Implementing Path 3-Coloring and Path 3-Choosing Algorithms on Plane Graphs

Aven Bross

August 26, 2016

Abstract

Path coloring a graph partitions its vertices into sets inducing a disjoint union of paths. In this project we consider several algorithms to compute path colorings of graphs embedded in the plane. We first implement an algorithm to path 3-color plane graphs from Poh's proof in [a]. Second, we present a linear time implementation of an algorithm to path 3-choose plane graphs from the independant work of Hartman [b] and Skrekovski [c].

1 Introduction

All graphs discussed in this project will be simple, undirected, and finite. We say a graph is planar if it may be drawn in the plane without edge crossings, known as a plane drawing. A planar embedding of a graph is an ordering of the edges around each vertex according to a plane drawing. A plane graph is a graph with a planar embedding. A k-coloring of a graph partitions its vertices into k color classes. Such a coloring is called proper if each color class consists of snonadjacent vertices.

Appel and Haken [d,e] display that all planar graphs have a proper 4-coloring. This result is best possible and known as the Four Color Theorem. Generalizations of proper coloring were introduced in [f,g,h] allowing color classes to form forests, or allowing vertices share a color with some bounded number of neighbors. Cowen et al. ([j]) show all planar graphs may be 3-colored such that each vertex shares a color with at most two neighbors.

We will be considering the problem of path coloring, producing a k-coloring of a graph such that each color class induces a disjoint union of paths. A disjoint union of paths is equivalent to a linear forest, a forest where each component is a path. This coloring was introduced by Harary in [i]. Note that this is similar to the defective coloring of Cowen et al. above, with the added restriction that path coloring forbids cycles. In [l] Poh displayes that all planar graphs have a path 3-coloring. We present an implementation of Poh's algorithm to path 3-color plane graphs.

Given a list of k colors for each vertex, a k-list-coloring, or k-choosing, assigns each vertex a color from its list. If a graph has a proper k-choosing it is said to be k-choosable. List-coloring was first introduced by Erdös et al. in [m]. Thomassen in [n] proves that all planar graphs are 5-choosable. Planar graphs that are not 4-choosable are described by Mirzakhani in [o] and Voigt in [p], so Thomassen's result is best possible. Jensen and Toft in [t] note that Thomassen's proof yields a linear algorithm for 5-choosing plane graphs.

Hull and Eaton in [q] prove planar graphs are 3-choosable such that each vertex shares a color with at most two neighbors, and furthermore show this result is best possible. Hartman in [r] and Skrekovski in [s] independantly provide similar proofs that planar graphs are path 3-choosable. Hartman claims the proof yields a linear time algorithm for path 3-list-coloring, and thus path 3-coloring, plane graphs. We present a linear time implementation of Hartman and Skrekovski's algorithm.

2 Path 3-Coloring Plane Graphs

We first restate the theorem and proof of Poh [l]. This proof yields a simple algorithm for path 3-coloring plane graphs.

Theorem 1. Let G be a 2-connected weakly triangulated plane graph or a complete graph on two vertices and suppose the outer face C has been 2-colored such that each color class induces a non-empty path. This 2-coloring may be extended to a path 3-coloring of G such that no vertex in $V(C)$ recieves a same color neighbor in $V(G) \setminus V(C)$.

Proof. If $|V(G)| \leq 3$ the coloring follows trivially. Let $|V(G)| > 3$ and suppose the theorem holds for all graphs H with $|V(H)| < |V(G)|$. Let $P = p_0 \dots p_n$ and $Q = q_0 \dots q_m$ denote the two induced paths from the 2-coloring of C such that the edges p_0q_0 and p_nq_m are in C. Suppose there exist uncolored vertices, that is $V(G) \setminus V(C) \neq \emptyset$.

Let t_0 be the vertex forming a face with p_0 and q_0. If $t_0 \in P$, this face is already colored and we consider the graph bounded by $P - p_0$ and Q. Similarly, if $t_0 \in Q$ then the inductive hypothesis applies to the graph bounded by P and $Q - q_0$. Let t_1 be the vertex forming a face with p_n and q_m and proceed in the same manner until t_1 is not in either path.

Suppose there exists an induced path T from t_0 to t_1. We color T the remaining color not assigned to P or Q and apply the inductive hypothesis to the subgraph bounded by P and T, and the subgraph bounded by T and Q. With only the path T in common between the two subgraphs, the combined 3-coloring forms a path coloring of G.

Suppose no such path exists from t_0 to t_1. Since G is weakly triangulated there must exist an edge $p_iq_j \in E(G) \setminus E(C)$ with $p_i \in P$ and $q_j \in Q$. We separately apply the inductive hypothesis to the subgraph bounded by $p_0 \dots p_i$ and $q_0 \dots q_j$, and the subgraph bounded by $p_i \dots p_n$ and $q_j \dots q_m$. The two subgraphs only share the vertices p_i and q_j, thus the combined 3-coloring forms a path coloring of G.

□

Corollary 2. All plane graphs are path 3-colorable.

Proof. Let G be a plane graph. Since adding edges does not make G easier to color, augment G with edges until it is triangulated. Path 2-color the outer triangle and apply Theorem 1 to path 3-color G.

□

Poh's Algorithm

Let G be a weakly triangulated plane graph with outer face formed by the two monocolor paths P and Q, each colored sdistinct color. Poh's path 3-coloring algorithm proceeds as follows:

1. If all vertices in G are colored, terminate;
2. Locate the vertex t_1. If $t_1 \in P$ remove p_n and repeat. If $t_1 \in Q$ remove q_m and repeat;

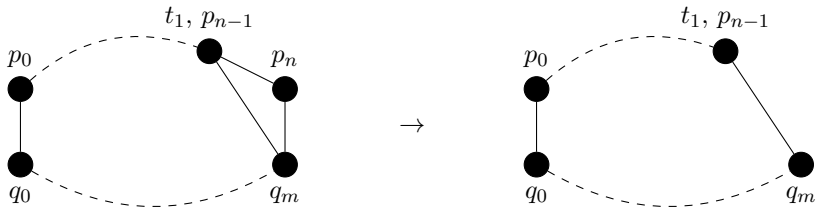

Figure 2.1 The case $t_1 \in P$.

3. Perform a breadth first search from t_1;
4. Terminate the search when we find a vertex u with adjacent neighbors p_i and q_j;
5. Color with the remaining color the induced ut_1-path determined by the breadth first search;
6. Apply the algorithm to the subgraph bounded by the p_0p_i-path and q_0q_i-path, the subgraph bounded by the p_ip_n-path and ut_1-path, and the subgraph bounded by the ut_1-path and q_jq_m-path.

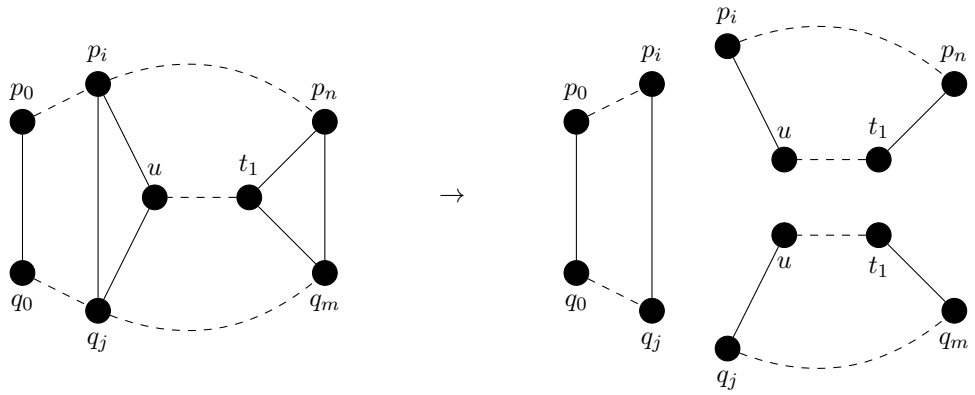

Figure 2.2 Dividing G along the edge p_iq_j and the ut_1-path.

Implementing Poh's Algorithm

Let the plane graph G be represented as an incidence list (or adjacency list) and an ordering of edges (neighbors) around each vertex following a combinatorial embedding. We track the paths P and Q by marking each vertex with its respective path and storing the path start and end vertices p_0, p_n, q_0, and q_m. We first find t_1 by looking through the ordered neighbors of q_m and take the vertex counterclockwise past p_n. This step is repeated until the graph is colored or $t_1 \notin P \cup Q$.

We perform a breadth first search starting at t_1 and storing parents for each vertex visited. Vertices marked to be in P or Q will be ignored, in this way containing the search within the current bounded subgraph. The search terminates once a vertex u with adjacent neighbors $p_i \in P$ and $q_j \in Q$ has been reached. An induced ut_1-path is produced by backtracking through the search from u. We color and mark each vertex on the new path with the remaining color not used to color P or Q.

To color the remaining graph we recurse on both the region bounded by the p_ip_n-path and ut_1-path, and the region bounded by the ut_1-path and q_jp_m-path. If $p_i = p_0$ and $q_j = q_0$ we are done and this mimicks the case of a t_0t_1 path in with $u = t_0$. If $p_i \neq p_0$ or $q_j \neq q_0$ we handle the remaining subgraph by recursing on the region bounded by the p_0p_i-path and q_0p_j-path. Note each recursive step is independant and vertex marks are shared so the algorithm may instead proceed iteratively by pushing paths, represented by their start and end vertices, into a stack or queue.

Time Complexity

Locating t_1 requires a single neighbor lookup. The amortized complexity of a neighbor lookup is $O(|E|/|V|)$. Each vertex may be t_1 at most once so over the entire graph we perform at most $|V|$ neighbor lookups. In planar graphs $|E| \leq 3|V| - 6$, and $O(|E|) = O(|V|)$. Therefore, the amortized complexity of this step is $O(|V|^2/|V|) = O(|V|)$. We also perform at most one breadth first search from each t_1 with complexity $O(|V|)$. Therefore the complexity of the serach step over the entire graph is $O(|V|^2)$. This gives an overall amortized complexity of $O(|V| + |V|^2) = O(|V|^2)$.

3 Path 3-Choosing Plane Graphs

The following is a restatment of a theorem of Hartman [r] and Skrekovski [s]. We provide a slight modification of the proof technique that wraps the path coloring and path removal portions of Hartman's proof into a single inductive step that considers only the last vertex colored on the current path, which will be denoted p. This modification is motivated by the relative simplicity of removing a single outer face vertex, compared with removing an entire path. We will show this proof yields a linear time algorithm and implementation, as claimed by Hartman in [r].

The informal idea followed in both the proof and algorithm is to start with vertices x and y on the outer face and draw a path clockwise between them using vertices on the outer face. We mark the current color of the path with α and the current end of the path with p. At each step we will remove p and look for a vertex between p and y on the outer face to color α and extend our path. If no such vertex is found, or we reach and remove y, we move p back to x and start a new path.

To avoid dealing with disconnected graphs the algorithm will separately consider the 2-connected components produced when removing p results in cutvertices. In fact, we incrementally remove p and halt once a single cutvertex is found. The algorithm then considers the two maximal 2-connected components separately. We will continue to remove p from the component it is contained in.

Suppose C is the outer cycle of a 2-connected weakly triangulated plane graph G. Using notation from [r] for $u, v \in V(C)$ we let $C[u, v]$ denote the path from u to v clockwise along the outer face. If we wish to exclude u or v from this path we will use parenthesis, $C(u, v)$. Similarly, for $v \in V(G)$ and $u, w \in N(v)$ we let $[u, w]_v$ denote the path from u to w clockwise around v, assuming

triangulated faces.

Note that in all figures solid circles denote vertices yet to be colored, and colored vertices will be labeled with their assigned color. We have α denote the current path color and a label β represent coloring v from $L(v)\setminus\{\alpha\}$. If a vertex v is unlabled it represents arbitrary coloring from $L(v)$.

Theorem 3. Let G be a 2-connected weakly triangulated plane graph, or a complete graph on one or two vertics, with outer face C. Let $x, y \in V(C)$ be not necessarily distinct, potentially precolored vertices. Let $p \in C[x, y]$ be precolored some color α. Suppose $L(v)$ assigns a list of colors to each $v \in V(G)$ that has not been precolored such that

$$\begin{array}{ll} |L(v)| \geq 1 & \text{if } v = x \text{ or } v = y; \\ |L(v)| \geq 2 & \text{if } v \in V(C) \setminus \{x, y\}; \\ |L(v)| \geq 3 & \text{otherwise.} \end{array}$$

If $p \neq x$, let $\alpha \notin L(v)$ for any $v \in V(C[x, p))$. Also assume the precoloring is a path coloring.

The coloring may be extended to a path choosing of G from L such that x, y, and p each share a color with at most one neighbor. If $x = y$ then x and y each share a color with none of their neighbors. If $y = p$, or y is immediately prior to p on the outer face and $\alpha \notin L(y)$, then y shares a color with none of its neighbors.

Proof. If $|V(G) \leq 2$ the theorem easily follows. Suppose $|V(G)| > 2$ and the theorem holds for all graphs H with $|V(H)| < |V(G)|$. Let $C = c_0c_1 \dots c_n$ denote, in clockwise order, the outer face of G with $p = c_0$. There are several cases to consider. Let c_i be the next vertex in $V(C) \cap N(p)$ counterclockwise from c_n. Let G_0 be the subgraph bounded by the cycle formed from $C[c_i, c_n]$ and $[c_n, c_i]_p$. If $c_i \neq c_1$ let G_1 be the sugbraph bounded by the cycle formed from $C[p, c_i]$ and the edge pc_i. As seen in Figure 3.1 $G = G_0 \cup G_1$ and $V(G_0) \cap V(G_1) = \{c_i\}$. We will display in each case that the inductive hypothesis holds for each subgraph and their union still forms a path choosing of G from L. If $c_i = c_1$ we will say G_1 does not exist and handle this case specially, noting $G_0 = G - p$.

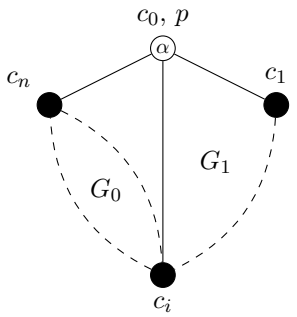

Figure 3.1 The subdivision of G into G_0 and G_1.

For all $v \in (N(p) \cap V(G_0)) \setminus \{x, y\}$ we note if $v = c_n$ or $v = c_i$ then $|L(v)| \geq 2$, and $|L(v)| \geq 3$ otherwise. We define a new list assignment L_0 such that $L_0(v) = L(v) \setminus \{\alpha\}$ for $v \in N(p) \cap V(G_0)$, and $L_0(v) = L(v)$ for $v \in V(G_0) \setminus N(p)$. Note that all $v \in N(p) \cap V(G_0)$ will be on the outer face of G_0. Thus $|L_0(v)| \geq 3$ for all interior vertices v of G_0. Except for a few special cases for y mentioned below, $|L_0(v)| \geq 1$ for all $v \in \{x, y, c_i, c_n\}$. Finally, $|L_0(v)| \geq 2$ for all other v on the outer face of G_0. By choosing G_0 from L_0 we ensure either p recieves no α colored neighbors in G_0 unless c_i is colored α. If c_i is colored α then p recieves no α colored neighbors in G_1 other than c_i.

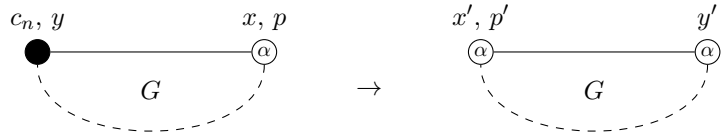

Figure 3.2 The case $x = p$, $y = c_n$, and $\alpha \in L(y)$.

If $\alpha \in L(y)$ and $y \in V(G_0) \cap N(p)$ it may be that $|L_0(y)| = 0$. Note that in the case y has been precolored a color distinct from α we will say $\alpha \notin L(v)$. To handle this we first note $y \in \{c_n, c_i\}$. The case $y = c_i$ will require no special treatment as c_i will naturally be colored α. Suppose $\alpha \in L(y)$ and $y = c_n$. Then $x = p$, since otherwise $y \in C[x, p)$ and $\alpha \notin L(y)$. Color y with α and apply the inductive hypothesis to choose G from L with $x' = y$, $y' = x$, and $p' = y$. Otherwise suppose if $\alpha \in L(y)$ then $y \neq c_n$.

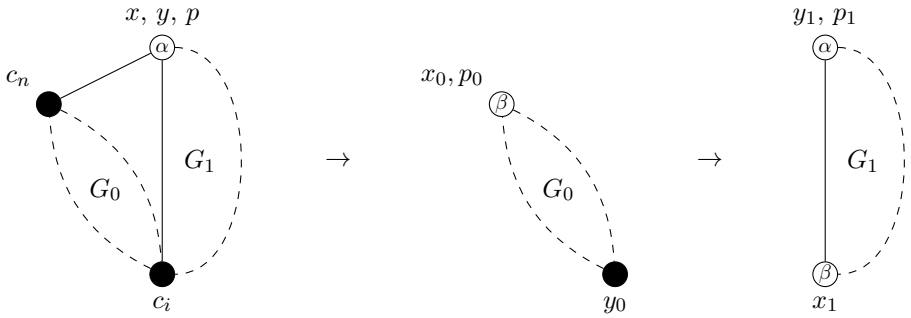

Figure 3.3 The case $x = y = p$.

Suppose $x = y = p$. Color c_i from $L_0(c_i)$. Apply the inductive hypothesis to choose G_0 from L with $x_0 = c_n$, $y_0 = p_0 = c_i$. If G_1 exists we apply the inductive hypothesis again to choose G_1 from L with $x_1 = c_i$, $y_1 = y$, and $p_1 = p$. Since c_i recieves at most one neighbor in each G_0 and G_1, the combined coloring forms a path choosing of G from L.

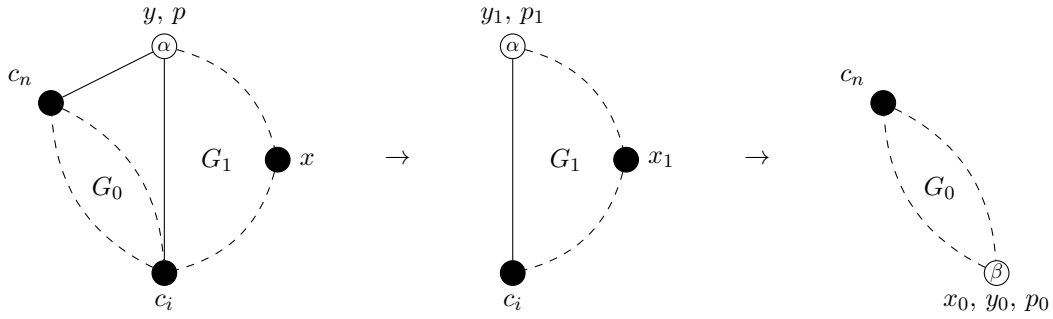

Figure 3.4 The case $y = p$, $x \neq p$, and $c_i \in C[x, p)$ (shown is the case $x \neq c_i$).

Suppose $y = p$, $x \neq p$, and $c_i \in C[x, p)$. If G_1 exists, apply the inductive hypothesis to choose G_1 from L with $x_1 = x$, $y_1 = y$, and $p_1 = p$. Apply the inductive hypothesis to choose G_0 from L_0 with $x_0 = y_0 = p_0 = c_i$. If G_1 exists note c_i was precolored from the choosing of G_1 and recieves no same color neighbors in G_0. Thus the combined coloring forms a path choosing of G from L.

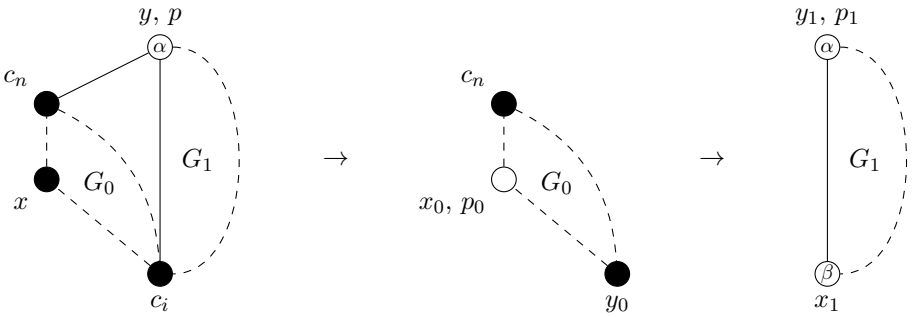

Figure 3.5 The case $y = p$, $x \neq p$, and $c_i \notin C[x, p)$.

Suppose $y = p$, $x \neq p$, and $c_i \notin C[x, p)$. Color x from $L_0(x)$ and apply the inductive hypothesis to choose G_0 from L_0 with $x_0 = p_0 = x$ and $y_0 = c_i$. If G_1 exists we apply the inductive hypothesis again to choose G_1 from L with $x_1 = c_i$, $y_1 = y$, and $p_1 = p$. Notice c_i recieves at most one same color neighbor in each G_0 and G_1.

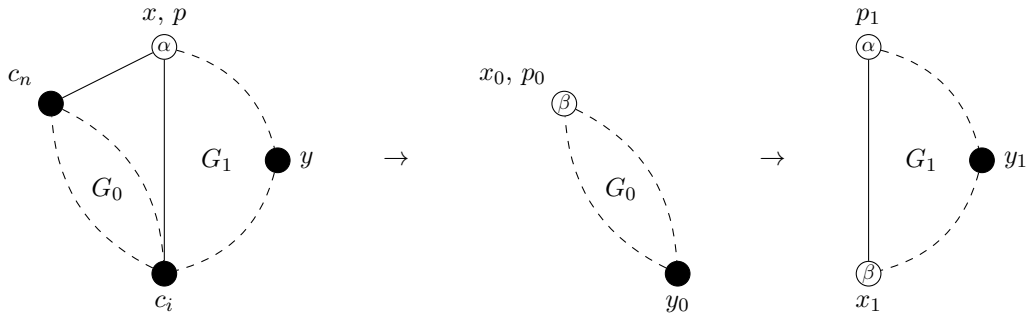

Figure 3.6 The case $x = p$, $y \neq p$, and $c_i \in C(y, x)$.

Suppose $x = p$, $y \neq p$, and $c_i \in C(y, x)$. Apply the inductive hypothesis to choose G_0 from L_0 with $x_0 = p_0 = c_n$ and $y_0 = c_i$. In this case G_1 must exist and we apply the inductive hypothesis to choose G_1 from L with $x_1 = c_i$, $y_1 = y$, and $p_1 = p$. Notice c_i recieves at most one same color neighbor in each G_0 and G_1.

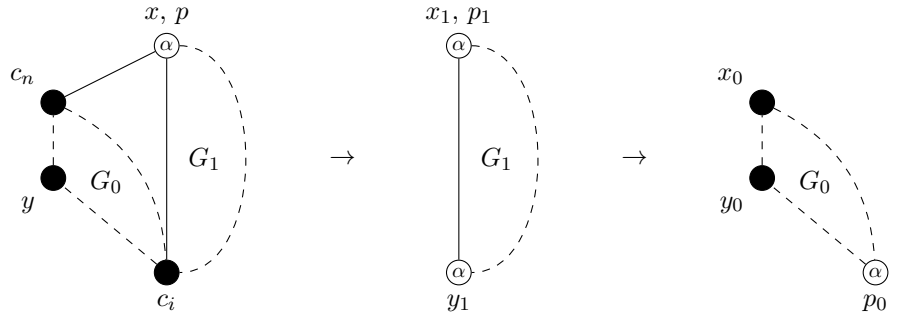

Figure 3.7 The case $x = p$, $y \neq p$, and $c_i \notin C(y, x)$ (shown is the case $y \neq c_i$ and $\alpha \in L(c_i)$).

Suppose $x = p$, $y \neq p$, and $c_i \notin C(y, x)$. If $\alpha \in L(c_i)$ we set $p_0 = c_i$ and color c_i with α. Otherwise, set $p_0 = c_n$ and color it with the first color in $L_0(c_n)$, note $|L_0(c_i)| \geq 2$ in this case. Apply the inductive hypothesis to choose G_0 from L_0 with $x_0 = c_n$, $y_0 = y$, and p_0. If G_1 exists, apply the inductive hypothesis to choose G_1 from L with $x_1 = p_1 = p$ and $y_1 = c_i$. Notice if $p_0 = c_i$, c_i recieves at most one same color neighbor in G_0 and the single same color neighbor p in G_1.

Furthermore, p will recieve no same color neighbor in G_1 other than c_i. If $p_0 \neq c_i$, then $y_1 = c_i$ is immediatley prior to $p_1 = p$ and $\alpha \notin L(c_i)$. Thus c_i will recieve no same color neighbors in G_1.

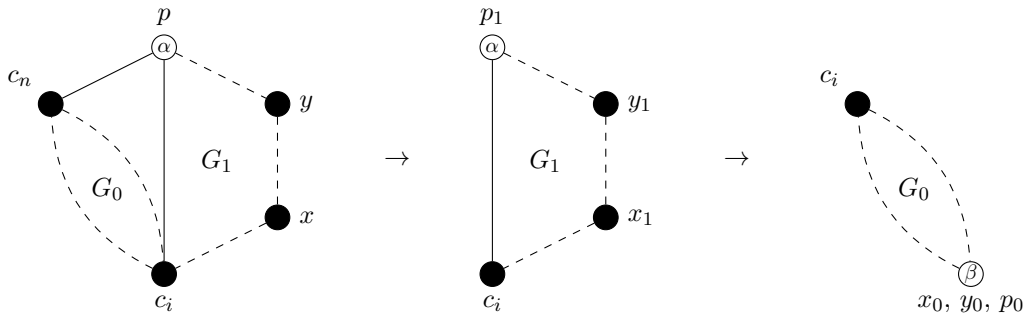

Figure 3.8 The case $x \neq p$, $y \neq p$, and $c_i \in C[x, p)$ (shown is the case $x \neq c_i$).

Suppose $x \neq p$, $y \neq p$, and $c_i \in C[x, p)$. In this case G_1 must exist and we apply the inductive hypothesis to choose G_1 from L with $x_1 = x$, $y_1 = y$, and $p_1 = p$. Apply the inductive hypothesis again to choose G_0 from L_0 with $x_0 = y_0 = p_0 = c_i$. Note that c_i was precolored from the choosing of G_1 and recieves no same color neighbors in G_0 so the combined coloring forms a path choosing of G from L.

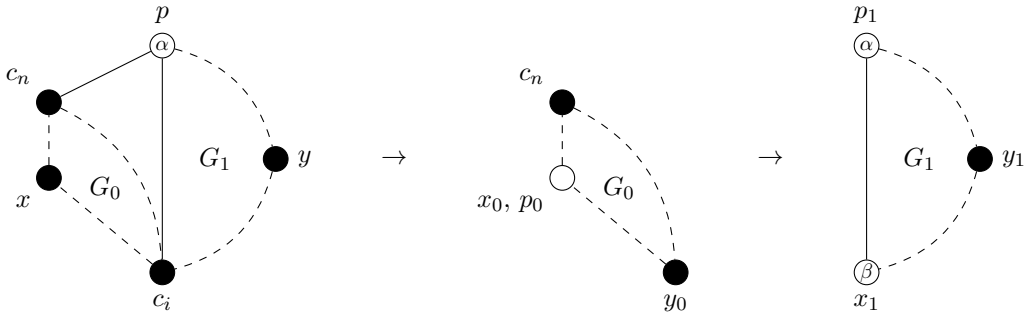

Figure 3.9 The case $x \neq p$, $y \neq p$, and $c_i \in C(y, x)$.

Suppose $x \neq p$, $y \neq p$, and $c_i \in C(y, x)$. Apply the inductive hypothesis to choose G_0 from L_0 with $x_0 = p_0 = x$ and $y_0 = c_i$. In this case G_1 must exist and we apply the inductive hypothesis again to choose G_1 from L with $x_1 = c_i$, $y_1 = y$, and $p_1 = p$. Notice c_i recieves at most one same color neighbor in each G_0 and G_1.

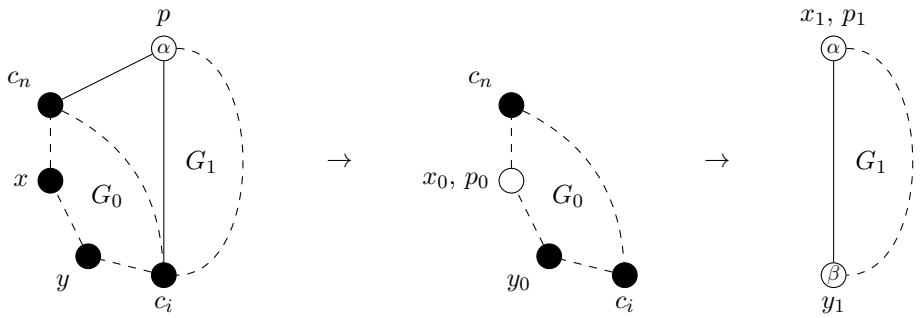

Figure 3.10 The case $x \neq p$, $y \neq p$, and $c_i \in C(p, y]$ (shown is the case $y \neq c_i$ and $\alpha \notin L(c_i)$).

Finally, suppose $x \neq p$, $y \neq p$, and $c_i \in C(p, y]$. If $\alpha \in L(c_i)$ we set $p_0 = c_i$ and color c_i with α. Otherwise, define $p_0 = c_1$ and color it from $L_0(c_1)$, noting $|L_0(c_i)| \geq 2$ in this case. Apply the

inductive hypothesis to choose G_0 from L_0 with $x_0 = x$, $y_0 = y$, and p_0. If G_1 exists, apply the inductive again hypothesis to choose G_1 from L with $x_1 = p_1 = p$ and $y_1 = c_i$. Note that c_i was precolored by our choosing of G_0. If $p_0 = c_i$, c_i recieves at most one same color neighbor in G_0 and the single same color neighbor p in G_1. Furthermore, p will recieve no same color neighbor in G_1 other than c_i. If $p_0 \neq c_i$, then $y_1 = c_i$ is immediatley prior to $p_1 = p$ and $\alpha \notin L(c_i)$. Thus c_i will recieve no same color neighbors in G_1.

□

Corollary 4. All planar graphs are path 3-choosable.

Proof. Let G be a plane graph and L an assignment of size 3 color lists to each vertex of G. As adding edges does not make G easier to color, augment G with edges until it is triangulated (maximally planar). Assign x and y to be distinct vertices on the outer face. Color x from $L(x)$ and assign $p = x$. Apply Theorem 3 to path choose G from L accordingly.

□

Path 3-Choosing Algorithm

Let G be a weakly triangulated plane graph with vertices x, y, and p defined and colored in accordance with Theorem 3. We will again denote the outer face of G as $C = c_0c_1 \dots c_n$ with $c_0 = p$. We will denote the embedding ordered $N(p)$ as $n_0 \dots n_k$, with $n_0 = c_1$ and $n_k = c_n$. We will denote the parameters for an application of the algorithm as the tuple (G, x, y, p). The path 3-choosing algorithm proceeds as follows:

1. If p has not been colored, color p from $L(p)$;
2. If $n_k = y$ and $\alpha \in L(y)$, color y with α and apply the algorithm to G with $x' = p' = y$ and $y' = p$;
3. Remove α from $L(n_k)$;
4. For $j = k - 1$ to 0: if $n_j \in C$ terminate, otherwise remove α from $L(n_j)$;
5. If $c_i \notin C(p, y]$ remove α from $L(c_i)$;
6. We now have $n_j = c_i \in C$ and define G_0 as the subgraph with outer face $C_0 = c_i \dots c_n n_{k-1} \dots n_{j-1}$ and G_1 as the subgraph with outer face $c_0 \dots c_i$. If $c_i = c_1$ we ignore G_1;
   (a) If $x = y = p$, apply the algorithm first to (G_0, c_n, c_i, c_n) and then to (G_1, c_i, p, p);
   (b) If $y = p$, $x \neq p$, and $c_i \in C[x, p)$, apply the algorithm first to (G_1, x, p, p) and then (G_0, c_i, c_i, c_i);
   (c) If $y = p$, $x \neq p$, and $c_i \in C(y, x)$, apply the algorithm first to (G_0, x, c_i, x) and then to (G_1, c_i, p, p);
   (d) If $x = p$, $y \neq p$, and $c_i \in C(y, x)$, apply the algorithm first to (G_0, c_n, c_i, c_n) and then to (G_1, c_i, p, y);
   (e) If $x = p$, $y \neq p$, $c_i \in C(p, y]$, and $\alpha \in L(c_i)$, color c_i with α and apply the algorithm first to (G_1, p, c_i, p) and then to (G_0, c_n, y, c_i);
   (f) If $x = p$, $y \neq p$, $c_i \in C(p, y]$, and $\alpha \notin L(c_i)$, apply the algorithm first to (G_1, p, c_i, p) and then to (G_0, c_n, y, c_n);

(g) If $x \neq p$, $y \neq p$, $c_i \in C[x,p)$, apply the algorithm first to (G_1, x, p, y) and then to (G_0, c_i, c_i, c_i);

(h) If $x \neq p$, $y \neq p$, $c_i \in C(y,x)$, apply the algorithm first to (G_0, x, c_i, x) and then to (G_1, c_i, y, p);

(i) If $x \neq p$, $y \neq p$, $c_i \in C(p,y]$, and $\alpha \in L(c_i)$, color c_i with α and apply the algorithm first to (G_0, x, y, c_i) and then to (G_1, p, c_i, p);

(j) If $x \neq p$, $y \neq p$, $c_i \in C(p,y]$, and $\alpha \notin L(c_i)$, apply the algorithm first to (G_0, x, y, x) and then to (G_1, p, c_i, p).

Implementing the Path 3-Choosing Algorithm

Let G be a 2-connected weakly triangulated plane graph represented as an incidence list (or adjacency list) and an ordering of edges around each vertex following a planar embedding. We will track subgraphs by marking each vertex with a *state* of *interior*, *face*, or *colored*. Vertices will only change in state from *interior* to *face* to *colored*. Each *face* and *colored* vertex will have a *neighbor range* that tracks the range of edges in its ordered incidence list that belong to the current subgraph. This range will start with the next vertex on the outer face and end with the previous. Therefore we may walk clockwise along the outer face from a *face* vertex v simply by proceeding to the first vertex in its *neighbor range*.

To be able to perform the path 3-choosing algorithm above we must also determine where each *face* vertex lies on the outer face with respect to x, y, and p. Each vertex will recieve a mark associating it with a face segment. We will denote $C[x,p)$ as s_p, $C(p,y]$ as s_y, and $C(y,x)$ as s_x. Each face segment may simply be represented by an integer type. The face segments s_p and s_y will need to be joined in the case $\alpha \notin L(c_i)$ and $c_i \in C(p,y]$ since p will be moved away to start a new path. We utilize a disjoint set data structure, as described in [u], to allow efficient union and find operations on face segments.

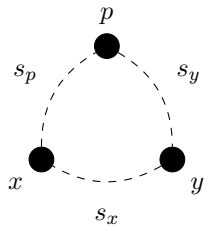

Figure 3.11 The face segments of G.

Some additional steps must be added to the path 3-choosing algorithm to maintain each data structure during its operation. In the case that we color y and rename vertices in step (2) we must also rename face segments to swap s_x with s_y (see figure 3.2). In step (4) we perform the following for each n_j, not including c_i, in addition to the regular steps of the algorithm:

1. Initialize n_j's *neighbor range* and set its *state* to *face*;
2. Set n_j's *face location* to s_p, creating s_p if it doesn't exist;
3. Contract the *neighbor range* of n_j to remove p from its incidence list.

Once the above procedure terminates we will have located c_i. We have almost fully determined G_0 and G_1 by our tracking of *state* and *neighbor range*. It only remains to find the *neighbor range* for c_i in both G_0 and G_1. This can be achieved by locating p in c_i's incidence list and splitting the *neighbor range* of c_i at the pc_i edge. Comparing c_i, x, y, and p, along with the *face location* and *state* properties for each we may determine which case to apply in step (6). In each recursive call we must also pass any face segments that appear and rename each accordding to their position with respect to the new assignments of x, y, and p. As noted earlier, in (6e) and (6j) for G_0 we set s_y to be the union of s_p and s_y (see figure 3.10).

Time Complexity

First note that G is planar so $O(|E|) = O(|V|)$. When a vertex is assigned as p we will iterate through its incidence list at most once. This is due to the fact that when each neighbor n_j is hit, we remove p from n_j's *neighbor range*. Once each neighbor has been hit, p has been effectively removed from the graph. Therefore step (4) iterates through each edge in $E(G)$ at most twice, once for each incident vertex. Furthermore, steps (1) through (3) will occur at most once for each iteration of step (4). Thus this process has complexity $O(|E|) = O(|V|)$.

We will be performing numerous unions and finds throughout the operation of the algorithm, but no more than $O(|V|)$ by the same reasoning as above. The only unions that will be made are between s_p and s_y. We may also notice that the segment s_p is always a singleton in our disjoint set structure, only created when new *interior* vertices are added to the outer face while removing p. Therefore the complexity of each union and find operation is $O(1)$ and the complexity of maintaining our face segment disjoint set structure over the entire operation of the algorithm is $O(|V|)$.

The remaining expensive operation is locating the edge c_ip in c_i's incidence list. If c_ip is an edge on the outer face, $c_i = c_1$ and this lookup is constant time using the tracked *neighbor range*. Otherwise, c_ip is not an edge in the outer face. We perform a neighbor lookup to find the edge with amortized complexity $O(|E|/|V|)$. After this step c_ip will be in the outer face of both G_0 and G_1. Therefore we perform at most one lookup for each edge in $E(G)$. This gives an amortized complexity of $O(|E||E|/|V|) = O(|V|)$.

Therefore the amortized complexity of the path 3-choosing algorithm is $O(|V|)$.