AN INTRODUCTION TO CHROMATIC SUMS*

Ewa Kubicka and Allen J. Schwenk
Department of Mathematics & Statistics
Western Michigan University

Abstract

We introduce the new concept of the chromatic sum of a graph G, the smallest possible total among all proper colorings of G using natural numbers. We show that computing the chromatic sum for arbitrary graphs is an NP-complete problem. Indeed, a polynomial algorithm for the chromatic sum would be easily modified to compute the chromatic number. Even for trees the chromatic sum is far from trivial. We construct a family of trees to demonstrate that for each k, some trees need k colors to achieve the minimum sum. In fact, we prove that our family gives the smallest trees with this property. Moreover, we show that asymptotically, for each value of k, almost all trees require more than k colors. Finally, we present a linear algorithm for computing the chromatic sum of an arbitrary tree.

1. THE CHROMATIC SUM

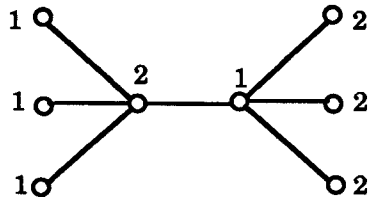
In this article we wish to introduce a new variation on the chromatic number of a graph. Instead of minimizing the number of colors in a proper coloring, we choose instead to minimize the sum of these colors. More formally:

* Research supported in part by the Office of Naval Research Contract N00014-86-K-0566.

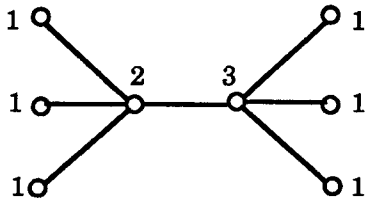
DEFINITION 1. Let $G = (V,E)$ be a graph with vertex set V and edge set E. We define the *chromatic sum* of graph G, $\Sigma(G)$, to be the minimum sum $\sum_{v \in V} c(v)$ taken over all proper colorings c of graph G using natural numbers; i.e. $c: V \to \mathbf{N}$ and $c(v) \neq c(w)$ whenever $(v,w) \in E$.

DEFINITION 2. A proper coloring c of a graph G is called a *best coloring* of G whenever $\sum_{v \in V} c(v) = \Sigma(G)$.

Example 1. Consider the tree T and two proper colorings c and c' shown in Figure 1. We see that $\Sigma(T) = 11$ and c' is the unique best coloring. This example illustrates the surprising feature that the minimum total may very well be achieved by using more than the minimum number of colors.



T with coloring c, $\sum_{v \in V} c(v) = 12$



T with coloring c', $\sum_{v \in V} c'(v) = 11$

Figure 1. Two proper colorings of the tree T.

THEOREM 1. The following decision problem Π is NP-complete :

Generic instance : Graph $G = (V,E)$, positive integer K.

Question : Is there a proper coloring c of graph G such that $\sum_{v \in V} c(v) \leq K$?

Proof: It is easy to see that $\Pi \in NP$ since a nondeterministic algorithm need only guess a color assignment to the vertices of G and check (in polynomial time) whether that color assignment is proper and whether its sum does not exceed K.

Consider another decision problem Π' :

Generic instance : Graph $G = (V,E)$, positive integer $K \leq |V|$.

Question : Is G K-colorable ?

Since Π' is known to be NP-complete, we shall show that Π is also by transforming Π' into problem Π. Let an arbitrary instance of Π' be given by a graph $G = (V,E)$ and a positive integer $K \leq |V|$. We must construct a graph $G' = (V',E')$ and a positive integer K' such that there exists a proper coloring c' of G' with the property : $\sum_{v \in V} c'(v) \leq K'$ if and only if G is K-colorable.

Let graph $G' = G \times K_K$ be the cartesian product of gragh G with a complete graph with K vertices. That is, if $V = \{v_1, v_2, ..., v_n\}$ then $V' = \{v_j^i \mid 1 \leq i \leq K, 1 \leq j \leq n\}$ and $(v_j^i, v_s^r) \in E'$ whenever either ($j=s$ and $i \neq r$) or ($i=r$ and $(v_j, v_s) \in E$). Also let $K' = n \cdot \frac{K(K+1)}{2}$. Assume that G is K-colorable and let $c: V \rightarrow \{1,2,...,K\}$ be a proper coloring of G. We can envision graph $G' = (V',E')$ in the following way: $V' = \{v_1^1, ..., v_n^1, ..., v_1^i, ..., v_n^i, ..., v_1^K, ..., v_n^K\}$ where vertices $v_1^i, ..., v_n^i$ form the i^{th} copy of G for $1 \leq i \leq K$ and vertices $v_j^1, ..., v_j^K$ form the j-th copy of K_K for $1 \leq j \leq n$.

Example 2.

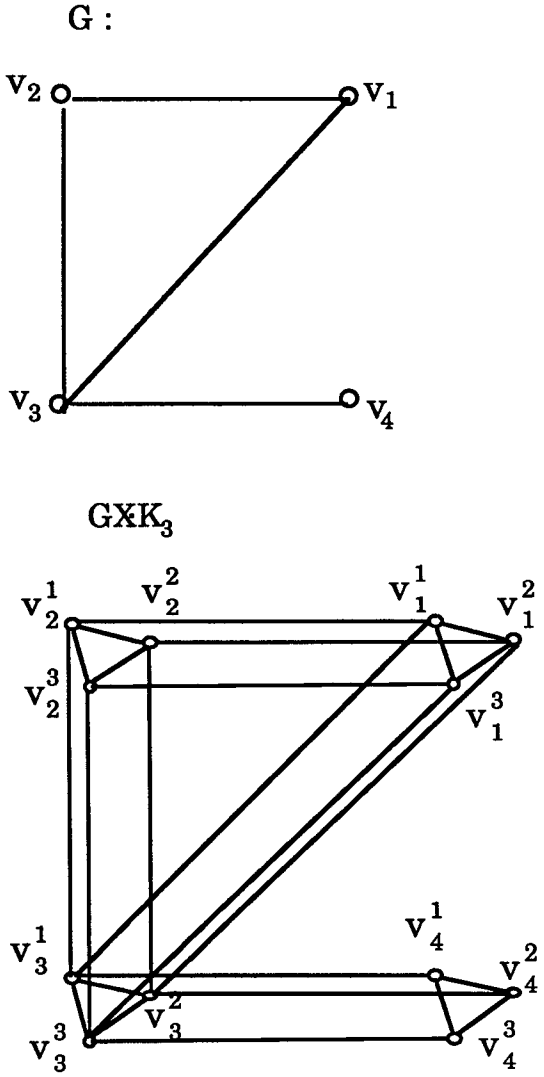


Figure 2. The construction of $G' = G \times K_K$.

We can extend coloring c to a proper coloring c' of graph G' in the following way : the first copy of G has the same coloring scheme as G , in the second copy colors are shifted by 1 in a cyclic manner and so on, i.e. :

$c'(v_j^i) = (c(v_j) + i - 2)_{\text{mod } K} + 1$. Thus $G' = (V',E')$ is K–colorable and $\sum_{v \in V'} c'(v) = n \cdot \frac{K(K+1)}{2} = K'$.

Conversely, suppose that there exists a proper coloring c' of graph G' such that $\sum_{v \in V} c'(v) \le K'$. Since each copy of K in G' has to use K different colors, the smallest possible sum of colors, $\frac{K(K+1)}{2}$, can be obtained only if we use colors 1 through K. In order for any proper coloring of G' to have its sum of colors bounded above by K', we must use colors 1 through K at each copy of K_K in G'. Therefore G' is K–colorable and so is its subgraph G.

Notice that $|V'| = K \cdot n \le n^2 = |V|^2$ and $|E'| = K \cdot |E| + n \cdot \frac{K(K-1)}{2} < n^3$. Thus the instance (G',K') can be constructed from the instance (G,K) in polynomial time.

We have shown that the NP-complete problem Π' can be polynomially transformed into problem Π. Therefore Π is also NP-complete. ■

COROLLARY 1. The optimization problem of finding the chromatic sum can be solved in polynomial time if and only if P = NP.

Proof : Any algorithm for finding the chromatic sum can be used as an algorithm for solving the decision problem Π. Therefore if the chromatic sum can be always found in polynomial time, $\Pi \in P$, and thus P = NP.

Conversely assume that P = NP which implies the existence of a polynomial algorithm A^* for solving Π. For graphs of order n, we know that $n \le \Sigma(G) \le \frac{n(n+1)}{2}$. Thus, by using a binary search procedure, we can find $\Sigma(G)$ by a sequence of at most $\lceil \log_2(\frac{n(n-1)}{2}) \rceil$ calls on A^* giving to A^* as inputs the graph G and different values of K. ■

It is well known that trees are bipartite and thus we can always color them properly using only colors 1 and 2. However Example 1 shows that sometimes we are forced to use additional colors to obtain the chromatic sum. We will construct now, for each k, a tree of smallest order in which color k is forced to appear in every best coloring. We first construct the family of rooted trees T_k^m recursively. We assume that removing the root r leaves a forest in which each new tree is rooted at the vertex that used to be adjacent to r. Let T_1^1 be the rooted tree with one vertex. Tree T_k^m is the unique tree such that

$T_k^m - r = \bigcup_{i=1}^{k-1} (m + k - i)T_i^1$. Examples of this construction are shown in Figure 3.

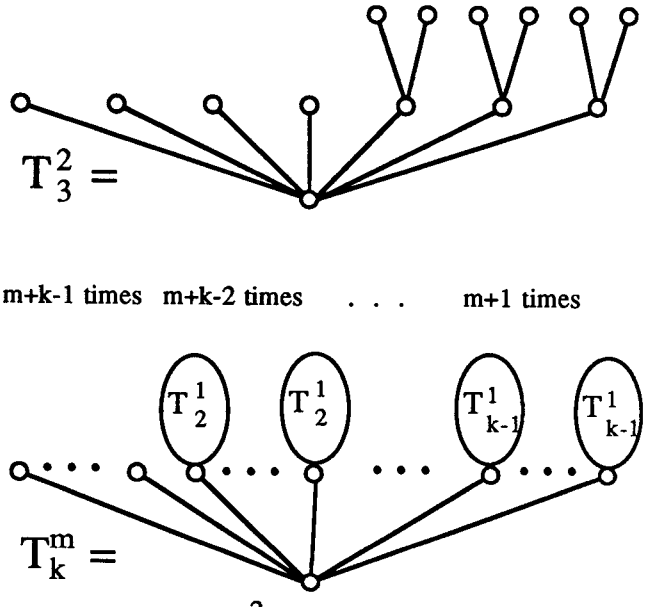


Figure 3. The tree T_3^2 and more generally T_k^m.

LEMMA 1. For $k \geq 2$, the tree T_k^m is the smallest rooted tree for which in every best coloring color k is forced to appear at the root and any change of that color to a lower color must increase the sum of colors by at least m.

Proof : (by induction on k)

(1). It is easy to see that T_2^m has to have color 2 at the root in any best coloring and changing it to 1 costs exactly m. Note that $| T_2^m | = m + 2$. Let T be a smallest rooted tree having that property and let c be its best coloring. Let b denote the number of vertices of color 2 in T. Then T must have at least b + m vertices colored with 1 because otherwise interchanging colors 1 and 2 in T would give a proper coloring of T with the root colored with 1 and the increase of the sum of colors by less then m. Thus $| T | \geq m+2$.

(2). Assume that the lemma is true for all $i \leq k$. Let T(i,m) denote a tree of the smallest order where color i is forced to appear at the root and it's change costs at least m. Consider T(k+1,m) and it's best coloring c. We will show that T(k+1,m) must be T_{k+1}^m. After removing the root from T(k+1,m) we are left with a forest of rooted trees. Let $F_1(k+1,m)$ denote the subforest containing all those trees with roots colored with 1. $F_1(k+1,m)$ is a smallest forest of the property that changing the color 1 at the roots to any other color costs at least k+1–1+m. Therefore, for $1 < k$, $F_1(k+1,m) = F_1(k,m+1)$ which is the similar forest for T_k^{m+1}. Thus using the inductive hypothesis we have that $F_1(k+1,m) = (k+1-1+m)\, T_1^1$. Now the only thing left is to show that $F_k(k+1,m) = (m+1)\, T_k^1$. The forest $(m+1)\, T_k^1$ has all the roots colored with k and any change of that color costs at least m+1. Consider the subtree T_0 of T(k+1,m) consisting of the largest connected component of T(k+1,m) containing the root and only vertices colored with k+1 and k. Among all possible T(k+1,m) let's select that one which has the fewest number of vertices colored with k occuring in the corresponding subtree T_0. Let b denote the number of vertices of T_0 colored with k+1. Then, similarly as in (1) , we can show that T_0 must have at least m + b vertices colored with k, let's call them $v_1, v_2, ..., v_r$ ($r \geq m + b$). Let S_i denote the subtree of T(k+1,m) with root v_i which is formed after deleting all edges between v_i and any vertex colored k+1. We claim that there must be at least m+1 of these subtrees S_i for which $| S_i | \geq | T_k^1 |$. Assume this is not true. Recall that T_k^1 is the smallest rooted tree in which color k is forced at the root. Thus $| S_i | < | T_k^1 |$ means that S_i has a best coloring c' which uses a different color at the root. If this color is smaller then k we can change our best coloring of T(k+1,m) by just using c' on S_i obtaining a best coloring of T(k+1,m) with smaller number of vertices colored with k in T_0 – a contradiction. If, on the other hand, in all S_i's for which $| S_i | < | T_k^1 |$ we could change root color from k to k+1 at no cost then swapping colors k+1 and k throughout T_0 would produce a best coloring of T(k+1,m) with root colored by k and with the increase of sum of colors less then m, another contradiction. Therefore

$$| F_k(k+1,m) | \geq | (m+1)T_k^1 |. \quad \blacksquare$$

Denote now by T_k the unrooted tree formed by adding an edge between the roots of two copies of T^2_{k-1}.

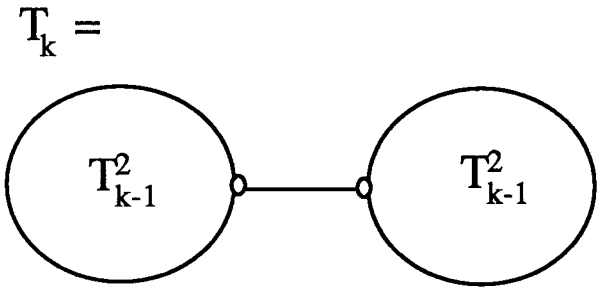


Figure 4. The smallest tree requiring k colors in a best coloring.

THEOREM 2. T_k is the smallest tree in which color k is needed in every best coloring. For $k \geq 2$, the order of T_k is given by

$$|T_k| = \frac{1}{\sqrt{2}}[(2+\sqrt{2})^{k-1} - (2-\sqrt{2})^{k-1}].$$

Proof : Let F_k be a smallest tree that requires k colors in every best coloring, and let c be its best coloring. Locate an edge $e = (v_1, v_2)$ joining a vertex of color k to one of color k - 1. Removing edge e leaves two trees S_1 and S_2 such that every best coloring of T_i must color the roots v_i with k-1 and changing that color costs at least 2. Thus $S_1 = S_2 = T^2_{k-1}$.

Let $t^m_k = |T^m_k|$. The equation defining these trees immediately gives the recurrence

$$t^m_k = 1 + \sum_{i=1}^{k-1} (k+m-i)t^1_i.$$

We first show by induction that the subsequence with $m = 1$ satisfies the simpler recurrence $t^1_k = 4t^1_{k-1} - 2t^1_{k-2}$.

The induction is based on the instance $k = 3$ verified by the values $t^1_1 = 1, t^1_2 = 3$, and $t^1_3 = 10$. Next assume that $k \geq 4$ and that the recurrence has already been proved for all $3 \leq i < k$. Note that

$$t^1_k - 4t^1_{k-1} + 2t^1_{k-2} =$$

$$1 + \sum_{i=1}^{k-1}(k+1-i)t^1_i - 4 - 4\sum_{i=1}^{k-2}(k-i)t^1_i + 2 + 2\sum_{i=1}^{k-3}(k-1-i)t^1_i.$$

Adjusting indices in the second and third summations and collecting the isolated terms gives

$$t^1_k - 4t^1_{k-1} + 2t^1_{k-2} =$$

$$-1 + \sum_{i=1}^{k-1}(k+1-i)t^1_i - 4\sum_{i=2}^{k-1}(k+1-i)t^1_{i-1} + 2\sum_{i=3}^{k-1}(k+1-i)t^1_{i-2}.$$

The induction hypothesis guarantees that all the terms with $i \geq 3$ cancell leaving

$$t^1_k - 4t^1_{k-1} + 2t^1_{k-2} = -1 + k\,t^1_1 + (k-1)\,t^1_2 - 4(k-1)\,t^1_1 = 0.$$

But the same proof works for t^m_k provided that $m \geq k + 2$.

This time the remaining uncancelled terms are

$$t^m_k - 4t^m_{k-1} + 2t^m_{k-2} =$$

$$-1 + (k+m-1)t^1_1 + (k+m-2)\,t^1_2 - 4\,(k+m-2)\,t^1_1 = 0.$$

We specifically need the solution when $m = 2$. Given the starting values $t^2_2 = 4$ and $t^2_3 = 14$, it is routine to solve the recurrence for $k \geq 2$ to find that

$$t^2_k = \frac{1}{2\sqrt{2}}[(2+\sqrt{2})^k - (2-\sqrt{2})^k].$$

Of course $|T_k| = 2t^2_{k-1}$. ■

COROLLARY 2. For every positive integer k, almost every tree requires at least k colors in its best coloring; i.e.: $\frac{t_n(k)}{t_n} \to 1$ as $n \to \infty$, where $t_n(k)$ is the number of trees of order n which have a best coloring using less then k colors and t_n is the number of all trees of order n.

Proof. This is an immediate consequence of the theorem due to A. Schwenk [4] stating that for every rooted tree L, almost every tree has L as a limb. We simply set L to be T_k^1. ■

THEOREM 3. Let T be a tree of order $n \geq 2$. Then $n + 1 \leq \Sigma(T) \leq \lfloor 1.5n \rfloor$. Moreover, for every k between n+1 and $\lfloor 1.5n \rfloor$, there is a tree T of order n such that $\Sigma(T) = k$.

Proof. We have to use at least 2 colors and thus $\Sigma(T) \geq n+1$. For a star K(1,n-1) this lower bound is attained. A tree is a bipartite graph and therefore we can always color it properly using colors 1 and 2 , coloring all vertices in the possibly bigger partite set with color 1 and all vertices in the other partite set with color 2. Thus $\Sigma(T) \leq \lceil \frac{n}{2} \rceil + 2\lfloor \frac{n}{2} \rfloor = \lfloor 1.5n \rfloor$. The upper bound is attained by a path P_n.

Define B(m,b) to be a "broom - like" tree consisting of a vertex adjacent to m end-vertices and a path of length b - 1. This family is illustrated in Figure 5.

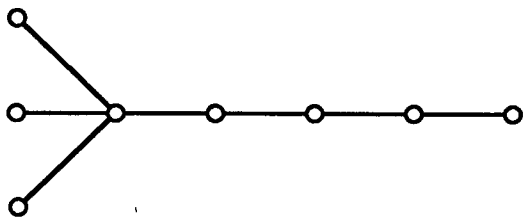


Figure 5. The broom B(3,5).

Let $n + 1 < k < \lfloor 1.5n \rfloor$. We can represent k as $k = n + 1 + b$, for some b. It is easily verified that

$$\Sigma(B(n - 2b - 1, 2b + 1)) = k. \blacksquare$$

2. LINEAR ALGORITHM FOR FINDING THE CHROMATIC SUM FOR A TREE OF ORDER N.

We assume that the program has the information about the structure of the examined tree, i.e. at each vertex it is known how many sons this vertex has and there is an access to the first son and to the next brother (sibling). The vertices are listed in a preorder traversal, that is the root of the tree is first and if we go from the last vertex to the first we encounter all the children vertices before we examine the parent vertex.

For each vertex I of the tree TREE[1] the program keeps the following information :

TREE[I].MINSUM – chromatic sum for the tree TREE[I] for which I is the root,

TREE[I].RCOLOR – color of the root of TREE[I] - vertex I in the best coloring produced so far by the algorithm,

TREE[I].DELTA – how much the sum of colors in the tree TREE[I] increases if we change the color of the root I,

TREE[I].NCOLOR – next best color for I,

TREE[I].NOSONS – number of sons of vertex I,

TREE[I].SON – first son of vertex I,

TREE[I].BROTHER – next brother of vertex I,

COLORADD[K] - the increase in the total of the chromatic sums TREE[I].MINSUM for all the sons I of a given vertex v when we insist on coloring vertex v with color K.

The procedure starts from the leaves and goes up to the root. In steps 4 – 9 the values MINSUM, RCOLOR, DELTA, NCOLOR are easily determined for leaves. In steps 11 – 48 those values are determined for a vertex I based on already known MINSUMs, RCOLORs, DELTAs, and NCOLORs for all sons of vertex I. TREE[1].MINSUM gives the chromatic sum of the tree being examined.

```
1. For  I=N  downto  1  do
2.   if TREE[I].NOSONS=0
3.     then
4.       begin
5.         TREE[I].MINSUM=1
6.         TREE[I].RCOLOR=1
7.         TREE[I].DELTA=1
8.         TREE[I].NCOLOR=2
9.       end
10.    else
11.       begin
12.       SON=TREE[I].SON
13.       MINTOTAL=0
14.      for K=1 to TREE[I].NOSONS+2   do
15.         COLORADD[K]=K
16.      for K=1  to TREE[I].NOSONS   do
17.         begin
18.         MINTOTAL=MINTOTAL+TREE[SON].MINSUM
19.         COLORADD[TREE[SON].RCOLOR]=
               COLORADD[TREE[SON].RCOLOR] +
               TREE[SON].DELTA
20.         SON=TREE[SON].BROTHER
21.         end
22.      SUM1=∞
23.      SUM2=∞
24.      for  K=1  to  TREE[I].NOSONS+2   do
25.           begin
26.           VALUE=COLORADD[K]
27.           if (VALUE<SUM1)
28.               then
29.                 begin
30.                   COLOR2=COLOR1
31.                   SUM2=SUM1
32.                   COLOR1=K
33.                   SUM1=VALUE
34.                 end
35.               else
36.               if  (VALUE<SUM2)
37.                   then
38.                     begin
39.                       COLOR2=K
40.                       SUM2=VALUE
41.                     end
42.          end
43.     TREE[I].MINSUM=SUM1+MINTOTAL
44.     TREE[I].RCOLOR=COLOR1
45.     TREE[I].DELTA=SUM2-SUM1
46.     TREE[I].NCOLOR=COLOR2
47.   end
```

ANALYSIS OF COMPLEXITY

We will sum up the number of instructions executed for each vertex I in a given tree of order N. The total number of instructions is at most

$$\sum_{I=1}^{N} (8+(TREE[I]+2)\cdot 10) \;=\; 28N + 10(N-1).$$

Therefore the complexity of the algotithm is O(N). ■

NOTE 1. It is also possible to recover the actual best coloring produced by the algorithm at a cost of O(N).

REFERENCES:

[1] G. Chartrand and L. Lesniak, *Graphs & Digraphs 2nd Edition*, Wadsworth & Brooks/Cole, Monterey CA 1986.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability*, 1979.

[3] R. M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computation, Plenum Press, 1972.

[4] A. Schwenk, *Almost all trees are cospectral*, New Directions in the Theory of Graphs, (F. Harary, ed.) Academic Press, New York, 1973, 155-163.