Polynomial Algorithm for Finding Chromatic Sum for Unicyclic and Outerplanar Graphs

Ewa M. Kubicka

University of Louisville

Abstract

The chromatic sum of G, $\sum(G)$, is the minimum sum of vertex colors, taken over all proper colorings of G using natural numbers. In general, finding $\sum(G)$ is NP-complete. This paper presents polynomial time algorithms for finding the chromatic sum for unicyclic graphs and for outerplanar graphs.

1. Introduction

The concept of chromatic sum of a graph was introduced in [3] and [4]. If $G = (V, E)$ is a graph with vertex set V and edge set E, then the chromatic sum of G denoted by $\sum(G)$, is the minimum sum of vertex colors $\sum_{v\in V} c(v)$, taken over all proper colorings c of G using natural numbers. A proper coloring c of a graph G is called a best coloring of G whenever $\sum_{v\in V} c(v) = \sum(G)$. The smallest number of colors used in a best coloring is called the strength ($s(G)$) of a graph G. The strength of a graph G might be larger than the chromatic number, $\chi(G)$, of G. For example, Figure 1 represents the smallest tree requiring 3 colors for any best coloring.

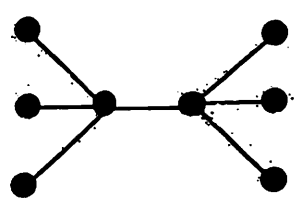

Figure 1.

In fact, it was shown in [1], that the strength of a tree can be arbitrary large. Similar fact is true for maximal outerplanar and maximal planar graphs [5], [6].

The problem of finding chromatic sum is NP-complete in general ([4]), it is NP-complete even for interval graphs [2]. However there is a linear algorithm for finding chromatic sum for trees [4]. The purpose of this paper is to present

polynomial algorithms for computing the chromatic sum for two new families of graphs, unicyclic graphs and outerplanar graphs.

2. Unicyclic Graphs.

The algorithm for finding chromatic sum for unicyclic graphs will use the known [4] linear algorithm for finding chromatic sum for trees (actually a small modification of it).

We will recall now the algorithm for trees from [4]. Given a tree T we root it at an arbitrary vertex and label vertices using preorder traversal. At each vertex v we keep record of the following set of data:

- Minsum(v) = chromatic sum for the subtree of T rooted at v, T_v.
- Rcolor(v) = color of v in a best coloring of T_v.
- Delta(v) = the difference between the chromatic sum of T_v and the minimum sum of colors if we change Rcolor(v) to some other color, denoted Ncolor(v).
- Ncolor(v) = next best color for v.

Knowing these data for all children of a vertex, the algorithm computes the corresponding entries for the parent note. Thus, after reaching the root r of the tree T, the algorithm finds the chromatic sum of T together with the root color $c(r)$ and the second smallest sum of colors for T with root color different from $c(r)$. It is easy to see that after traversing the tree for the second time in the reverse order, we can reconstruct both best and second best colorings of the whole tree.

We will use two straightforward modifications of the above algorithm.
The first modification, Algorithm A, will produce the chromatic sum for T, here denoted by $\sum_1(G)$, with the color c_1 of the root, the smallest sum of colors $\sum_2(G)$ with the color c_2 of the root different from c_1, and finally the smallest sum of colors $\sum_3(G)$ with the color c_3 of the root different from c_1 and from c_2. Shortly, the output of the algorithm is: $(\sum_1,c_1),(\sum_2,c_2),(\sum_3,c_3)$.

The second modification, Algorithm B, produces the minimum sum of colors F_1 and a second minimal sum of colors F_2 with different root colors and with restriction that a specified color is forbidden at a specified vertex z other than the root.

Obviously, both Algorithm A and Algorithm B are linear.

Theorem 1.
There exist a linear algorithm for finding the chromatic sum of a unicyclic graph.

Proof. Let G be a unicyclic graph. Select a vertex, say x, on a unique cycle of G. Let y and z be its neighbors on the cycle. After deleting two edges xy and xz from G, we obtain a disconnected graph with two components, trees T_x and T_y. We root the tree T_x at vertex x and T_y at y (see Figure 2).

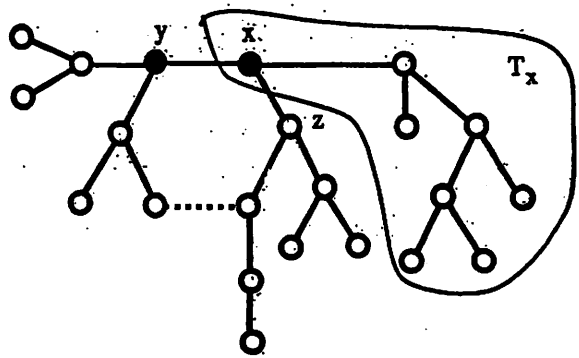

Figure 2.

First we apply Algorithm A to the tree T_x. Its output will be $(X_1,c_1),(X_2,c_2),(X_3,c_3)$, where X_1, X_2, and X_3 are best, second best and third best sums of colors for T_x with corresponding colors of root x, namely c_1,c_2, and c_3, where $c_1 \neq c_2, c_3 \neq c_2$, and $c_3 \neq c_1$. In order to avoid a color conflict on vertices z, x and y in G, we run three times the modified Algorithm B for T_y. First we forbid the color c_1 to appear at vertex z, and we obtain the following information: $\left(Y_1',b_1'\right)$ and $\left(Y_2',b_2'\right)$, where Y_1' is the smallest sum of colors for T_y, b_1' is the color of the root y and similarly Y_2',b_2' is the second smallest sum of colors for T_y, the color of y, respectively. For the second time we run Algorithm B with color c_2 forbidden on vertex z obtaining the following data: $\left(Y_1'',b_1''\right)$, $\left(Y_2'',b_2''\right)$. And finally we run Algorithm B with color c_3 forbidden on vertex z obtaining: $\left(Y_1''',b_1'''\right)$, $\left(Y_2''',b_2'''\right)$.

Let us define three values:

$$K=\begin{cases} X_1+Y_1', \text{ if } c_1 \neq b_1 \\ X_1+Y_2', \text{ otherwise} \end{cases}$$

$$L=\begin{cases} X_2+Y_1'', \text{ if } c_2 \neq b_1'' \\ X_2+Y_2'', \text{ otherwise} \end{cases}, \text{ and}$$

$$M=\begin{cases} X_3+Y_1''', \text{ if } c_3 \neq b_1''' \\ X_3+Y_2''', \text{ otherwise} \end{cases}.$$

We will prove that $\sum(G)=\min\{K,L,M\}$.

Consider any best coloring c of G with sum of colors $\sum(G)$. When restricted to tree T_x, this coloring has sum of colors equal to X_1, X_2, or X_3. This follows from the fact that at most two different colors can occur on vertices y and z, and at least one of the colors among c_1,c_2,c_3 is available for vertex x.

Therefore, it is sufficient to use no colorings on T_x other than those produced by Algorithm *A*. If the coloring (X_1,c_1) is used for T_x, then for a proper coloring of T_y (the rest of *G*) we cannot use color c_1 on z. The minimum sum of colors for the whole graph G is given by K. If the coloring (X_2,c_2) is used for T_x, then color c_2 cannot be used on z and the minimum sum of colors for the whole graph G is L. And finally, with coloring (X_3,c_3) on T_x, the minimum sum of colors for G is M. ⊗

Comment. It can be shown that if the coloring (X_3,c_3) has to be used on T_x, then the corresponding coloring on T_y is in fact a best coloring of this tree with no restrictions. However this observation does not simplify the algorithm.

3. Outerplanar Graphs.

The construction of the algorithm for finding chromatic sum for outerplanar graphs will be done in three steps. First we will consider maximal outerplanar graphs, then 2-connected outerplanar graphs, and finally all outerplanar graphs.

An outerplanar graph is such a graph which has a planar imbedding with all its vertices lying on the boundary of the exterior region. A maximal outerplanar graph is such an outerplanar graph that addition of any edge to it destroys the outerplanar property. It is well known that all interior regions of maximal outerplanar graphs are triangles. Given a maximal outerplanar graph G embedded in the plane, we select a vertex for every interior region (a triangle) of *G*. Two such vertices are adjacent if the corresponding regions share an edge. A graph constructed in this way is called a spine of the graph *G*. It is known that the spine of a maximal outerplanar graph is a tree (see Figure 3).

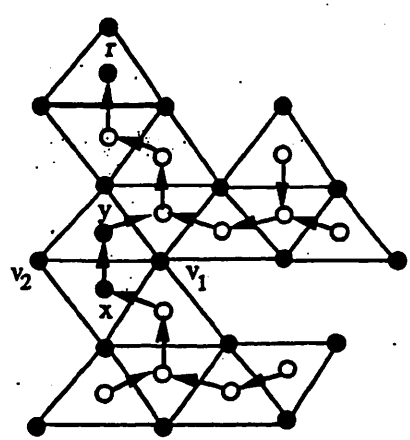

Figure 3.

Theorem 2.
There is a polynomial algorithm for finding chromatic sum for a maximal outerplanar graph.

Proof. Let G be a maximal outerplanar graph with the spine S. Select a leaf of the spine as the root r, and consider S to be a directed rooted tree with all arcs directed towards r. The Algorithm M for finding the chromatic sum of G will examine every triangle of G in order given by a preorder traversal of spine S. If x, $x \neq r$, is a vertex of the spine adjacent to the vertex y and if triangular regions of G corresponding to x and y share the edge v_1v_2, then Algorithm M computes all minimal sums of colors for the graph induced by triangles visited so far for all possible pairs of colors assigned to vertices v_1 and v_2.
The possible pairs of colors for v_1 and v_2 are

$$(c(v_1),c(v_2)), \text{ where } 1 \le c(v_i) \le \deg(v_i)+1, \text{ i}=1,2, \text{ and } c(v_1) \neq c(v_2).$$

We consider three cases depending on in-degree of vertex x.

Case 1. The vertex x has in-degree 0 or, equivalently, x is a leaf of S.
Then $\sum(c(v_1),c(v_2)) = 6$, if $(c(v_1),c(v_2)) = (1,2),(1,3),(2,1)$, or $(3,1)$, $\sum(c(v_1),c(v_2)) = c(v_1)+c(v_2)+2$, if one of the colors is 1 and the other is at least 4. Also, $\sum(c(v_1),c(v_2)) = c(v_1)+c(v_2)+1$, if none of the colors $c(v_1),c(v_2)$ is 1.

Case 2. The vertex x has in-degree 1.
Let H_x be the graph induced by triangles visited so far (predecessors of x). Let z be the immediate predecessor of x and v_1w the edge common to the triangles corresponding to x and z (see Figure 4).

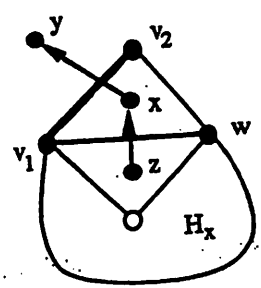

Figure 4.

Then $\sum(c(v_1),c(v_2)) = \min_k \sum_{H_x}(c(v_1),k)+j$, where the minimum is taken over all integers k such that $1 \le k \le \deg w+1$ and $k \neq c(v_1),c(v_2)$.

Case 3. The vertex x has in-degree 2.
Let l and r be immediate predecessors of x. Denote the graph induced by triangles corresponding to l and all its predecessors by H_x^l. Denote the graph induced by triangles corresponding to r and all its predecessors by H_x^r (see Figure 5).

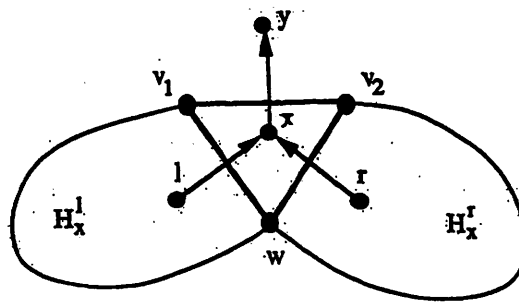

Figure 5.

Then $\sum(c(v_1),c(v_2)) = \min_k\left(\sum_{H_x^l}(c(v_1),k) + \sum_{H_x^r}(k,c(v_2)) - k\right)$, where the minimum is taken over all integers k such that $1 \le k \le \deg w + 1$ and $k \ne c(v_1)$, $c(v_2)$.

Finally we reach the root r whose only predecessor is z and v_1v_2 is the edge shared by the triangles corresponding to z and r. Let w be the third vertex of the triangle corresponding to r (see Figure 6).

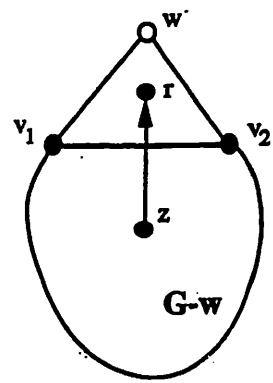

Figure 6.

We have already computed all minimal sums of colors for the graph G-w with fixed colors on vertices v_1 and v_2. Let $\sum_G(i)$ denote the minimum sum of colors for the whole graph G with the color i on w. Then

$\sum_G(i) = \sum_{G-w}(c(v_1),c(v_2)) + i$, where the minimum is taken over all pairs $(c(v_1),c(v_2))$ such that $c(v_1) \neq i$, $c(v_2) \neq i$, and $c(v_1) \neq c(v_2)$.
Finally, $\sum(G) = \min_{i=1,2,3} \sum_G(i)$.

If d_i are degrees of vertices of G, then the number of operations Algorithm M performs is equal to

$$(d_{i_1}+1)(d_{j_1}+1)(d_{k_1}+1)+(d_{i_2}+1)(d_{j_2}+1)(d_{k_2}+1)+\ldots+(d_{i_{n-2}}+1)(d_{j_{n-2}}+1)(d_{k_{n-2}}+1)$$

where d_i, d_j, d_k are degrees of vertices of triangles (n - 2 of them if the order of G is n).
Since there is always at least one new vertex in the next product, this sum is bounded by $(\Delta+1)(\Delta+1)(d_1+d_2+\ldots+d_{n-2})$ where Δ is maximal degree of G, and, therefore, is of order n^3. ⊗

If an outerplanar graph is 2-connected, then its interior regions are n-gons with $n \geq 3$. The spine of such a graph is also a tree but degrees of vertices in this tree might be larger than 3.

Theorem 3.
There exists a polynomial time algorithm for finding the chromatic sum for a 2-connected outerplanar graph.

Proof. Let G be a 2-connected outerplanar graph with the spine S. Select a leaf on S to be its root r and direct all edges of S towards its root. The Algorithm 2C for finding the chromatic sum for G will examine all interior regions of G in order given by a preorder traversal of S. Every vertex x of S, $x \neq r$, is adjacent to exactly one vertex y of S. Suppose that the region corresponding to x is an n-gon with the vertices $v_1, v_2, \ldots, v_n$ and also v_1v_n is the edge of the region corresponding to vertex y (see Figure 7).

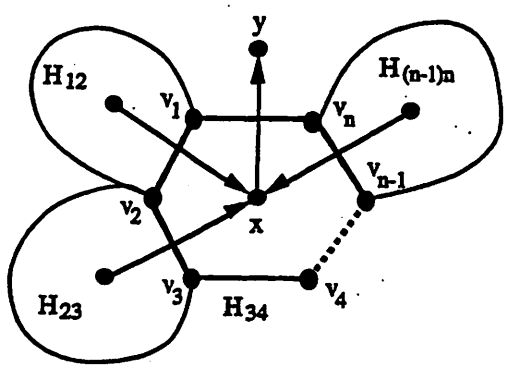

Figure 7.

For every edge v_iv_{i+1}, i=1,2,...,n-1, let $H_{i,i+1}$ denote the following graph:

(a) if v_iv_{i+1} is on the boundary of the exterior region, then $H_{i,i+1}$ is the graph induced by this single edge;

(b) if v_iv_{i+1} is on the boundary of the interior region corresponding to vertex z (a predecessor of x), then $H_{i,i+1}$ is the graph induced by the triangles corresponding to z and all its predecessors.

The Algorithm C is a modification of Algorithm M. If a triangular region is encountered, Algorithm C behaves exactly like Algorithm M. If an n-gon $v_1,v_2,...,v_n$ is visited with $n \geq 4$, the Algorithm C finds minimal sums of colors for the following graphs:

- H_{12} for every pair of fixed colors on vertices v_1 and v_2,
- $H_{12} \cup H_{23}$ for every pair of fixed colors on vertices v_1 and v_3,
- $H_{12} \cup H_{23} \cup H_{34}$ for every pair of fixed colors on vertices v_1 and v_4,

  and finally
- $H_{12} \cup H_{23} \cup ... \cup H_{n-1n}$ for every pair of fixed colors on vertices v_1 and v_n.

The last graph, $H_{12} \cup H_{23} \cup ... \cup H_{n-1n}$, is the graph induced by vertices of all regions corresponding to predecessors of y.

The complexity of Algorithm C is also cubic. ⊗

Theorem 4.
There exists a polynomial time algorithm for finding the chromatic sum for an outerplanar graph.

Proof. Let G be an outerplanar graph that is not 2-connected. Consider blocks of G (maximal 2-connected subgraphs). Each block is either a 2-connected outerplannar graph or K_2. An end-block is a block containing only one cut-vertex of G. Put all blocks in an order $B_1,B_2,...,B_k$ such that all end-blocks precede other blocks. After all end-blocks are deleted from G, the remaining blocks are divided into two groups: new end-blocks and the other blocks. New end-blocks precede other blocks. This procedure is continued until all blocks are put in order. For example, if an outerplanar graph has a block structure depicted in Figure 8, then one of the possible orderings of blocks is given.

We use Algorithm C for each end-block in such a way that all minimal sums of colors are produced with a fixed color i on its unique cut-vertex v, $1 \leq i \leq \deg v+1$. Then we examine all remaining blocks in the selected order, every time finishing with finding all minimal sums of colors with fixed color i on the new unique cut-vertex. When running Algorithm C for such a block we have to adjust the partial chromatic sum at each vertex that was a previously considered cut-vertex. For example, if Algorithm C is run for block B_6 (Figure 8) and vertex u is encountered the first time, we find color i for which the subgraph induced by block B_4 and all vertices of B_6 visited so far has minimal sum of colors. Then we continue to examine the rest of block B_6 until we reach the cut-vertex v.

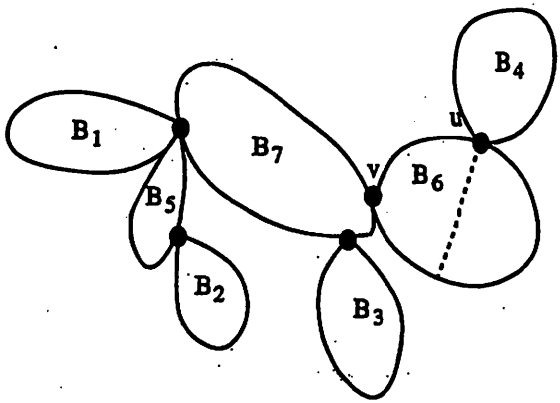

Figure 8.

We will remember for this vertex all minimal sums of colors for the graph induced by blocks B_4 and B_6 for each color i on vertex v, $1 \le i \le \deg v+1$.

As for previous algorithms, the complexity of this one is cubic. ⊗

References

[1] P. Erdös, E. Kubicka, and A. Schwenk, Graphs that Require Many Colors to Achieve their Chromatic Sum, *Congressus Numerantium* 71 (1990), 17-28.

[2] L.G. Kroon, A. Sen, H. Deng, and A. Roy, The Optimal Cost Chromatic Partition Problem, *Journal of Algorithms* 34 (2000), 54-89.

[3] E. Kubicka, The chromatic sum and efficient tree algorithms, Ph. D. Dissertation, Western Michigan University, 1989.

[4] E. Kubicka, and A. Schwenk, An Introduction to Chromatic Sums, *Proc. ACM 1989, Computer Science Conference*, 39-45

[5] J. Mitchem, and P. Morris, On the Cost Chromatic Number of Graphs, *Discrete Mathematics* 172 (1997), 201-211.

[6] J. Mitchem, P. Morris, and E. Schmeichel, On the Cost Chromatic Number of Outerplanar, Planar, and Line Graphs, *Dicussiones Mathematicae Graph Theory* 17 (1997), 1-13.