles $K_j$ have also nodes at level $i$, while the new cycles of Step 1b have nodes at level $i+1$ of $G$. By the definition of levels, the subgraph induced by nodes at the same level is outerplanar (all the nodes are on the boundary). Thus if all the nodes of a cycle are at the same level of $G$, the cycle cannot have any external chords.

We will refer to the edges that have color $c_i$ as the $c_i$ edges, and to the blocks that are dominated by a node $v_i$ as the $v_i$ blocks. We call the edges that were already colored when EXPAND($C$) started, the *previous* edges; by *current* edges, we mean the edges that are colored in Step 1a. We must show that a current edge does not conflict with a previous or current edge of the same color. We shall show the correctness of EXPAND by examining each one of the three cases of the algorithm in turn. Given the analysis of Section 2, the proof is straightforward, but tedious.

*Case* 1. Node $v_s$ is the leader $v_1 = 1$, or $v_s$ does not dominate any blocks. Then the node layout part of the algorithm is identical to that of Section 2, except that we choose a particular ordering method for the $v_1$ blocks. The edge coloring part is almost the same, where colors $c_3$ and $c_4$ are used in place of colors 2 and 3 there, and colors $c_1$ and $c_2$ are used in place of color 1 there. In particular, with this correspondence of colors, the binding and level 1 edges are colored exactly as before if $c_0 = c_2$; if $c_0 \neq c_2$, we just switched the color of the edges that belong to $v_1$ blocks with the color of the back edges to the leader $v_1$. We shall take each color in turn, list the current edges with this color, and verify that they do not conflict with each other, or with any previous edges of the same color.

*Color* $c_1$. The current edges are (1) back edges to nodes $v_j$ with $j > 1$, (2) the edge $(v_j, v_{j+1})$ for each $j \geq s+1$, and in case this edge is a chord, the edges connecting the leader $v_j$ of $K_j$ to other nodes of $K_j$ (i.e., the edges or chords $(v_j, i)$ of $C$,

where $m+1 \leqslant v_{s+1} \leqslant v_j$ and $v_j < i \leqslant v_{j+1}$), and (3) if $c_0 \in \{c_3, c_4\}$ the edges of the $v_1$ blocks. The nodes of $v_1$ blocks are placed between $N_1$ and node 2, and the nodes assigned to $v_j$ blocks with $j > 1$ are placed right after $v_j$. (Recall that, if $s > 1$, then $v_s$ does not dominate any blocks.) It is clear that no two current edges conflict. Also, the current edges of (1) and (3) do not conflict with any previous edges at all. By P5, the edges of (2) conflict only with previous edges of color $c_2$.

*Color* $c_2$. The current edges are (1) if $c_0 = c_2$ the back edges to $v_1$, and (2) the edges $(v_j, i)$ where $v_2 \leqslant v_j < v_s$ and $v_j < i \leqslant v_{j+1}$. Clearly, there is no conflict between any two current edges. By P5, the current edges of (2) conflict only with previous edges of color $c_1$. We shall argue that the back edges to the leader do not conflict with any previous $c_0$ edges, without using the assumption that $c_0 = c_2$. The back edges to the leader conflict only with previous edges that go from $N_1$ to node 2 or to a node outside the interval $[1, 2]$. By property P6, there is no previous edge from node 2 to $N_1$ with color $c_0$. Also, since the previous layout $L$ was legal and the edge $(1, 2)$ was colored $c_0$, no previous edge which is incident to $N_1$ and exits the interval $[1, 2]$ has color $c_0$.

*Color* $c_3$. The current edges are (1) if $c_0 = c_3$ the back edges to the leader, (2) some forward binding and level 1 edges, (3) if $s \neq 1$ the edge $(v_s, v_{s+1})$ and possibly some more edges from $v_s$ or $v_{s+1}$ to nodes of $C$ between $v_s$ and $v_{s+1}$. If $s \neq 1$, then $v_s$ does not dominate any blocks, and thus no nodes are inserted between $v_s$ and $v_{s+1}$. Therefore the edges of (3) do not conflict with any other current edges; by P5, they only conflict with previous edges that have color $c_1$ or $c_2$. The back edges to the leader can only conflict with forward and level 1 edges which are incident to nodes assigned to the $v_1$ blocks. We shall argue that if $c_0 = c_3$, then the latter edges do not have color $c_3$. If $c_0 = c_3$, then the edges of the $v_1$ blocks have color $c_1$ and the forward edges incident to nodes assigned to the $v_1$ blocks have color $c_4$ (opposite to $c_0$). Also, if a level 1 edge is incident to a node of a $v_1$ block but belongs to another block $B$ not dominated by $v_1$, then $B$ is a child of a $v_1$ block with a different dominator; thus, $B$ is assigned color $c_4$ (opposite to $c_0$) and also the edge of $B$ receives color $c_4$. It follows from the analysis of Section 2, that there is no conflict between any two current $c_3$ edges.

The forward binding and level 1 edges that are not incident to nodes assigned to $v_1$ blocks conflict only with previous edges that have color $c_1$ or $c_2$ (by P5). If there are more $c_3$ edges of type (2), then as we just observed, $c_0 \neq c_3$, and these edges conflict only with previous edges that have color $c_0$, $c_1$ or $c_2$ (by P4 and P5). The back edges to the leader do not conflict with any previous $c_0$ edges, as we argued for $c_2$.

The proof for the $c_4$ edges is similar to $c_3$. Thus, we have shown that the layout after Step 1a is legal. We shall argue now that the layout after Step 1a satisfies the properties P1–P6 for all the new cycles. As we examine each new cycle, we will also observe the effect of the expansions of other cycles to make sure that they do not violate the properties. Note that we do not have to show that the expansion of

other cycles preserves the legality of the layout; this was done already for Step 1a, and from then on it is taken care of by the induction hypothesis.

A cycle $K_j$ has the form $v_j, v_j+1, ..., v_{j+1}$. For all $j$ it is clear that P1 is satisfied, and that it cannot be violated after other new cycles are expanded. We also observe that the intermediate nodes of $K_j$ (i.e., all the nodes except $v_j$ and $v_{j+1}$) are not adjacent to any nodes which lie (in the planar embedding) inside $K$ or inside another cycle $K_i$, because of the planarity of the graph. Therefore, apart from the edges incident to the leader of $K_j$, neither Step 1a nor the expansion of any other new cycle will add any edges to the intermediate nodes of $K_j$. We consider the cases $j=1$, $1<j<s$, $j=s\neq 1$, $s<j<k$. In each case, we will state the turning point and the first and second colors. The verification of the properties is routine.

*Cycle* $K_1$. The leader is the first node and its color is $c_0$. Step 1a inserted nodes only between $N_1$ and node 2. If $s=1$ (i.e., $m+1\leqslant v_2$), then $K_1$ inherits from $C$ the turning point $m$, the free point, the first color $c_1$ and the second color $c_2$. If $s\neq 1$, then the turning point of $K_1$ is $v_2$ (the last node), and the first color is $c_1$. We may take $c_2$ (or any other color besides $c_1$) as the second color. The expansion of the cycles bounding nontrivial $v_1$ blocks will only add some more nodes between $N_1$ and node 2. (We use O1 here.) The expansion of other cycles will have no effect on $K_1$.

*Cycle* $K_j$ *with* $1<j<s$. The leader is $v_j$, the first node of this cycle, with color $c_2$. Step 1a inserted nodes only next to the leader. We let the turning point be $v_{j+1}$ (the last node of $K_j$) and the first color $c_1$; we may take $c_2$ as the second color. The expansion of cycles bounding nontrivial $v_j$ blocks may introduce some more nodes between $v_j$ (the leader of $K_j$) and $v_j+1$; expansion of other cycles will have no effect. The case $s<j$ is analogous.

*Cycle* $K_s$, with $s\neq 1$. Since $v_s$ does not dominate any blocks, no new nodes have been inserted between $v_s$ and $v_{s+1}$. This implies in particular that the expansion of other cycles has no effect on $K_s$. Besides the nodes of $K_s$, the only other nodes in the interval $(v_s, v_{s+1})$ are the nodes of $N_2$. The cycle $K_s$ inherits the border and the free point from $C$. Thus, if its leader is $v_s$ (in which case $v_s<m$), then the turning point is $m$, the first color is $c_1$ and the second color is $c_2$. If the leader of $K_s$ is $v_{s+1}$ (in which case $m+1<v_{s+1}$), then the turning point is $m+1$, the first color is $c_2$ and the second color is $c_1$. Note that in either case there are no nodes between the leader and the next node of $K_s$. Why did we not take $v_s$ as the leader even if it was the turning point $m$? If the attachment point of $C$ was $m+1$ (the second node of the cycle $K_s$), then P5 and P6 could be violated because the color of the edges between $N_2$ and $m+1$ is not restricted.

Consider now a nontrivial block $B$ and let $\bar{B}$ be the cycle that bounds $B$. Let $v_f$ be the dominator of $B$, $u$ its leader, $u_1, ..., u_t$ the rest of its nodes in clockwise order, and $u_q$ the node of highest index $q$ which is adjacent to $v_f$. The nodes of $\bar{B}$ are laid out in counterclockwise order $u, u_t, ..., u_q, u_{q-1}, ..., u_1$. If $B$ is laid out by the consecutive method, then all its nodes except for the leader are consecutive. If $B$ is laid

out by the nested method, then there may be more nodes between $u_q$ and $u_{q-1}$; namely, the nodes that are assigned to blocks which are descendants of $B$ and are dominated by the same node $v_f$. As we have already shown (Lemma 4), $u_q$ is the leader of the highest such block (the child of $B$). Thus, the nodes that are laid out between $u_q$ and $u_{q-1}$ are not adjacent to any nodes of $\bar{B}$ except $u_q$. Properties P2 and P3 are satisfied with $u_q$ as the turning point and the attachment point. The expansion of other cycles may introduce some more nodes between $u$ and $u_t$, and between $u_q$ and $u_{q-1}$ (if $B$ was laid out by the nested method). P2 and P3 are not affected.

P4 is clearly satisfied: the edges of $B$ incident to the leader are all colored with the color of the block, or with color $c_1$ if $B$ is dominated by $v_1$ and $c_0 \neq c_2$. The expansion of other cycles has no effect on P4, since $\bar{B}$ does not share edges with any other cycle.

We will show now that P5 is satisfied. From Lemma 4, the nodes $u_{q-1}, ..., u_1$ are only incident to back edges and to edges of block $B$. Apart from the edges going to the leader $u$ of $B$, we have only colored the back edges incident to $u_{q-1}, ..., u_1$, and they all have the same color ($c_1$ or $c_0$ if $B$ is dominated by $v_1$). This is the second color of $\bar{B}$; note that it is different than the leader's color. Until $\bar{B}$ gets expanded, no other edges incident to $u_{q-1}, ..., u_1$ will be colored. Thus, clause (a) of P5 holds.

The nodes $u_t, ..., u_{q+1}$ (if $q < t$) are not incident to any back edges. Apart from the edges to $u$, all other colored edges incident to these nodes are forward edges or level 1 edges which belong to blocks that are children of $B$ and are not dominated by $v_f$. (The only possible child of $B$ that is dominated by $v_f$ has $u_q$ as its leader.) Thus, all colored edges incident to $u_t, ..., u_{q+1}$ which do not go to the leader $u$, have the same color, namely $c_3$ or $c_4$, opposite to the color of $B$. This is the first color of $\bar{B}$ and is clearly different than the leader's color. Some more edges may be added to the nodes $u_t, ..., u_{q+1}$ during the expansion of other cycles. These are edges interior to other blocks which have one of these nodes as their leader. By O2, all these new edges are colored with the first color of $\bar{B}$. This establishes clause (b) of P5.

If $B$ is laid out by the consecutive method, then the edges that exit the interval $[u_q, u_{q-1}]$ and do not go to the leader of $\bar{B}$ are: the back edges incident to $u_q$ and $u_{q-1}$ (second color), the forward edges from $u_q$ (first color) and the level 1 edges incident to $u_q$ which belong to children of $B$ (first color). If the nested method is used to lay out the blocks dominated by $v_f$, then we associate the same color with all the $v_f$ blocks. It follows easily that all edges exiting the interval $[u_q, u_{q-1}]$ have either the first color (forward binding and level 1 edges) or the second color (back edges). The same is true of all new such edges that are colored during the expansion of other cycles. Thus, P5 is satisfied. P6 follows from P5, because the leader's color is different than the second color.

The proof that O1 and O2 are satisfied at the end of EXPAND($C$) is easy. Clearly, Step 1a inserts nodes only in the reserved space of $C$, and the reserved space of all the new cycles is contained in the space of $C$. O1 follows. All edges incident to the leader which are colored in Step 1a receive color $c_0$. The rest of the edges incident to the leader are inside $K_1$. Since node 1 is the leader of $K_1$ with

color $c_0$, the rest of the edges incident to the leader will also be colored $c_0$ (by the induction hypothesis). O2 follows.

*Case* 2. If $v_s$ and $v_{s+1}$ are consecutive nodes of $C$, then we must have $v_s = m$ and $v_{s+1} = m+1$, because $v_s \leqslant m < m+1 \leqslant v_{s+1}$. Thus, 2a applies in this case. If $v_s$ and $v_{s+1}$ are not consecutive then either 2b or 2c applies.

Note that in Case 2b, we must have $v_s < m$ because node $m$ is adjacent to $m+1$. Similarly, in Case 2c, we have $v_{s+1} > m+1$. Thus, in all three cases, the inner nodes are inserted in the reserved space of $C$.

The correctness for the portion of the layout that is outside the interval $[v_s, v_{s+1}]$ follows as in Case 1. We only have to pay attention to the interval $[v_s, v_{s+1}]$. The only current edges entering this interval are the edges from node $u$ (the leader of $R$) to the other nodes of $R$. All these edges have the same color, namely the color of $R$ ($c_3$ or $c_4$), which is opposite to the color of the edge $(v_s, v_{s+1})$. It is straightforward to verify that the coloring is legal, and check that P1–P6 hold for the cycles bounding the nontrivial $v_s$ blocks. We shall only check the properties for the cycle $K_s$.

Suppose that Case 2c applies; Case 2b is analogous. Then the leader is $v_{s+1}$. Note that we can color legally the edges from $v_{s+1}$ to other nodes of $K_s$ with $c_1$ because all these nodes are greater than or equal to $m+1$. P1 and P4 clearly hold. Since $v_{s+1} > m+1$, initially there are no nodes between $v_{s+1}$ and $v_{s+1}-1$ (the second node of $K_s$ as seen from the leader's viewpoint). During Step 1a, all new nodes inserted in the interval $(v_s, v_{s+1})$ are next to the leader $v_{s+1}$. None of them is adjacent to $v_{s+1}-1$; hence P6 holds. The cycle $K_s$ inherits the border and the free point of $C$. Thus, the turning point of $K_s$ is $m+1$, the first color is $c_2$ and the second color is $c_1$. The portion of the layout between $v_{s+1}-1$ (the second node of $K_s$) and $v_s$ (the last node of the cycle) does not change during Step 1a, nor does it change during the expansion of other cycles. Thus, P2, P3, and P5 hold.

*Case* 3. The correctness of the portion of the layout that is outside the interval $[v_s, v_l]$ follows as in Case 1. Thus, we may ignore this portion. If node $u$ was not outside the interval $[v_s, v_l]$ but was laid out right before $v_l$ (as it would if $C'$ was expanded normally), then the correctness for the portion of the layout between $v_s$ and $v_l$ would also follow from Cases 1 and 2. The only anomaly is the position of $u$.

We only have to pay attention to the edges incident to $u$ (verify they are legally colored), and to the cycle $\bar{R}$ that bounds the block $R$ if $R$ is nontrivial (verify that $\bar{R}$ satisfies P1–P6).

All edges from $u$ entering the interval $[v_s, v_l]$ go to other nodes of $R$, and they all have the color of the block $R$. As far as the legality of the coloring is concerned, the only difference caused by the position of $u$, is that the edges from $u$ to nodes of $R$ conflict with some more edges indicent to $v_l$ (some more back edges, and the edges from $v_l$ to $v_{l-1}$ and other nodes of $K_{l-1}$); all these edges have color $c_1$.

Now we verify the properties P1–P6 for the cycle $\bar{R}$ that bounds the block $R$. The cycle $C'$ is laid out in counterclockwise order from its leader's viewpoint (right-to-left). Therefore, each cycle bounding a (nontrivial) block is laid out in clockwise

order from right to left starting with the leader. Or, in other words, an inner cycle is laid out in counterclockwise order from left to right ending with the leader. The only exception is block $R$ whose leader $u$ is laid out first. Let $u_1, ..., u_t$ be the nodes assigned to $R$ in clockwise order. The nodes of $R$ are laid out in the order $u, u_t, ..., u_1$. Thus, P1 is satisfied. Since the leader's color for $C'$ is the same as the second color, the blocks dominated by $v_l$ are laid out using the consecutive method. Therefore, all of the nodes of $R$ are consecutive in the layout, except for the leader. Let $u_q$ be the node of $R$ with least index $q$ which is adjacent to $v_l$; it is possible that $q = 1$ or $q = t$. The turning point of $R$ is $u_q$. Clearly, P2, P3, and P4 are satisfied.

Apart from the edges of block $R$, nodes $u_t, ..., u_{q+1}$ (if $q \leqslant t$) are only incident to back edges to $v_l$. These edges have all color $c_1$, which is the first color of $\bar{R}$. Nodes $u_{q-1}, ..., u_1$ (if $q > 1$) are incident to (1) forward edges (in the right-to-left direction), and (2) level 1 edges that belong to blocks which are children of $R$. Since the $v_l$ blocks are laid out by the consecutive method, all these edges have the same color, $c_3$ or $c_4$, opposite to the color of $R$. This is the second color. Edges incident to $u_q$ have the first or the second color. Thus, P5 holds, and P6 follows because the leader's color is different than the second color. We remark here that using the nested method for the $v_l$ blocks could lead to violation of P5 and P6. (This is actually the only reason that we chose the consecutive method for the leader's blocks in Case 1 when the leader's color coincides with the second color.) For, suppose that $u_q = u_t$, and that $v_l$ dominates another block $B$. Then the leader of $B$ is $u_q = u_t$ (Lemma 4), and the nodes of $B$ are laid out to the left of $u_t$; i.e., between $u$ and $u_t$. From the coloring rules, we would assign to $B$ the same color as $R$, and we would give this color to all the edges from $u_t$ to other nodes of $B$. This would violate P5 and P6.

Clearly, the expansion of other cycles does not cause the insertion of any nodes between $u_t$ and $u_1$. Some nodes may be inserted between the leader $u$ and $u_t$. Some edges may also be added to the nodes $u_q, ..., u_1$ from nodes that are inside blocks, which have one of these nodes as their leader. All these new edges will get the second color by (O2). It follows that P1–P6 are not affected by the expansion of other cycles.

THEOREM 2. *The algorithm embeds any planar graph in four pages.*

## 5. TIME COMPLEXITY

We assume that the planar graph $G$ is represented as follows. For each node $u$ we have a doubly linked circular list of the edges incident to $u$ in the order in which they appear around $u$ in the planar embedding. That is, for each edge in the list, the "next" pointer points to the counterclockwise next edge, and the "previous" pointer points to the previous edge (the clockwise next). We assume also that the two occurrences of an edge $(u, v)$, in the list of $u$ and in the list of $v$, are linked to each other, so that we can jump from one list to the other in constant time. This

representation of a planar graph can be constructed by planarity algorithms, such as the Hopcroft–Tarjan algorithm [HT], in linear time.

Recall how we can trace the boundaries of the faces with this representation: to trace the boundary of a face containing the edge $(u_0, u_1)$, traverse the edge from $u_0$ to $u_1$; find the next edge on $u_1$'s list (i.e., the edge which follows counterclockwise the edge $(u_1, u_0)$), say edge $(u_1, u_2)$, and traverse it in the direction $u_1 \to u_2$. Similarly, traverse the next edge from $u_2$, and so on, until finally the edge $(u_0, u_1)$ is traversed again in the direction $u_0 \to u_1$. This gives a traversal of the boundary of one face containing the edge $(u_0, u_1)$.[2] The traversal is in the clockwise or counterclockwise direction according as the face is an inner face or the outer face. The other face containing the edge $(u_0, u_1)$ (if it exists) can be obtained similarly by starting the traversal in the direction $u_1 \to u_0$.

The representation of a planar graph by the circular adjacency lists does not specify a unique planar embedding. It only fixes the faces of the embedding. We may choose an arbitrary face as the outer face. Once an outer face has been chosen, the planar embedding is uniquely determined.

We assume that our graph $G$ is triangulated. If it is not, then it can be easily augmented in linear time to a trangulated graph.

We represent a (possibly partial) layout of the nodes by a doubly linked linear list. The representation of the coloring of the edges is not important. For example, we may have for each color a list of the edges with this color. We have an array indexed by the nodes; the entry for each node points to the node's location in the list representing the node layout. If the node has not been laid out yet, the pointer is *nil*. Thus, we can tell in constant time whether a node $u$ has been laid out, and we can also insert other nodes immediately before or right after $u$ in the layout.

Note that, except for the nodes of the outer face, when a new node is laid out in Step 1a of EXPAND($C$), it is adjacent to a node of $C$, i.e., to a node of the preceding level which has already been laid out. For reasons that we will explain later, we have an additional array indexed by the nodes. This array contains for each node $u$ that has been laid out, the color of a binding edge (any one) from $u$ to a node at the preceding level.

The final data structure is a stack $S$ that contains the cycles waiting to be expanded. Actually, it is not really important that $S$ be a stack; it can be shown that the final layout does not depend on the order in which we expand the remaining cycles. The stack order corresponds to the recursive structure of EXPAND. The record in the stack $S$ for a remaining cycle $C$ does not contain a listing of the nodes of $C$; this would cause much copying, leading potentially to quadratic time. The record for $C$ contains the following information: the (name of the) leader of $C$, the second node of $C$ (the one next to the leader); the turning point; the other border node; a bit indicating which of the two border nodes is the free point; the long edge and a pointer to its occurrence in the leader's list; the leader's color $c_0$, the first color $c_1$ and the second color $c_2$; a bit indicating whether the leader is the first or the last

[2] We assume $G$ is connected. If $G$ is not connected, then a face may have a disconnected boundary. This traversal would then trace only one component of the boundary.

node of $C$ in the layout, and a bit indicating whether $C$ is laid out in clockwise or counterclockwise order.

For the purposes of the correctness proof, it was more convenient to assume that the edges from the leader to the other nodes of $C$ are all colored before EXPAND($C$) is called. For implementation purposes, it is more convenient to assume that only the long edge is colored; the coloring of the rest of the edges is the responsibility of EXPAND($C$). The only changes in Step 1a are: (1) we give color $c_0$ to the edge $(v_1, v_2)$, and (2) we only have to color the long edge of each cyce $K_j$ and not the rest of the edges incident to the leader of $K_j$.

Initially, we choose arbitrarily a face as the outer face. We lay out the three nodes of the cycle bounding the outer face and insert the corresponding record into the stack $S$.

In the general step, we pop from the stack the top record, say for cycle $C$. We use the notation of the algorithm. Suppose, without loss of generality, that the leader of $C$ is the first node 1, and that $C$ is laid out in clockwise order. Starting from the long edge in the leader's adjacency list, keep taking the next edge (in counterclockwise order around the leader) until an edge is found to a node which has been already laid out. (It will help the reader to look at Fig. 1 during this discussion.) Note that this node that has been already laid out, is precisely $v_2$, the second node of the cycle $K$. (Recall how we can find out the faces of a planar graph from its representation by the circular adjacency lists; since no nodes inside $C$ have been laid out yet, we are tracing the boundary of that inner face of $G_c$ which contains the long edge.) Note also that the edges incident to the leader that we scanned, are precisely the binding edges from the leader $v_1$ to inner nodes inside $K$. We move now to the adjacency list of $v_2$. Starting from the edge $(v_2, v_1)$, we keep taking the next edge, until an edge is found to a node which has been already laid out; this is node $v_3$. Continuing in this way we can find the cycle $K$.

We can also form a sequence $\beta = b_1 b_2 \cdots b_r$ of the binding edges in the order they are scanned; first the edges of $v_1$, then of $v_2$, etc. We observe that any two consecutive edges in the sequence $\beta$ (where we also let $b_{r+1} = b_1$) belong to a common face of the graph $G$. Since $G$ is triangulated, this implies that: (1) if $b_j$ and $b_{j+1}$ are incident to the same outer node $v_i$, say $b_j = (v_i, x_j)$ and $b_{j+1} = (v_i, x_{j+1})$, then $(x_j, x_{j+1})$ is an edge which in fact lies on the boundary of the inner graph; (2) if $b_j$ and $b_{j+1}$ are incident to different (and therefore consecutive) outer nodes, say to nodes $v_i$ and $v_{i+1}$, then $b_j$ and $b_{j+1}$ are incident to the same inner node, say node $u$, and $(u, v_i, v_{i+1})$ is a face.

Let $\alpha$ be the sequence of edges defined as follows:

$\alpha \leftarrow$ empty sequence;
*for* $j = 1$ to $r$ *do*
    *if* the inner node, say $x_j$, of $b_j$ is different
    than the inner node, say $x_{j+1}$, of $b_{j+1}$ *then*
    append the edge $(x_j, x_{j+1})$ to $\alpha$

We regard the edges of $\alpha$ as being directed from $x_j$ to $x_{j+1}$. As we observed,

either $x_j = x_{j+1}$, or $(x_j, x_{j+1})$ is an edge on the boundary of the inner graph $I$. Thus, the sequence $\alpha$ describes a closed walk on the boundary of the inner graph. The first (and last) node is the inner node of the edge $b_1$; this is precisely the first inner node $a$. Since every inner node is incident to a binding edge, $\alpha$ visits all the inner nodes. It is easy to see that $\alpha$ actually traces the boundary of the inner graph in clockwise order. (If we take two consecutive edges of $\alpha$, then there are only binding edges between them.)

From the traversal $\alpha$, we can easily extract the boundaries of the blocks of $I$ as follows: Initialize an empty stack. Take the edges of $\alpha$ in sequence. If the next edge of $\alpha$ say edge $(x_j, x_{j+1})$, enters a new node $x_{j+1}$ that has not been previously visited, then push the edge onto the stack; otherwise, pop the stack up to the previous ocurrence of $x_{j+1}$. The popped edges trace the boundary of a block $B$ of the inner graph. The leader of $B$ is node $x_{j+1}$, the first node of $B$ that was reached in the traversal which started at the first inner node $a$.

We may construct the traversal $\alpha$ of the boundary of the inner graph at the same time that we scan the adjancency lists of the outer nodes and form the sequence $\beta$ of binding edges. Or we may construct and process $\alpha$ in a separate pass: From the long edge in the leader's adjacency list, we take the binding edge $b_1$ to the first inner node $a$. We move to $a$'s list. From then on, we keep scanning the lists of the inner nodes in clockwise order, skipping over the binding edges (edges that go to nodes which are already laid out). While processing the traversal $\alpha$, we find the blocks. For each block $B$, we record its nodes, the leader, the position in the leader's list of the first and last edge of the block $B$ in $\alpha$ (one of these will be the long edge for $B$), and the first and last outer nodes that see the block. These are easy to find. For example, if the first edge of $B$ is the edge $(x_j, x_{j+1})$ connecting the inner nodes of the binding edges $b_j$ and $b_{j+1}$, then the first outer node that sees $B$ is the common outer node of $b_j$ and $b_{j+1}$.

If the first inner node $a$ belongs to more than one block, or belongs to one block that is not seen by the leader $v_1$ of $C$, then we add a fictitious first inner node, as explained in Section 2. We can easily construct the tree $T$ of blocks.

To perform Step 1a, we have to find $v_s$ first, in order to determine which case applies. If the cycle $K$ contains one of the border nodes, then it is easy to tell which node is $v_s$; this is the case, for example, if $C$ has no chords and $C = K$, or if the turning point is the last node of the cycle. What if $K$ does not contain any border nodes? Then, from property P5, every intermediate node of $K$ is incident to edges of only one color: the first color if the node is left of the border, and the second color if it is right of the border. We consult the array which gives for every node the color of an incident edge, to determine which side of the border each node of $K$ lies in. This way, we can find $v_s$ and determine if it dominates any blocks. If Case 1 applies, then we can insert the inner nodes into the layout in time proportional to their number. Note here that the record for $C$ includes the second node 2; thus, we can insert the appropriate nodes before node 2 in constant time. All other inner nodes are inserted next to nodes of $K$. We can easily color the edges, also in time proportional to their number.

If $s \neq 1$ and $v_s$ dominates some blocks, then we find the highest such block $R$ and the last outer node $v_l$ that sees $R$. If Case 2 applies, we have to find out which one of the subcases 2a, 2b, or 2c is appropriate. Since we know the two border nodes, we can tell whether 2a applies. Suppose that it does not. Let $j$ be the largest node of $C$, other than $v_{s+1}$, which is adjacent to $v_s$. If $j \geqslant m+1$, then Case 2c applies; if $j \leqslant m$, then Case 2b applies. To determine $j$, we scan the adjacency list of $v_s$ (in counterclockwise order), starting from the edge $(v_s, v_{s+1})$, until we find the first edge to a node that has been already laid out; it is easy to see that this is node $j$. We can determine whether $j \geqslant m+1$ or $j \leqslant m$ in constant time by looking up $j$ in the color array.

Once we know which subcase applies, we can easily complete Step 1a in linear time. Case 3 is analogous. Finally, the preparation for the recursive calls, i.e., the insertion of the appropriate records into the stack of cycles that remain to be expanded, takes constant time per cycle. It is easy to see that the information in these records is readily available. Maybe the only thing that is not entirely obvious is how to tell whether an edge $(v_j, v_{j+1})$ of $K$ is a chord of $C$, in which case we have to expand the corresponding cycle $K_j$. For $j=1$, we know the second node 2 of $C$ from the record of $C$. For $1 < v_j < m$, the next node of $C$ (i.e., node $v_j+1$) is the node that follows $v_j$ in the initial layout (by P2). Similarly with the nodes $v_j$, where $m < v_j < p$. For $v_j = m$ the turning point, the next node of $C$ is of course the other border node, which is included in the record of $C$.

To summarize, Step 1a takes time linear in the number of edges that are colored in this step, plus the time we spent in Case 2 to determine which subcase applies (i.e., to find the largest node $j$ of $C$ adjacent to $v_s$). Since every edge is colored exactly once, the contribution of the first term over the whole graph is linear. We claim that the contribution of the second term is also linear. To see this, just notice that the edges incident to $v_s$, which we scanned to find out $j$, will not be scanned again in the process of determining which subcase of Case 2 applies for any other cycle. We conclude:

THEOREM 3. *The algorithm executes in time linear in the size of the graph.*

## REFERENCES

[BC] F. BERNHART AND B. KAINEN, The book thickness of a graph, *J. Combin. Ser. B* **27** (1979), 320–331.

[BS] J. F. BUSS AND P. W. SHOR, On the pagenumber of planar graphs, *in* "Proceedings, 16th Annu. ACM Symp. on Theory of Computing 1984, 98–100.

[CLR1] F. R. K. CHUNG, F. T. LEIGHTON, AND A. L. ROSENBERG, Diogenes: A methodology for designing fault-tolerant VLSI processor arrays, *in* "13th Internat. Conf. on Fault-tolerant Computing, 1983," pp. 22–32.

[CLR2] F. R. K. CHUNG, F. T. LEIGHTON, AND A. L. ROSENBERG, Embedding graphs in books: A graph layout problem with applications to VLSI design, *SIAM J Algebraic Discrete Methods* (1986).

[EI] S. EVEN AND A. ITAI, Queues, stacks and graphs, *in* "Theory of Machines and Computations" (Z. Kohavi and A. Paz, Eds.), pp. 71–86, Academic Press, New York, 1971.

[GJMP] M. R. GAREY, D. S. JOHNSON, G. L. MILLER, AND C. H. PAPADIMITRIOU, The complexity of coloring circular arcs and chords, *SIAM J. Algebraic Discrete Methods* **1** (1980), 216–227.

[GO] M. C. GOLUMBIC, "Algorithmic Graph Theory and Perfect Graphs, "Academic Press, New York, 1980.

[H] L. HEATH, Embedding planar graphs in seven pages, *in* "Proceedings, 25th Annu. Symp. on Foundations of Computer Science, 1984," pp. 74–83.

[HT] J. HOPCROFT AND R. E. TARJAN, Efficient planarity testing, *J. Assoc. Comput. Mach.* **21** (1974), 549–568.

[I] S. ISTRAIL, An algorithm for embedding planar graphs in six pages, manuscript, 1986.

[R] A. L. ROSENBERG, The Diogenes approach to testable fault-tolerant arrays of processors, *IEEE Trans. Comput.* **C-32** (1983), 902–910.

[T] R. E. TARJAN, Sorting using networks of queues and stacks, *J. Assoc. Comput. Mach.* **19** (1972), 341–346.

[W] A. WIGDERSON, "The Complexity of the Hamiltonian Circuit Problem for Planar Graphs," Princeton Univ. technical report.

[Y] M. YANNAKAKIS, in preparation.