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 [9] 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 [1, Conj. 16] that every planar graph is path 3-colorable. This was proven independently by Poh [7, Thm. 2] and by Goddard [5, Thm. 1].

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

It is easily shown that the "3" in Theorem 1.1 is best possible. In particular, Chartrand & Kronk [4, Section 3] gave an example of a planar graph whose vertex set cannot be partitioned into two subsets, each inducing a forest.

Hartman [6, Thm. 4.1] proved a list-coloring generalization of Theorem 1.1 (see also Chappell & Hartman [3, 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.* $\square$

Essentially the same technique was used by Škrekovski [8, 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 [2].

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 $0 \ldots n-1$, 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 connected planar graph with n vertices and m edges, represented via adjacency lists. The input size will be n, the number of vertices. Note that $\mathcal{O}(m) = \mathcal{O}(n)$.

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 algorithm.

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 Adj'[v]. For each neighbor u in Adj'[v] let $r(v,u)$ be a reference to u's entry in Adj'[v] and append the pair $(v, r(v,u))$ to Wrk[u].

Step 3: For each v from $n-1$ to 0 iterate through Wrk[v]. At the pair $(u, r(u,v))$ in Wrk[v] note that the last element of Wrk[u] is $(v, r(v,u))$. Look up and assign references for the edge uv in Adj'. 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.

At a given iteration of Step 3 in Algorithm 2.1 let v be the current vertex. 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 Adj'[u] and Adj'[w], and also removed the pair $(w, r(w,u))$ from Wrk[u]. The current iteration will handle all edges $uv \in E(G)$ with $v > u$.

3 Path Coloring: the Poh Algorithm

We will first describe Poh's algorithm for path 3-coloring plane graphs. If C is a cycle in a plane graph G we will use Int(C) to denote the subgraph of G consisting of C and all interior vertices and edges.

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.

Case 1.1: Suppose C is an induced subgraph of G. Let $u, w \in V(G) - V(C)$ such that uv_1v_k and wv_lv_{l+1} are faces; 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 $P_3 = u_1u_2 \ldots u_r$ be a shortest

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 $C_1 = v_1 v_2 \dots v_l u_r u_{r-1} \dots u_1$ and $C_2 = u_1 u_2 \dots u_r v_{l+1} v_{l+2} \dots v_k$.

Case 1.2: Suppose C is not an induced subgraph. Then there exists an edge $v_i v_j \in E(G) - E(C)$ such that $i \le l < j$. Let $C_1 = v_1 v_2 \dots v_i v_j v_{j+1} \dots v_k$ and $C_2 = v_i v_{i+1} \dots v_j$.

Step 2: Apply Algorithm 3.1 to $\mathrm{Int}(C_1)$ and $\mathrm{Int}(C_2)$.

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

4 Path List Coloring: the Hartman-Škrekovski Algorithm

ZZZ

References

[1] 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.

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

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

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

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

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

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

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

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