Simple planar graph partition into three forests

Roberto Grossi\textsuperscript{a,\dagger}, Elena Lodi\textsuperscript{b}

\textsuperscript{a}Dipartimento di Sistemi e Informatica, Universit\'a di Firenze, via Lombroso 61/7, 50134 Firenze, Italy
\textsuperscript{b}Dipartimento di Matematica, Universit\'a di Siena, via del Capitano 15, 53100 Siena, Italy

Received 27 April 1996; received in revised form 28 October 1997; accepted 1 December 1997

Abstract

We describe a simple way of partitioning a planar graph into three edge-disjoint forests in $O(n \log n)$ time, where $n$ is the number of its vertices. We can use this partition in Kannan et al.'s graph representation (1992) to label the planar graph vertices so that any two vertices' adjacency can be tested locally by comparing their names in constant time. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Planar graphs; Forest partition; Edge coloring

1. Introduction

A number of ways of storing a graph compactly can be found in the literature [2, 3, 10–12, 14, 19]. Among other things, this is useful to save space in distributed environments where the adjacency matrix of the given graph cannot be stored in each node. Kannan et al. [12] describe an elegant method for representing the adjacency matrices of some graph families without having the whole adjacency information available in every single vertex. Among the graph families examined, the authors study the graphs with bounded arboricity $k$, i.e., the graphs which can be decomposed into $k$ edge-disjoint spanning forests for the minimum integer $k$ (see Nash–Williams' theorem [4, 15]). Given one such graph $G$, Kannan et al. assign names to its vertices so that the adjacency of any two vertices can be tested locally by comparing their corresponding names. To this end, they prelabel the vertices with distinct integers and partition $G$ into $k$ forests. A vertex’s name is then given by the $(k+1)$-tuple made up of the vertex’s label and its parents’ labels in the $k$ forests. The authors use Picard and Queryranne’s algorithm to partition the graph into $k$ edge-disjoint forests.

\dagger This work is partially supported by MURST of Italy.
\dagger Corresponding author. E-mail: grossi@dsi.unifi.it.
in $O(n^2m \log^2 n)$ time (or $O(n^4)$ time for dense graphs), where $n$ is the number of vertices and $m$ is the number of edges. A subsequent result by Gabow [6] gives a faster partitioning algorithm whose running time is $O(kn\sqrt{m + kn\log n})$.

In their paper, Kannan et al. examine the special case of planar graphs, that is, graphs that can be drawn on the plane so that no two edges intersect (the resulting drawing is called planar embedding [9]). Since planar graphs have $m = O(n)$ edges and $k = 3$ arboricity [16], their technique immediately produces vertex names which are quadruples. In this case, we have to face the problem of partitioning a planar graph into three forests. We can still use Gabow’s algorithm for this and do the task in $O(n\sqrt{n\log n})$ time. In this paper, we use the planarity hypothesis in order to describe a simple $O(n\log n)$-time partitioning into three edge-disjoint forests.

We use a well-known corollary to Euler’s famous theorem on planar graphs (1750) stating that every planar graph contains a vertex whose degree is five at most [16] (where a vertex degree is defined as the number of its incident edges) and the Jordan curve theorem stating that a closed curve $C$ with no crossings divides the plane into two disjoint regions whose boundary is $C$. We also introduce a graph transformation that maintains the planar embedding and a certain kind of edge coloring. Given a color $a$ and an edge color labeling, we say that there is an $a$-cycle if we find a cycle whose edges are all the same color $a$. We label the edges with three colors, so that for any color $a$ there are no $a$-cycles. We call this kind of edge coloring three-color cycle-free labeling (in short, 3CF coloring). Consequently, partitioning a planar graph into three forests amounts to determining a 3CF coloring. Any two edges are the same color if and only if they belong to the same forest.

Some comments are in order. Partitioning a planar graph into four forests can be done in linear time while obtaining five or more forests is straightforward as we can always choose a vertex with degree five at most (see [5]). In order to get $k = 3$ forests, we can try some intuitive approaches, such as repeatedly removing all of the spanning forest’s edges from the graph (see Fig. 1), but they do not seem to work properly. Our solution works for the problem regarding the three forests by a detailed case analysis. We first show a graph reduction maintaining planar embedding in order to obtain a 3CF coloring. We then describe our planar graph partitioning algorithm. When treating planar graphs, our algorithm can be used in Kannan et al.’s representation [12] instead of the aforementioned algorithms.

2. Graph reduction

We let $G$ be an undirected planar graph having $n$ vertices and $m$ edges, with $m \leq 3n - 6$. We denote the set of $G$’s edges by $E$ and fix one of $G$’s planar embeddings (no two edges intersect each other in the plane). We assume that the embedding is represented as follows: Each vertex $u$ has an associated adjacency list that contains the vertices adjacent to $u$ taken clockwise. We define a graph transformation that maintains this embedding.
Given a vertex $u \in G$ with degree $d \leq 5$, let $w_0, \ldots, w_{d-1}$ be its adjacent vertices (also called neighbors) in clockwise order. In this paper, we use the convention that subscripted indices are modulo $d$; that is, $w_i$ denotes $w_{i \mod d}$ for every integer $i$.

We now define operation $\text{Reduce}(G, u)$ to handle $u$ according to its degree $d$:

Case $d = 3$: We remove $u$ and the edges linking $u$ to $w_0, \ldots, w_{d-1}$ (see Fig. 2(i)).

Case $d = 4$: We find a pair $w_i, w_{i+2}$ of $u$'s neighbors, such that edge $(w_i, w_{i+2})$ does not belong to $E$. This pair exists because of the Jordan curve theorem. We then remove $u$ and the edges linking it to $w_0, \ldots, w_3$. We add a new edge $(w_i, w_{i+2})$ (see Fig. 2(ii)).

Case $d = 5$: We determine an edge, i.e., $(w_{i-1}, w_{i+1})$, that belongs to $E$. If this edge does not exist, then we pick out an arbitrary pair $w_{i-1}, w_{i+1}$ of $u$'s neighbors. We then replace $u$ and the edges linking it to $w_0, \ldots, w_4$ with new edges $(w_{i-2}, w_i)$ and $(w_i, w_{i+2})$. Moreover, we install edges $(w_{i-1}, w_i)$ and $(w_i, w_{i+1})$ if they do not already exist (see Fig. 2(iii)).

Note that $(w_{i-2}, w_i)$ and $(w_i, w_{i+2})$ cannot belong to $E$. Indeed: (1) If $(w_{i-1}, w_{i+1})$ does not exist for every $i$, then $(w_{i-2}, w_i)$ and $(w_i, w_{i+2})$ cannot exist either. (2) If $(w_{i-1}, w_{i+1})$ exists, then $w_{i-1}, u, w_{i+1}$ are the vertices in a cycle that divides the plane into two regions: $w_i$ must be in one of the two regions, while $w_{i-2}$ and $w_{i+2}$ must be
in the other region. The existence of either \((w_{i-2}, w_i)\) or \((w_i, w_{i+2})\) would contradict the Jordan curve theorem (analogously to the case \(d = 4\)).

Our graph reduction maintains the initial graph embedding (this implies that the resulting graph is planar).

**Lemma 1.** Every planar embedding for a graph \(G\) is also a planar embedding for the graph obtained by applying \(\text{Reduce}(G, u)\).

**Proof.** By checking the three operations above, it is soon obtained. The details can be found in [7]. \(\Box\)

### 3. Graph 3CF coloring

We now take the sequence of graphs \(G_n, G_{n-1}, \ldots, G_1\), such that \(G_n = G\) and \(G_{j-1}\) is obtained by applying \(\text{Reduce}(G_j, u_j)\) (for \(j = n, \ldots, 2\)), where \(u_j\) denotes one of \(G_j\)'s minimum-degree vertices. By the definition of \(\text{Reduce}\), \(G_j\) has \(j\) vertices and is still
planar by Lemma 1. Consequently, $u_j$’s degree is five at most by Euler’s theorem and $G_n, G_{n-1}, \ldots, G_1$ is a well-defined sequence of planar graphs.

Our basic idea in obtaining $G$’s 3CF coloring consists of processing the graphs in reverse order: $G_1, \ldots, G_{n-1}, G_n$. We use three colors (blue, green and red) and variables $a, b, c, x, y \in \{\text{blue, green, red}\}$ to denote these colors. We show that $G_j$’s 3CF coloring can be obtained from $G_{j-1}$’s by induction for $j = 2, 3, \ldots, n$. The inductive basis holds because $G_1$ has no edges.

For $j > 1$, we let $E_j$ be the set of $G_j$’s edges and their 3CF coloring be represented by a mapping $C_j : E_j \rightarrow \{\text{blue, green, red}\}$ that assigns the colors to $G_j$’s edges, such that for any color $\alpha \in \{\text{blue, green, red}\}$ there is no $\alpha$-cycle. We denote $u_j$’s neighbors by $w_0, \ldots, w_{d-1}$ (we have $d \leq 5$) and use $e_i = (u_j, w_i)$ to indicate the edge in $E_j$ linking $u_j$ to its neighbor $w_i$. Since we have $C_{j-1}$ by induction, we show how to obtain $C_j$ according to $u_j$’s degree $d$:

Case $d = 3$: We have $E_j = E_{j-1} \cup \{e_0, \ldots, e_{d-1}\}$ and three new edges $e_0, \ldots, e_{d-1}$ at most. We define $C_j$ to be the same as $C_{j-1}$ when its edges are in $E_{j-1} \subseteq E_j$ and have enough colors for assigning different colors to the remaining edges $e_0, \ldots, e_{d-1}$ in $E_j$ (see Fig. 3(i)).

Case $d = 4$: We have $E_j = E_{j-1} - \{l\} \cup \{e_0, \ldots, e_3\}$, where $l = (w_i, w_{i+2})$. We let $a = C_{j-1}(l)$ be $l$’s color in $G_{j-1}$ (see Fig. 3(ii)). We define $C_j$ to be the same as $C_{j-1}$ when its edges are in $E_{j-1} - \{l\} \subseteq E_j$. For the remaining edges in $E_j$, we set $C_j$ as follows: we assign color $a$ to both $e_i$ and $e_{i+2}$ and the other two colors to the edges in $\{e_0, \ldots, e_3\} - \{e_i, e_{i+2}\}$.

Case $d = 5$: We have $E_j = E_{j-1} - E' - \{l_1, l_2\} \cup \{e_0, \ldots, e_4\}$, where $l_1 = (w_i, w_{i+2})$, $l_2 = (w_i, w_{i+2})$ and $E' \subseteq \{(w_{i-1}, w_i), (w_i, w_{i+1})\}$ denotes the set of edges added by $\text{Reduce}(G_j, u_j)$ in order to link $w_i$ to $w_{i-1}$ and $w_{i+1}$ when the corresponding edge
is not in $E_j$ (see Fig. 4, right). Let $a = C_{j-1}(l_1)$ and $b = C_{j-1}(l_2)$, and define $C_j$ to be the same as $C_{j-1}$ when its edges are in $E_{j-1} - E' - \{l_1, l_2\} \subseteq E_j$. Note that the edges in $E' \cup \{l_1, l_2\}$ are discarded because they do not belong to $E_j$. We now have to define $C_j$ for the edges left: $e_0, \ldots, e_4$. There are two possible cases:

1. If $a = b$, then we assign color $a$ to $e_{i-2}, e_i$ and $e_{i+2}$ and the other two colors to the edges in $\{e_0, \ldots, e_4\} - \{e_{i-2}, e_i, e_{i+2}\}$ (see Fig. 4(1)).

2. If $a \neq b$, then we let $c$ be the remaining color. We first define a boolean predicate $\text{path}_{j-1}(v_1, v_2, x, l)$: it is true when there is a path of edges belonging to $E_{j-1} - \{l\}$, such that all the edges are color $x$ and connect vertices $v_1$ and $v_2$ together. We now
need three subcases to assign colors to $e_0, \ldots, e_4$ according to path$_{j-1}$ (we also need to change some previously assigned colors in the last subcase):

2a. If path$_{j-1}(w_{i-2}, w_{i-1}, a, l_1)$ is false (see Fig. 4(2a)), then we assign color $a$ to $e_{i-2}$ and $e_{i-1}$, color $b$ to $e_i$ and $e_{i+2}$, and color $c$ to $e_{i+1}$.

2b. If path$_{j-1}(w_{i+1}, w_{i+2}, b, l_2)$ is false (see Fig. 4(2b)), we assign the colors symmetrically as in subcase 2a: we assign color $b$ to $e_{i+1}$ and $e_{i+2}$, color $a$ to $e_i$ and $e_{i-2}$, and color $c$ to $e_{i-1}$.

2c. In all other cases, we take $(w_{i-1}, w_i)$'s color $x$ and $(w_i, w_{i+1})$'s color $y$ assigned by $C_{j-1}$ (see Fig. 4(2c)). These two edges exist in $E_{j-1}$ thanks to Reduce. If they belong to $E_j$, they are not discarded and their colors are changed by $C_j$: $(w_{i-1}, w_i)$'s color becomes $a$ and $(w_i, w_{i+1})$'s color becomes $b$. We assign color $x$ to $e_{i-1}$ and color $y$ to $e_{i+1}$, and we assign color $a$ to $e_{i-2}$, color $b$ to $e_{i+2}$, and color $c$ to $e_i$.

We now prove that the above case analysis produces $G_j$'s 3CF coloring. We first examine the more involved subcase 2c when $d = 5$ and then go on to the other cases (Lemma 2).

Let us assume that $C_{j-1}$ is a 3CF coloring. We claim that in subcase 2c for $d = 5$, no $x$-cycles traversing either $(w_{i-1}, w_i)$ or $(w_i, w_{i+1})$, or one of $u_j$’s incident edges are possible, where $x \in \{\text{blue, green, red}\}$. In order to see why, we let $a, b$ and $c$ denote the three colors (without specifying them) and $x$ and $y$ the two (maybe equal) colors used in subcase 2c (see Fig. 4(2c)). According to the Jordan curve theorem, we partition the plane into an internal region that is delimited on the outside by the convex hull of $u_j$’s neighbors $w_0, \ldots, w_4$ (i.e., the region delimited by the embedding of $u_j$ and its neighbors, together with their linking edges) and an external region (i.e., what is left by removing the internal region). We now prove our claim. Let us assume by contradiction that an $x$-cycle exists in $G_j$ and traverses either $(w_{i-1}, w_i)$ or $(w_i, w_{i+1})$, or one of $u_j$’s incident edges. Three cases follow according to color $x$ (see Fig. 4(2c)):

(1) Case $x = c$. Since $u_j$’s only incident edges having color $c$ in $G_j$ are the ones linking $u_j$ to $w_{i-1}, w_i$ and $w_{i+1}$ at most, the $c$-cycle traverses two of these edges. Let us assume that they are $(w_{i-1}, u_j)$ and $(u_j, w_i)$ without any loss in generality (and so $x = c$). We can deduce that a path (whose edges are all color $c$) connects $w_{i-1}$ to $w_i$ in the external region. Since this path also exists in $G_{j-1}$, and $(w_{i-1}, w_i)$ is color $x = c$ in $G_{j-1}$, we obtain a $c$-cycle in $G_{j-1}$ (a contradiction).

(2) Case $x = a$. Color $x$ satisfies $x \neq a$ because of $C_{j-1}$ (see Fig. 4(2c), right). Let us examine $G_j$. We deduce that $(w_{i-1}, w_i)$, $e_{i-2}$ and $e_{i+1}$ are the only edges whose color can be $a$ in the internal region. Therefore, the $a$-cycle traverses both $(w_{i-1}, w_i)$ and one of $u_j$’s incident edges, or it traverses either $(w_{i-1}, w_i)$ or one of $u_j$’s incident edges.

If the $a$-cycle traverses both $(w_{i-1}, w_i)$ and one of $u_j$’s incident edges, then $y = a$ because $e_{i+1}$ is the only edge (other than $e_{i-2}$) incident to $u_j$ whose color is $a$. We deduce that $w_i$ and $w_{i+1}$ are connected by a path (in the external region) whose edges are all color $a$. This path is also in $G_{j-1}$, and $(w_i, w_{i+1})$’s color in $G_{j-1}$ is $y = a$; therefore, we obtain an $a$-cycle in $G_{j-1}$ (a contradiction).
If the $a$-cycle only traverses one of $u_j$’s incident edges, then $y = a$. We deduce that $w_{i-2}$ and $w_{i+1}$ are connected by a path (in the external region) whose edges are all color $a$. Since this path is also in $G_{j-1}$ and both $(w_{i-2}, w_i)$ and $(w_i, w_{i+1})$ are color $a$ in $G_{j-1}$, we obtain an $a$-cycle in $G_{j-1}$ (a contradiction).

If the $a$-cycle only traverses $(w_{i-1}, w_i)$, then its endpoints are connected together by a path (in the external region) whose edges are all color $a$. This path exists also in $G_{j-1}$. Since $\text{path}_{j-1}(w_{i-2}, w_{i-1}, a, l_1)$ is true and $(w_{i-2}, w_i)$ is color $a$, we have a contradiction in $G_{j-1}$.

(3) Case $a = b$ is analogous to case $a = a$ (the $b$-cycle involves $(w_i, w_{i+1})$ and $y \neq b$).

Lemma 2. Given a $3CF$ coloring for $G_{j-1}$’s edges, we can determine a $3CF$ coloring for $G_j$’s edges, where $2 < j \leq n$.

Proof. Since $C_{j-1}$ is a $3CF$ coloring for $G_{j-1}$’s edges, we show that for any $\alpha \in \{\text{blue, green, red}\}$ there are no $\alpha$-cycles produced in $G_j$ by $C_j$’s colors. Let us assume that an $\alpha$-cycle exists by contradiction. Since the edges that are added to form $E_j$ are all incident to $u_j$, the $\alpha$-cycle must traverse at least two of $u_j$’s incident edges (except for the subcase 2c discussed previously). We go on to prove that we always obtain a contradiction according to our case analysis.

In case $d \leq 3$ (see Fig. 3(i)), we use different colors and so no two edges can be incident to $u_j$ and be the same color. This means that no $\alpha$-cycles can be created at all.

In case $d = 4$ (see Fig. 3(ii)), we can only have $\alpha = a$ because it is the only color assigned to two edges incident to $u_j$. We previously saw that the two edges are $e_i$ and $e_{i+2}$. However, going on to replace the two edges with $l = (w_i, w_{i+2})$ would produce an $a$-cycle in $G_{j-1}$ and therefore contradict the hypothesis that $C_{j-1}$ is a $3CF$ coloring.

In case $d = 5$, since subcase 1 is similar to case $d = 4$ (see Figs. 3(ii) and 4.1), we focus our attention on subcases 2a–2c. Let $a, b$ and $c$ be the different colors used. In subcase 2a (see Fig. 4(2a), right), an $a$-cycle is not possible in $G_j$ because it would traverse $e_{i-2}$ and $e_{i-1}$ and imply that $\text{path}_{j-1}(w_{i-2}, w_{i-1}, a, l_1)$ is true in $G_{j-1}$, where $l_1 = (w_{i-2}, w_i)$. A $b$-cycle would not be possible in $G_j$ either because it would traverse $e_i$ and $e_{i+2}$ and imply the existence of a $b$-cycle in $G_{j-1}$ traversing $(w_i, w_{i+2})$, which contradicts the fact that $C_{j-1}$ is a $3CF$ coloring. Finally, no $c$-cycles exist because $e_{i+1}$ is the only edge (incident to $u_j$) whose color is $c$. The same holds for the symmetrical subcase 2b. Subcase 2c is special because we change two edges’ colors and so an $a$-cycle can traverse them whether or not it also traverses $u_j$’s incident edges. However, the claim discussed before this lemma shows that no such $a$-cycles are possible. This completes our case analysis.

In brief, we showed that it is possible to build a $3CF$ coloring from $C_{j-1}$ and we proved the lemma’s statement. □

Corollary 3. We can always determine a $3CF$ coloring for a planar graph $G$. 

Proof. The sequence \( G_n, \ldots, G_1 \) is well formed by Lemma 1. \( G_1 \) trivially has a 3CF coloring because it only contains one vertex. By induction and Lemma 2, we have that \( G_n = G \) and so \( G \) has a 3CF coloring.

4. Edge-disjoint forests' construction

Our algorithm for building a partition into three forests applies the Reduce operation to the input graph until a single vertex is obtained. Then, it examines the sequence of intermediate planar graphs so obtained backward, and assigns the colors to their edges following the case analysis discussed in Section 3. Its high-level description is shown in Fig. 5. We give some comments below and specify the relevant implementation details.

We first execute Hopcroft and Tarjan's linear-time algorithm [9] for finding \( G \)'s planar embedding in step (1). This is useful in step (2) to represent the adjacency lists in \( G \) according to its embedding. The adjacency between any two vertices can be verified in constant time and linear space (say, by an easy-to-compute partition into \( k = 5 \) forests and by Kannan et al.'s adjacency method [12]). In steps (3)–(6), we obtain the sequence of graphs \( G_n, G_{n-1}, \ldots, G_1 \). We maintain an array indexed by the vertices' degrees, in which the vertices of the same degree are kept in a doubly linked list. We are able to determine \( u_j \) in constant time by scanning the array's first five entries. When a vertex's degree changes because of \( \text{Reduce}(G_j, u_j) \), we update the array and the adjacency lists in constant time. We store \( u_j \) and the \( O(1) \) edges involved by \( \text{Reduce} \) into a stack cell, so as to be able to obtain \( G_j \) from \( G_{j-1} \) subsequently. We spend a total of \( O(n) \) time in steps (3)–(6). We produce a 3CF coloring by means of steps (7)–(10). Initially, \( C_1 \) is empty as \( G_1 \) is just an isolated vertex. In step (9), we retrieve \( u_j \) and the edges that contributed to get \( G_{j-1} \) from \( G_j \) in constant time. At this point, we execute step (10) to apply our case analysis presented in Section 3. The efficient implementation of this step deserves more discussion below. Finally, we give the three forests as output in step (11). Each forest consists of the edges in \( G - G_n \) whose colors are identical in the 3CF coloring \( C_n \).

---

1. Execute Hopcroft-Tarjan algorithm to find \( G \)'s planar embedding;
2. \( G_n := G; \quad /* \text{with its planar embedding} */ 
3. for \( j := n \) downto 2 do
   4. Find a min-degree vertex \( u_j \) in \( G_j \);
   5. \( G_{j-1} := \text{Reduce}(G_j, u_j); \)
   6. Push \( u_j \), its incident edges and the edges in \( G_j \) into a stack;
   7. \( C_1 := \text{empty}; \quad /* \text{initial 3CF coloring} */ 
8. for \( j := 2 \) to \( n \) do
   9. Pop \( u_j \) and its companion edges from the stack;
10. Compute \( C_j \) from \( C_{j-1} \) by the case analysis on \( u_j \) given in Section 3;
11. Output the three forests, each forest identified by the edges with same color in \( C_n \).

---

Fig. 5. Pseudocode for partitioning an \( n \)-vertex planar graph \( G \) into three forests.
In the rest of this section, we describe the data structures and the algorithms to implement step (10) in $O(\log n)$ time. Let us therefore examine the corresponding case analysis, presented in Section 3. Let $d$ be $u_j$'s degree. When $d \leq 4$, we only have to treat the edges retrieved in step (9). When $d = 5$, we also need edges $(w_{i-1}, w_i)$ and $(w_i, w_{i+1})$ in Fig. 4(2c). However, we may have to check predicate $\text{path}_{j-1}(v_1, v_2, x, l)$ to see if there is a path of edges different from edge $l$, such that all the edges are color $x$ and connect vertices $v_1$ and $v_2$ together. This motivates the following intermediate subproblem: for a given color $x$, maintain a forest (i.e., the edges color $x$) under insertion and deletion of edges so as to answer queries $\text{path}_{j-1}(v_1, v_2, x, l)$.

We use Henzinger and King's technique [8] to solve our subproblem. For each (unrooted) tree $T$ in the forest, we maintain its Euler tour $ET(T)$: if $T$ has $q$ vertices, then $ET(T)$ is a sequence of $2q - 1$ symbols, which are the vertices visited in preorder after $T$ is rooted at a vertex. Every edge is visited twice and every vertex of degree $c$ occurs $c$ times in $ET(T)$, except the root, which occurs $c + 1$ times. We store $ET(T)$ in the leaves of a 2-3 tree (from left to right); we can split it at any leaf or concatenate it to another 2-3 tree in logarithmic time [1]. By using this, Henzinger and King show how to change the root, split a tree by means of an edge removal, merge two trees by linking their roots together through an edge insertion, and establish whether or not two vertices belong to the same tree, in logarithmic time per operation. We use these operations in our subproblem as follows.

In order to insert an edge $(u, v)$ in the forest, we take the tree $T_u$ containing $u$ and the tree $T_v$ containing $v$. We then make $T_u$ rooted at $u$ and $T_v$ rooted at $v$. We merge the two trees at their roots through edge $(u, v)$. Deleting an edge $(u, v)$ is analogous: we take the tree $T$ containing the edge and make it rooted at $u$. Then, we split the subtree rooted at $v$ (which a child of $u$). As a result, we obtain two smaller trees from $T$. Both insertion and deletion take logarithmic time (see [8] for more details).

In order to answer $\text{path}_{j-1}(v_1, v_2, x, l)$, where $l$ is a forest edge, we delete $l$. That is, the tree $T$ containing $l = (u, v)$ is split in two subtrees $T_u$ and $T_v$, the former containing $u$ and the latter containing $v$. Then, we check to see if both $v_1$ and $v_2$ belong to the same tree in the forest where $T$ is replaced by $T_u$ and $T_v$. The answer is returned by $\text{path}_{j-1}$. We then re-insert $l$ to get $T$ in place of $T_u$ and $T_v$ in the forest. The cost is logarithmic time because it takes a constant number of operations on the Henzinger–King data structure.

We now turn to our implementation of step (10) in the pseudocode shown in Fig. 5. We assume to have inductively computed coloring $C_{j-1}$ for graph $G_{j-1}$ (initially, for $j = 2$, this holds vacuously). Let us therefore assume that we have three Henzinger–King data structures, one per color. We only discuss how to implement case $d = 5$, subcase 2c, as the other cases are easier to handle. We perform the $\text{path}$ query necessary to case $d = 5$. Subsequently, we have to remove edges $l_1, l_2$ and the edges in $E'$ from $G_{j-1}$ (they were recorded in step (6) and retrieved in step (9)). We delete them from the Henzinger–King data structures of the proper color (e.g., $l_1$ is color $a$ and so we remove it from the data structure for $a$). We then change the color (i.e., remove from one of the Henzinger–King data structures and insert into another) of
edges \((w_{i-1}, w_i)\) and \((w_i, w_{i+1})\), if necessary. We insert \(e_0, \ldots, e_4\) into their proper data structures after their color is given. The resulting Henzinger–King data structures are correctly maintained for the next inductive step on \(j\). This completes the algorithmic description and shows that \(C_j\) can be obtained from \(C_{j-1}\) in logarithmic time. The total cost of steps (7)–(10) is \(O(n \log n)\). According to Corollary 3, we obtain our result:

**Theorem 4.** A planar graph with \(n\) vertices can be partitioned into three forests in \(O(n \log n)\) time.

5. Concluding remarks

We showed that partitioning a planar graph with \(n\) vertices into three forests takes \(O(n \log n)\) time. The source for such a cost is due to the possibility that the min-degree vertex has degree \(d = 5\). In this case, we have to answer the path query and maintain the Henzinger–King data structures at a logarithmic cost (we can avoid all these problems if we want to obtain four or more forests). If case \(d = 5\) never occurs, our partitioning algorithm clearly becomes linear time. It would be interesting to obtain a linear-time algorithm that also works for the general case. An independent result presented in [18] shows how to find an acyclic 3-coloring of planar graphs in linear time, with the colors assigned to the vertices. However, the resulting forests are vertex disjoint and so this result does not seem to apply directly to our problem, in which we require that the forests are edge disjoint, i.e., they can share some vertices.

Acknowledgements

We are in debt to Nicola Santoro for suggesting the problem to us. We also thank Srecko Brlek, Sajal Das, Donatella Merlini, Ugo Vaccaro and Jeff Westbrook for giving us some helpful references and research material.

References


