# AN ALTERNATING CHAIN ALGORITHM WITH ORDER FOR FOUR COLORING PLANAR GRAPHS

JOHN LANGE

Abstract. We present an alternating chain algorithm for coloring planar graphs. At any given stage of the algorithm some of the edges may be conflicted, that is have vertices of the same color. We resolve such conflicts by finding free alternating chains until we eventually obtain a proper coloring. The algorithm works because of the order in which things - vertices, edges, colors - are considered.

## 1. Introduction

Polynomial time algorithms for optimum graph coloring typically involve taking advantage of some special graph property. The algorithm presented here does not require a graph to have any given property but relies on principles of ordering to find the solution. The program was written using the boost BGL.

1991 *Mathematics Subject Classification.* Graph Algorithms.
*Key words and phrases.* Graph, Coloring, Alternating Chain.

## 2. The Algorithm

At each stage of the algorithm an edge is either light or heavy and the graph is properly colored wrt. it's heavy edges. Initially there are only light edges so the graph is properly colored wrt. the heavy edges by default.

**Definition 2.1.** We call an edge *conflicted* if both of it's vertices have the same color.

**Definition 2.2.** By an *ALTERNATING CHAIN* of $G$ we mean a simple path $P = (v_0, v_1, v_2, \ldots, v_n)$ of $G$ st.

(1) The first edge of $P$, $(v_0, v_1)$ is a light conflicted edge.
(2) Every edge of $P$, except the first, is heavy and is unconflicted as are all heavy edges, always.
(3) With each edge $(v_i, v_{i+1})$ of $P$ there is associated a colored arrow $A_{i+1}$ from $v_i$ to $v_{i+1}$. We denote the set of arrows as $\{A_i\}$ for $1 \leq i \leq n$.
(4) We denote an arrow's color by $c(A_i)$ and a vertex's color by $c(v_i)$. For each arrow except the last we have $c(A_i) = c(v_{i+1})$.

**Definition 2.3.** Let $C = (v_0, \ldots, v_n)$ be an alternating chain in $G$. We *apply* $C$ to $G$, denoted by $C(G)$ as follows

(1) For each arrow $A_i$ recolor $v_i$ to $c(A_i)$.
(2) Make the first edge of $C$ heavy. The graph will then contain one more heavy edge.

**Definition 2.4.** We say $C$ is a *free* alternating chain if $C(G)$ is properly colored wrt. it's heavy edges.

For example see Figure 4932. The light edges are turqoise, and the heavy edges are brown.

(1) First notice that the graph is properly colored wrt. the heavy edges.
(2) An alternating chain has been found, $C = (v_{14}, v_{20}, v_3, v_{28})$.
(3) The first edge of $C$, $(v_{14}, v_{20})$ is a light conflicted edge.
(4) Every edge of $C$, except the first, is heavy and is unconflicted as are all heavy edges, always.
(5) $A_1$ is green, $A_2$ is blue, $A_3$ is green.
(6) $c(A_1) = c(v_3)$, $c(A_2) = c(v_{28})$.

This alternating chain is then applied to the graph to get Figure 4933.

(1) We now have $c(v_{20}) = c(A_1),\ c(v_3) = c(A_2),\ c(v_{28}) = c(A_3)$.
(2) The first egde of $C$, $(v_{14}, v_{20})$ is now heavy.
(3) The graph is properly colored wrt. the heavy edges of which there is now one more.
(4) Because the graph is properly colored wrt. the heavy edges we say that $C$ was a *free* alternating chain.

**Definition 2.5.** We say that the conflict across edge $(v_{14}, v_{20})$ has been *resolved*.

We start the algorithm with a graph where all edges are light and all vertices are colored the same color, therefor all edges are conflicted. We resolve conflicts one edge at a time by applying free alternating chains as described above. Eventually there will be no conflicted edges and we have obtained a proper coloring.

## 3. Finding free alternating chains

3.1. **Marking light edges.** Let (a,b) be a light conflicted edge. For every color_X we mark (a,b) with color_X if there is no more than one heavy edge e st.

(1) source(e) = b
(2) e != (b,a)
(3) and color(target(e)) = color_X.

3.2. **Marking heavy edges.** Let (a,b) be a heavy edge. For every color_X we mark (a,b) with color_X if

(1) There is an edge (v,a) st. color( (v,a) ) = color(b)
(2) and there is no more than one heavy edge e st.
    (a) source(e) = b
    (b) e != (b,a)
    (c) and color(target(e)) = color_X.

3.3. **Finding the terminal edge of a free alternating chain.** We continue marking edges as described until we encounter an edge (a, b) marked with color_X st. there is no edge e satisfying the following conditions.

(1) e is heavy
(2) source(e) = b
(3) e != (b,a)
(4) and color(target(e)) = color_X.

In that case (a, b) is a *terminal* edge marked with color color_X.

3.4. **Obtaining a free alternating chain.** Once we have found a terminal edge (x,y) we find it's predecessor edge as follows.

(1) Let e be an in_edge into x, but not (y,x).
(2) If e is marked with color(y), e will be the predecessor of (x,y).
(3) We color e's associated arrow with color(y).
(4) We likewise find a predecessor of e, and so on till we arrive at a light edge.

The sequence of edges so obtained is the free alternating chain. The last edge found ie. the light edge will be the beginning of the alternating chain.

## 4. example

The examples have mostly free alternating chains of length one or two. There are some of length three at the end. B1 and Br2 are the best examples just because they have alternating chains of length three. See the explanations for each example.