# Path-Coloring Algorithms for Plane Graphs

Aven Bross
Department of Computer Science
University of Alaska
Fairbanks, AK 99775-6670
dabross@alaska.edu

Glenn G. Chappell
Department of Computer Science
University of Alaska
Fairbanks, AK 99775-6670
chappellg@member.ams.org

Chris Hartman
Department of Computer Science
University of Alaska
Fairbanks, AK 99775-6670
cmhartman@alaska.edu

June 1, 2017

*2010 Mathematics Subject Classification.* Primary 05C38; Secondary 05C10, 05C15.
*Key words and phrases.* Path coloring, list coloring, algorithm.

**Abstract**

A path coloring of a graph $G$ is a vertex coloring of $G$ such that each color class induces a disjoint union of paths. We present two efficient algorithms to construct a path coloring of a plane graph.

The first algorithm, based on a proof of Poh, is given a plane graph; it produces a path coloring of the given graph using three colors.

The second algorithm, based on similar proofs by Hartman and Škrekovski, performs a list-coloring generalization of the above. The algorithm is given a plane graph and an assignment of lists of three colors to each vertex; it produces a path coloring of the given graph in which each vertex receives a color from its list.

Implementations of both algorithms are available.

## 1 Introduction

All graphs will be finite, simple, and undirected. See West [10] for graph theoretic terms.

A *path coloring* of a graph $G$ is a vertex coloring (not necessarily proper) of $G$ such that each color class induces a disjoint union of paths. A graph $G$ is *path $k$-colorable* if $G$ admits a path coloring using $k$ colors.

Broere & Mynhardt conjectured [2, Conj. 16] that every planar graph is path 3-colorable. This was proven independently by Poh [8, Thm. 2] and by Goddard [6, Thm. 1].

**Theorem 1.1** (Poh 1990, Goddard 1991). *If $G$ is a planar graph, then $G$ is path 3-colorable.* □

It is easily shown that the "3" in Theorem 1.1 is best possible. In particular, Chartrand & Kronk [5, Section 3] gave an example of a planar graph whose vertex set cannot be partitioned into two subsets, each inducing a forest.

Hartman [7, Thm. 4.1] proved a list-coloring generalization of Theorem 1.1 (see also Chappell & Hartman [4, Thm. 2.1]). A graph $G$ is *path $k$-choosable* if, whenever each vertex of $G$ is assigned a list of $k$ colors, there exists a path coloring of $G$ in which each vertex receives a color from its list.

**Theorem 1.2** (Hartman 1997). *If $G$ is a planar graph, then $G$ is path 3-choosable.* □

Essentially the same technique was used by Škrekovski [9, Thm. 2.2b] to prove a result slightly weaker than Theorem 1.2.

We discuss two efficient path-coloring algorithms based on proofs of the above theorems. We distinguish between a *planar* graph—one that can be drawn in the plane without crossing edges—and a *plane* graph—a graph with a given embedding in the plane.

In Section 2 we outline our graph representations and the basis for our computations of time complexity.

Section 3 covers an algorithm based on Poh's proof of Theorem 1.1. The algorithm is given a plane graph; it produces a path coloring of the given graph using three colors.

Section 4 covers an algorithm based Hartman's proof of Theorem 1.2, along with the proof of Škrekovski mentioned above. The algorithm is given a plane graph and an assignment of a list of three colors to each vertex; it produces a path coloring of the given graph in which each vertex receives a color from its list.

Implementations of both algorithms are available; see Bross [3].

# 2 Graph Representations and Time Complexity

We will represent a graph via *adjacency lists*: a list, for each vertex $v$, of the neighbors of $v$. A vertex can be represented by an integer $0 \dots n-1$, where $n$ is the order of the graph.

A plane graph will be specified via a *rotation scheme*: a circular ordering, for each vertex $v$, of the edges incident with $v$, in the order they appear around $v$ in the plane embedding; this completely specifies the combinatorial embedding of the graph. Rotation schemes are convenient when we represent a graph using adjacency lists; we simply order the adjacency list for each vertex $v$ in clockwise order around $v$; no additional data structures are required.

We will assume an integer RAM model of computation. The input for each algorithm will be a triangulated plane graph with $n$ vertices and $m$ edges, represented via adjacency lists. The input size will be $n$, the number of vertices. Note that $\mathcal{O}(m) = \mathcal{O}(n)$, so it is equivalent to take the input size to be $m$, the number of edges. Moreover, arbitrary simple planar graphs may be plane embedded and triangulated in $\mathcal{O}(n)$ time, see [1].

In Section 4, given an edge $uv$, we will need an efficient operation to find $v$'s entry in $u$'s adjacency list from $u$'s entry in $v$'s list. An *augmented adjacency list* is an adjacency list such that for any edge $uv$, a reference to $v$'s entry in $u$'s list is stored in $u$'s entry in $v$'s list, and vice versa. Given an adjacency list representation of a graph, an augmented adjacency list representation may be constructed in $\mathcal{O}(m)$ time via the following procedure.

**Algorithm 2.1.**
**Input:** An adjacency list representation Adj of a graph $G$.
**Output:** An augmented adjacency list representation Adj' of $G$ with the same rotation scheme as Adj.
**Step 1:** Construct an augmented adjacency list copy Adj' of Adj with the reference portion of each entry uninitialized.
**Step 2:** For each vertex $v$ construct an array Wrk[$v$] storing vertex-reference pairs. For each $v$ from 0 to $n-1$ iterate through $\text{Adj}'[v]$. For each neighbor $u$ in $\text{Adj}'[v]$ append the pair $(v, r_v(u))$ to Wrk[$u$] where $r_v(u)$ is a reference to $u$'s entry in $\text{Adj}'[v]$.
**Step 3:** For each $v$ from $n-1$ to 0 iterate through Wrk[$v$]. Upon reaching each pair $(u, r_u(v))$ in Wrk[$v$] the last element of Wrk[$u$] will be $(v, r_v(u))$; for details on why this is, see the paragraphs below. Use $r_u(v)$ and $r_v(u)$ to look up and assign references for the edge $uv$ in Adj'[$u$] and Adj'[$v$]. Remove $(v, r_v(u))$ from the back of Wrk[$u$].

After completing Step 2 in Algorithm 2.1 the array Wrk[$v$] contains a pair $(u, r_u(v))$ for each neighbor of $v$, sorted in increasing order of by the neighbor $u$.

Let $v$ be the current vertex at a given iteration of Step 3 in Algorithm 2.1. For each edge $uw \in E(G)$ such that $u < w$ and $v < w$, prior iterations of Step 3 will have initialized the references for $uw$ in $\text{Adj}'[u]$ and $\text{Adj}'[w]$, and also removed the pair $(w, r_w(u))$ from Wrk[$u$]. Therefore for each $(u, r_u(v))$ in Wrk[$v$], the array Wrk[$u$] will contain only entries for vertices $w$ where $w \le v$. Hence, since Wrk[$u$] is sorted in increasing order, the last element of Wrk[$u$] must be $(v, r_v(u))$.

# 3 Path Coloring: the Poh Algorithm

We will first describe Poh's algorithm for path 3-coloring plane graphs.

**Algorithm 3.1** (Poh 1990)**.**
**Input:** A weakly triangulated plane graph $G$ with outer face a cycle $C = v_1, v_2, \ldots, v_k$ and a 2-coloring of $C$ such that each color class induces a path, $P_1 = v_1, v_2, \ldots, v_l$ and $P_2 = v_{l+1}, v_{l+2}, \ldots, v_k$ respectively.
**Output:** An extension of the 2-coloring of $C$ to a path 3-coloring of $G$ such that no vertex in $G - C$ receives the same color as a neighbor of that vertex in $C$.
**Step 1:** If $G = C$ then $G$ is already path 3-colored and we are done. Otherwise there are two cases to consider.

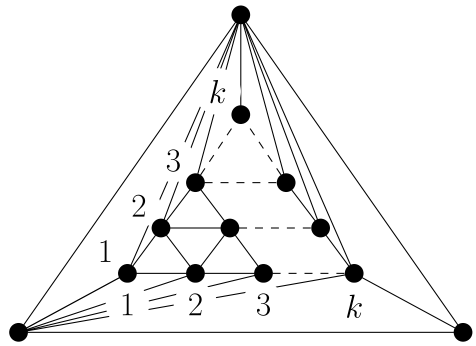


Figure 1: The collection of graphs $\{G_k\}_{k\in\mathbb{N}}$ on which Poh performs poorly.

**Case 1.1:** Suppose $C$ is an induced subgraph of $G$. Let $u, w \in V(G) - V(C)$ such that the cycles $u, v_1, v_k$ and $w, v_l, v_{l+1}$ each exist and are faces of $G$; note that $u$ and $w$ are unique, but may not be distinct. Since $C$ is induced and $G \neq C$, $G - C$ is connected. Let $P_3 = u_1, u_2, \ldots, u_r$ be an induced $u, w$-path in $G - C$. Color each vertex of $P_3$ with the third color not used in the 2-coloring of $C$. Let $C_1 = v_1, v_2, \ldots, v_l, u_r, u_{r-1}, \ldots, u_1$ and $C_2 = u_1, u_2, \ldots, u_r, v_{l+1}, v_{l+2}, \ldots, v_k$.
**Case 1.2:** Suppose $C$ is not an induced subgraph. Then there exists an edge $v_iv_j \in E(G) - E(C)$ such that $i \leq l < j$. Let $C_1 = v_1, v_2, \ldots, v_i, v_j, v_{j+1}, \ldots, v_k$ and $C_2 = v_i, v_{i+1}, \ldots, v_j$.
**Step 2:** Apply Algorithm 3.1 separately to the maximal subgraph of $G$ with outer face $C_1$ and to the maximal subgraph with outer face $C_2$.

Note that the graph $G$ is finite and the recursive step applies the algorithm to two proper subgraphs of $G$. Therefore Algorithm 3.1 must terminate.

Let $G$ be a triangulated plane graph. We may trivially path 2-color the outer triangle. Applying Poh's algorithm extends this coloring to a path 3-coloring of $G$.

In Poh's original proof he picked the induced $u, w$-path in Case 1.1 to be the shortest $u, w$-path. Thus a natural way to implement Poh's algorithm is to first locate $u$ and $w$, and then use a breadth-first search to to either construct a $u, w$-path or locate a chord edge if no such path is possible.

**Algorithm 3.2.**
**Input:** A cycle $C = v_1, v_2, \ldots, v_k$ in a triangulated plane graph $G$, where $G$ is represented by adjacency lists, and a 2-coloring of $C$ such that each color class induces a path, respectively $P_1 = v_1, v_2, \ldots, v_l$ and $P_2 = v_l, v_{l+1}, \ldots, v_k$.
**Output:** An extension of the 2-coloring of $C$ to a path 3-coloring of the maximal subgraph of $G$ with outer face $C$.
**Step 1:** Iterate Adj$[v_1]$ to locate the vertex $u$ immediately following $v_k$. Note that since $G$ is triangulated, $v_1, u, v_k$ is a face of $G$.
**Case 1.1:** If $u \in C$, then $u = v_{k-1}$, since $G$ is triangulated, and $C$ is not an induced cycle. We then apply Algorithm 3.2 to the cycle $C' = v_1, v_2, \ldots, v_{k-1}$.

**Case 1.2:** Perform a breadth-first of the maximal subgraph of $G$ with outer face $C$, starting from the vertex $u$. Terminate the search upon locating a vertex $w \notin C$ with adjacent neighbors $v_i \in P_1$ and $v_j \in P_2$ such that $i \neq 1$ or $j \neq k$. Backtrack along the breadth-first search to construct a minimal $u, w$-path $P_3 = u_1, u_2, \ldots, u_r$. Let $C_1 = v_1, v_2, \ldots, v_i, u_r, u_{r-1}, \ldots, u_1$ and $C_2 = u_1, u_2, \ldots, u_r, v_j, v_{j-1}, \ldots, v_k$. Apply Algorithm 3.2 separately to $C_1$ and $C_2$. If $i = l$ and $j = l + 1$ then $C$ was an induced cycle and we are done. Otherwise, also apply Algorithm 3.2 to $C_3 = v_i, v_{i+1}, \ldots, v_j$.

Unfortunately Algorithm 3.2 is not linear. To see this, consider the family of graphs $\{G_k\}_{k\in\mathbb{N}}$ depicted in Figure 1. Fix $k \in \mathbb{N}$ and note that $n = n(G_k) = k(k + 1)/2 + 3$. Assume that the outer triangle is path 2-colored such that the top vertex is assigned a color distinct from the bottom two. At iteration $i$ of Poh's algorithm the shortest path through the interior will be the path of length $l = k - i + 1$ directly along the base of the inner triangle. A breadth-first search of this inner triangle will hit all $l(l + 1)/2$ vertices in order to find this path. Therefore the total number of operations performed will be

$$\Theta\left(\sum_{l=1}^{k} \frac{l(l+1)}{2}\right) = \Theta(n^{3/2}).$$

So Poh's algorithm with breadth-first search is $\Omega(n^{3/2})$.

However, the correctness of Poh's algorithm only relied on locating some induced $u, w$-path. We will show below that there exists a linear time implementation of Poh's algorithm so long as we do not always find the shortest $u, w$-path.

The general strategy will be to walk clockwise along the colored path $P_1 = v_1, v_2, \ldots, v_l$ in the outer cycle $C$, marking those vertices interior to $C$ that have a neighbor in $P_1$. We will then construct an induced path $P_3 = u_1, u_2, \ldots, u_d$, consisting only of marked vertices, such that $C_1 = P_1 \cup P_3 \cup \{u_1v_1, u_dv_l\}$ is a cycle, and all marked vertices are contained in the maximal subgraph of $G$ with outer face $C_1$.

When denoting vertices in a $k$-cycle it will be a useful convention to treat vertex indices mod $(k+1)$. From now on if we have a cycle $C = v_1, v_2, \ldots, v_k$, then for any $i \in \mathbb{Z}$ we will define $v_i = v_j$ where $j \in \{1, \ldots, k\}$ and $j \equiv i \mod (k + 1)$.

**Algorithm 3.3.**
**Input:** An induced cycle $C = v_1, v_2, \ldots, v_k$ in a triangulated plane graph $G$, where $G$ is represented by adjacency lists, and a 2-coloring of $C$ such that each color class induces a path, respectively $P_1 = v_1, v_2, \ldots, v_l$ and $P_2 = v_l, v_{l+1}, \ldots, v_k$. Additionally, suppose that all vertices in $G - C$ with neighbors in $P_1$ are marked; denote this set of vertices $N(P_1)$.
**Output:** An extension of the 2-coloring of $C$ to a path 3-coloring of the maximal subgraph of $G$ with outer face $C$.
**Step 1:** If $S = \emptyset$ then $G - C$ is empty and thus $G$ is already path 3-colored.

Suppose that $S \neq \emptyset$. Let $u, w$ be the vertices in $S$ such that $v_1, v_k, u$ and $v_l, v_{l+1}, w$ are cycles in $G$. Observe there must exist a $u, v$-path in $G - C$ consisting of vertices in $S$. We will inductively construct an induced path $P_3 = t_1, \ldots, t_r$ such that $t_1 = u$, $t_r = w$, $t_1, \ldots, t_r \in S$, and all vertices in $S$ are contained in the maximal subgraph of $G$ with outer face $P_1 \cup P_3 \cup \{v_1u, v_lw\}$. Concurrently, we will also mark all vertices in $G - C - P_3$ with neighbors in $P_3$ and note any edges between vertices in $P_3$ and vertices in $P_1$ or $P_2$.

Initialize $t_1 = u$. We will also denote $v_k$ as $t_0$ so that $t_{i-1}$ is defined for $i = 1$.

Let $t_1, \ldots, t_i$ be the the induced path constructed so far. Iterate through $\text{Adj}[t_i]$ clockwise starting from $t_{i-1}$ until we reach a vertex $t_{i+1} \in S$ distinct from $t_{i-1}$ or we reach a vertex in $P_1$. If we reach a vertex in $P_1$ first then $t_i$ must be $w$ or there wouldn't be a $u, w$-path in $G$ consisting of vertices in $S$, a contradiction. If we reach a vertex $t_{i+1} \in S$ we add it to the path and continue.

While iterating through $\text{Adj}[t_i]$, let us also mark all neighbors between $t_{i-1}$ and $t_{i+1}$ to track vertices in $G - C - S$ with neighbors in $P_3$. Additionally record edges between $t_i$ and vertices in $P_1$ and $P_2$.

**Step 2:** Color the vertices on the path $P_3$ with the remaining color not used on vertices in $P_1$ or $P_2$. Define the cycles $C_1 = P_1 \cup P_3 \cup \{v_1u, v_lw\}$ and $C_2 = P_3 \cup P_2 \cup \{v_ku, v_{l+1}w\}$. In step 1 we recorded all chords of $C_1$ and $C_2$. Moreover, we have also marked vertices with neighbors in $P_1, P_3$. Therefore we may decompose $C_1, C_2$ into induced cycles and recursively apply Algorithm 3.3 to each.

Suppose that $G$ is a triangulated plane graph. We may path 2-color the outer triangle, mark all vertices with neighbors in one of the colored paths, and then apply Algorithm 3.3 to extend this to a path 3-coloring of $G$.

Note that while executing Algorithm 3.3 we only iterate through the adjacency list a vertex precisely when it is colored and added to a path. Therefore the algorithm is linear in the number of vertices.

# 4 Path List Coloring: the Hartman-Škrekovski Algorithm

ZZZ

# References


[1] J. Boyer and W. Myrvold, On the cutting edge: simplified $O(n)$ planarity by edge addition, *J. Graph Algorithms Appl.* **8** (2004), 241–273.

[2] I. Broere and C. M. Mynhardt, Generalized colorings of outerplanar and planar graphs, *Graph theory with applications to algorithms and computer science* (Kalamazoo, Mich., 1984), pp. 151–161, Wiley-Intersci. Publ., Wiley, New York, 1985.

[3] A. Bross, *Implementing path coloring algorithms on planar graphs*, Masters Project, University of Alaska, 2017, available from `http://github.com/permutationlock/path_coloring_bgl`.

[4] G. G. Chappell and C. Hartman, Path choosability of planar graphs, in preparation.

[5] G. Chartrand and H. V. Kronk, The point-arboricity of planar graphs, *J. London. Math. Soc.* **44** (1969), 612–616.

[6] W. Goddard, Acyclic colorings of planar graphs, *Discrete Math.* **91** (1991), no. 1, 91–94.

[7] C. M. Hartman, *Extremal Problems in Graph Theory*, Ph.D. Thesis, University of Illinois, 1997.

[8] K. S. Poh, On the linear vertex-arboricity of a planar graph, *J. Graph Theory* **14** (1990), no. 1, 73–75.

[9] R. Škrekovski, List improper colourings of planar graphs, *Combin. Probab. Comput.* **8** (1999), no. 3, 293–299.

[10] D. B. West, *Introduction to Graph Theory, 2nd ed.*, Prentice Hall, Upper Saddle River, NJ, 2000.