Approximation Algorithms for the Max-coloring Problem*

Sriram V. Pemmaraju and Rajiv Raman

The University of Iowa, Iowa City, IA 52242, USA
{sriram, rraman}@cs.uiowa.edu

Abstract. Given a graph $G = (V, E)$ and positive integral vertex weights $w : V \to \mathbf{N}$, the *max-coloring problem* seeks to find a proper vertex coloring of G whose color classes $C_1, C_2, \ldots, C_k$, minimize $\sum_{i=1}^{k} max_{v \in C_i} w(v)$. The problem arises in scheduling conflicting jobs in batches and in minimizing buffer size in dedicated memory managers.

In this paper we present three approximation algorithms and one inapproximability result for the max-coloring problem. We show that if for a class of graphs $\mathcal{G}$, the classical problem of finding a proper vertex coloring with fewest colors has a c-approximation, then for that class $\mathcal{G}$ of graphs, max-coloring has a $4c$-approximation algorithm. As a consequence, we obtain a 4-approximation algorithm to solve max-coloring on perfect graphs, and well-known subclasses such as chordal graphs, and permutation graphs. We also obtain constant-factor algorithms for max-coloring on classes of graphs such as circle graphs, circular arc graphs, and unit disk graphs, which are not perfect, but do have a constant-factor approximation for the usual coloring problem. As far as we know, these are the first constant-factor algorithms for all of these classes of graphs. For bipartite graphs we present an approximation algorithm and a matching inapproximability result. Our approximation algorithm returns a coloring whose weight is within $\frac{8}{7}$ times the optimal. We then show that for any $\epsilon > 0$, it is impossible to approximate max-coloring on bipartite graphs to within a factor of $(\frac{8}{7} - \epsilon)$ unless $P = NP$. Thus our approximation algorithm yields an optimum approximation factor. Finally, we also present an exact sub-exponential algorithm and a PTAS for max-coloring on trees.

1 Introduction

The *max-coloring problem* takes as input a vertex-weighted graph $G = (V, E)$ with weight function $w : V \to \mathbf{N}$. The problem requires that we find a proper vertex coloring of G whose color classes $C_1, C_2, \ldots, C_k$, minimize the sum of the weights of the heaviest vertices in the color classes, that is, $\sum_{i=1}^{k} max_{v \in C_i} w(v)$. When all the weights are one, this problem reduces to the classical problem of

* This research is partially supported by the National Science Foundation Grant DMS-0213305.

L. Caires et al. (Eds.): ICALP 2005, LNCS 3580, pp. 1064–1075, 2005.

finding a proper vertex coloring of a graph using fewest possible colors. For any color class C of G, we will use $weight(C)$ to denote $\max\{w(v) \mid v \in C\}$. The *weight* of a coloring $C_1, C_2, \ldots, C_k$ is then $\sum_{i=1}^{k} weight(C_i)$.

The max-coloring problem arises in two distinct applications. In one application the max-coloring problem models the problem of minimizing the total buffer size needed for memory management in wireless protocol stacks like GPRS or 3G [7] and in digital signal processing applications [2]. In general, programs that run with stringent memory or timing constraints use a dedicated memory manager that provides better performance than the general purpose memory management of the operating system. The most commonly used memory manager design for this purpose is the *segregated buffer pool*. The problem of minimizing the total size of the buffer pool corresponds to the max-coloring problem.

A second application of max-coloring arises in the scheduling of jobs with conflicts in a multiprocessor environment. In systems in which jobs require exclusive access to certain resources, a fundamental problem is of scheduling jobs onto processors such that jobs requiring access to the same resource are not scheduled together. The problem of scheduling jobs in conflict to processors can be modeled as a graph coloring problem. When jobs have different processing times, this is modeled as a generalized coloring problem on vertex weighted graphs. One such generalization that models the problem of scheduling conflicting jobs in batches to minimize the *makespan* or the time to complete all the jobs in the system corresponds to the max-coloring problem.

Our Results. Although graph coloring is hopelessly hard to approximate on general graphs, the underlying conflict graphs that arise in applications have more structure, and this structure can be exploited to obtain efficient exact or approximation algorithms for max-coloring. However, the max-coloring problem is hard even on instances where the coloring problem can be solved in polynomial time. In [7], the authors prove that max-coloring is NP-hard on interval graphs, even though there is a simple greedy algorithm for the usual coloring problem [1]. [7] also presents a 2-approximation for the max-coloring problem on interval graphs. For other classes of graphs, very little seems to be known about how to solve the max-coloring problem efficiently, either exactly or approximately. In this paper we present three approximation algorithms and one inapproximability result. We show that for any hereditary[1] class of graphs $\mathcal{G}$, if the usual vertex coloring problem has a c-approximation, then max-coloring has a $4c$-approximation on $\mathcal{G}$. One implication is that there is a 4-approximation algorithm to solve max-coloring on perfect graphs. For bipartite graphs we present an approximation algorithm and a matching inapproximability result. Our approximation algorithm always returns a coloring whose weight is within $\frac{8}{7}$ times the optimal and following this we show that for any $\epsilon > 0$, it is impossible to approximate max-coloring on bipartite graphs to with a factor of $(\frac{8}{7} - \epsilon)$ unless

[1] A class $\mathcal{G}$ of graphs is hereditary, if for any $G \in \mathcal{G}$, every induced subgraph of G is also in $\mathcal{G}$.

$P = NP$. Thus our approximation algorithm yields an optimum approximation factor. Finally, we also present an exact sub-exponential algorithm and a PTAS for trees. The max-coloring problem on trees was also studied by Guan and Zhu [3] where the authors present a polynomial time algorithm that finds an optimal max-coloring of a given tree, that uses exactly r colors for a fixed constant r.

2 Max-coloring Trees

The max-coloring problem has turned out to be surprisingly difficult even for trees. Though we believe that the problem can be solved in polynomial time, the two best algorithms we have are (i) a sub-exponential exact algorithm and (ii) a PTAS. We present these in this section. Our first observation is on the distribution of weights of color classes in an optimal max-coloring of bipartite graphs.

Lemma 1. *Let G be a bipartite graph. Let $\{C_1, C_2, \ldots, C_k\}$ be an optimal max-coloring of G with $w_i = weight(C_i)$ and $w_1 \geq w_2 \geq \cdots \geq w_k$. Then, we have that $w_i \geq \sum_{j=i+1}^{k} w_j$, $i = 1, \cdots, k-1$.*

Proof. If $w_i < \sum_{j=i+1}^{k} w_j$, then the subgraph induced by vertices in $\cup_{j=i}^{k} C_j$ can be colored with two colors with weight at most $2w_i$. This coloring has weight less than the weight of $\{C_1, C_2, \ldots, C_k\}$, a contradiction. □

Corollary 1. *Let G be a bipartite graph. Let $\{C_1, \cdots, C_k\}$ be an optimal max-coloring of G with $w_i = weight(C_i)$ and $w_1 \geq w_2 \geq \cdots \geq w_k$, we have that $\frac{w_i}{2} \geq w_{i+2}$, for $i = 1, \cdots, k-2$, and hence, $w_1 \geq 2^{\lfloor (i-1)/2 \rfloor} \cdot w_i$.*

Since the weights of the color classes decrease rapidly, we can expect that the max-color number of a tree may not be too high. Let $\chi_{mc}(G)$ denote the *max-color number* of a graph G, the minimum number of colors required in a minimum cost max-coloring. We now state three upper bounds on χ_{mc}.

Lemma 2. *Let T be an n-vertex tree with maximum degree Δ. Let W denote the ratio of the weight of the heaviest vertex to the weight of the least heavy vertex. Then, (i) $\chi_{mc}(T) \leq \Delta + 1$, (ii) $\chi_{mc}(T) \leq \lceil \log_2 n \rceil + 1$, and (iii) $\chi_{mc}(T) \leq \lceil \log_2 W \rceil + 1$.*

Proof. Let $k = \chi_{mc}(T)$ and let $\{C_1, \cdots, C_k\}$ be the color classes in an optimal max-coloring of T. Let $w_i = weight(C_i)$ and without loss of generality assume that $w_1 \geq w_2 \geq \cdots \geq w_k$.

(i) Suppose $\chi_{mc}(T) > \Delta + 1$. For each vertex v in C_k, we can find a color class C_i, $i < k$ such that v is not adjacent to any vertex in C_i. We can thus move each vertex in C_k to a lower color class thus decreasing the coloring weight, a contradiction. Note that this upper bound holds in general for any graph G.

(ii) For each $i > 1$, we can assume without loss of generality that every vertex $v \in C_i$ has a neighbor in C_j, for every $j < i$.

For each vertex $v \in C_1$, let $T(v)$ denote the rooted tree with one vertex, namely v. For each $v \in C_i$, $i > 1$, define $T(v)$ as the tree rooted at v, such that (i) the children of v in $T(v)$ are exactly the neighbors of v in T belonging to color classes $C_1, C_2, \ldots, C_{i-1}$, and (ii) for each child u of v, the subtree of $T(v)$ rooted at u is simply $T(u)$. For each i, $1 \leq i \leq k$, let $S_i = \min\{|T(v)| \mid v \in C_i\}$. In other words, S_i is the size of a smallest tree $T(v)$ rooted at a vertex v in C_i. Then,

$$S_1 = 1$$
$$S_i \geq \sum_{j=1}^{i-1} S_j + 1, \text{ for each } i > 1$$

This implies that $S_i \geq 2^{i-1}$, $1 \leq i \leq k$. Using the fact that $S_k \leq n$, we get $\chi_{mc} = k \leq \lfloor \log_2 n \rfloor + 1$.

(iii) Let $\ell = \min\{t \in \mathbf{N} \mid \text{ for all } v \in V(T), w(v) \geq w_1/2^t\}$. Therefore, $\ell = \lceil \log_2 W \rceil$. Recall that W is the ratio of the weights of the heaviest vertex to the lightest vertex. Consider the collection of disjoint intervals $\mathcal{I} = \{I_0, I_1, \ldots, I_{\ell-1}\}$, where $I_i = [\frac{w_1}{2^{i+1}}, \frac{w_1}{2^i})$, for $i = 1, \ldots, \ell - 1$ and let $I_0 = [\frac{w_1}{2}, w_1]$. Because of the choice of ℓ, for each vertex $v \in V(T)$, $w(v)$ belongs to exactly one interval I_j. Let $V_j = \{v \in V(T) \mid w(v) \in I_j\}$, $j = 0, 1, \ldots, \ell - 1$. We say that a vertex v *contributes* to a color class C_i if $v \in C_i$, and $w(v) = \max\{w(u) \mid u \in C_i\}$. The *contribution* of an interval I_j is the maximum number of vertices in V_j that contribute to distinct color classes.

Corollary 1 tells us that $w_i \geq 2 \cdot w_{i+2}$ for $i = 1, \cdots k-2$. This immediately implies that no interval I_j, $j = 1, 2, \ldots, \ell - 1$ has a contribution of more than two. Now suppose that intervals $I_{i_1}, I_{i_2}, \ldots, I_{i_t}$, $0 \leq i_1 < i_2 < \cdots < i_t \leq \ell - 1$, is the sequence of all intervals in $\mathcal{I}$, each of whose contribution is two. We now claim that for any pair of consecutive intervals I_p, $p = i_j$ and I_q, $q = i_{j+1}$, where $j < t$, there is an interval in $\{I_{p+1}, I_{p+2}, \ldots, I_{q-1}\}$ with contribution zero. If we can show this claim, then we can charge the "extra" contribution of each I_{i_j} to an interval between I_{i_j} and $I_{i_{j+1}}$, whose contribution is zero. Since there are ℓ intervals and since the contribution of I_{i_t} is at most two, there is a total contribution of at most $\ell + 1$, implying that there are at most $\ell + 1$ color classes.

We prove the above claim by contradiction, assuming that the contribution of every interval in $\{I_{p+1}, I_{p+2}, \ldots, I_{q-1}\}$ is one. Let $\{x_p, x_{p+1}, \ldots, x_q\} \cup \{y_p, y_q\}$ be vertices such that (i) for each $j = p, p+1, \ldots, q$, $x_j \in V_j$ and x_j contributes to some color class and (ii) for each $j \in \{p, q\}$, $y_j \in V_j$ and x_j and y_j contribute to distinct color classes. Since $x_j \in V_j$, $w(x_j) \geq \frac{w_1}{2^{j+1}}$, $j = p, p+1, \ldots, q$. Also, since $y_q \in V_q$, $w(y_q) \geq \frac{w_1}{2^{q+1}}$. Therefore,

$$\begin{aligned}
\sum_{j=p}^{q} w(x_j) + w(y_q) &\geq \sum_{j=p}^{q} \frac{w_1}{2^{j+1}} + \frac{w_1}{2^{q+1}} \\
&= w_1 \frac{2^{q-p+1} - 1}{2^{q+1}} + \frac{w_1}{2^{q+1}} \\
&= \frac{w_1}{2^p} > w(y_p)
\end{aligned}$$

This contradicts Lemma 1 and proves the claim. □

There are simple examples that show that the bounds in Lemma 2 are all tight [6].

Since the number of colors are at most $\lceil \log n \rceil + 1$, this immediately gives a simple sub-exponential time algorithm. Try all $\lceil \log n \rceil + 1$ possible colors for each vertex, and return a feasible coloring of minimum weight. This algorithm runs in $O(n^{\log n + 1})$ time.

Now we show that if the given tree has a constant number of distinct vertex weights, we can find an optimal max-coloring in polynomial time. We deal with the case of constant number of distinct weights via the solution to a problem called `FEASIBLE` k-`COLORING`.

`FEASIBLE` k-`COLORING`

INPUT: A tree T with weight function $w : V \rightarrow \mathbf{N}$, and a positive integer sequence $(W_1, W_2, \cdots, W_k)$ of positive integers, satisfying $W_1 \geq W_2 \geq \cdots \geq W_k$.

OUTPUT: Either a coloring of the tree into color classes $A_1, \cdots, A_k$, such that for all $v \in A_i$, $w(v) \leq W_i$ or if such a coloring does not exist, a report that no such feasible coloring exists.

There is a simple dynamic programming algorithm for solving `FEASIBLE` k-`COLORING` on trees in $O(nk)$ time [6].

The main idea underlying our PTAS is the reduction of the number of distinct weights of the vertices down to a constant. We then pick candidates for the weights of the color classes and for each such choice, using the algorithm for `FEASIBLE` k-`COLORING`, we test if there is a legal coloring of the tree with the chosen weights for the color classes.

We are given a tree T, with weight function $w : V \rightarrow \mathbf{N}$ and an $\epsilon > 0$. Let $c > 0$ be an integer such that $(2 \log c + 3)/c \leq \epsilon$, and let $\alpha = (W - 1)/c$ where W is the maximum weight of any vertex. Let $I_1, I_2, \cdots, I_c$ be a partition of the range $[1, W)$, where $I_i = [1 + (i-1)\alpha, 1 + i \cdot \alpha)$, $1 \leq i \leq c$. Let T' be a tree that is identical to T, except in its vertex weights. The tree T' has vertex weights $w' : V \rightarrow \mathbf{N}$ defined by the rule: for any $v \in V$, if $w(v) \in I_j$ then $w'(v) = 1 + (j-1) \cdot \alpha$ and if $w(v) = W$, then $w'(v) = W$. In other words, except for vertices with maximum weight W, all other vertices have their weights "rounded" down. As a result T' has $c+1$ distinct vertex weights. Now let OPT' denote the weight of an optimal max-coloring of T' and let $\mathcal{C}' = C'_1, C'_2, \ldots, C'_k$ be the color classes corresponding to OPT'. Since the weights of vertices have fallen in going from T to T', clearly $OPT' \leq OPT$. If we use the coloring $\mathcal{C}'$ for T, we get a coloring whose weight is at most $OPT' + k\alpha$. Substituting $(W-1)/c$ for α and noting that $W \leq OPT'$, we obtain that weight of $\mathcal{C}'$ used as a coloring for T' is at most $(1 + \frac{k}{c})OPT'$ We now show that given the distribution of vertex weights of T', $k = O(\log c)$. If $k = 2$ we are done, so assume that $k \geq 3$. To see this first observe that the weights of last three color classes C'_k, C'_{k-1}, and C'_{k-2} cannot all be identical, by Lemma 1. Also, observe that the possible vertex weights of T' are $1, 1 + \alpha, 1 + 2\alpha, \ldots$. Therefore, $weight(C'_{k-2}) \geq 1 + \alpha$. From Corollary 1, we obtain

$$1 + \alpha \leq weight(C'_{k-2}) \leq \frac{W}{2^{\lfloor (k-3)/2 \rfloor}}.$$

Solving this for k yields $k \leq 2\log_2(c) + 3$. Therefore, by our choice of c, we have

$$\frac{k}{c} \leq \frac{2\log_2(c) + 3}{c} \leq \epsilon.$$

Thus $(1+\epsilon)OPT'$ is an upper bound on the weight of $\mathcal{C}'$ used as a coloring for T. Since $OPT' \leq OPT$, we see that the weight of C' used as a coloring for T is at most $(1+\epsilon)OPT$.

To construct OPT' in polynomial time, for each $k = 1, \ldots, 2\lceil \log_2 c \rceil + 3$, we generate all $O(c^k)$ possible sequences of weights and call algorithm `FEASIBLE k-COLORING` for each subsequence and pick the coloring with the minimum weight. This gives OPT'. Each solution to `FEASIBLE k-COLORING` takes $O(nk)$ time, and we have $O(c^k)$ sequences, for $k = 1, \ldots, 2\lceil \log_2 c \rceil + 3$. Using the fact that $(2\log_2 c + 3)/c \leq \epsilon$, a little bit of algebra yields a running time that is linear in n and exponential in $1/\epsilon$.

3 Max-coloring Bipartite Graphs

This section presents an $\frac{8}{7}$-approximation algorithm for the max-coloring problem on bipartite graphs, followed by a hardness of approximation result that shows that for any $\epsilon > 0$, there is no $(\frac{8}{7} - \epsilon)$-approximation algorithm unless $P = NP$. Thus our approximation algorithm produces an optimal approximation ratio.

One feature of our approximation algorithm is that it uses at most 4 colors, even though though an optimal max-coloring of an n-vertex bipartite graph may need a $\Omega(n)$ colors [6]. Our PTAS for the max-coloring problem on trees relied on the fact that the `FEASIBLE k-COLORING` problem on trees can be solved in polynomial time for any k. However, `FEASIBLE k-COLORING` is NP-complete for bipartite graphs for $k \geq 3$ [5]. This has forced us to use a different approach for bipartite graphs. Another difference between max-coloring on trees and max-coloring on bipartite graphs is that in contrast to the $O(\log n)$ upper bound on the number of colors used by an optimal max-coloring for an n-vertex tree, there are simple examples of n-vertex bipartite graphs G with $\chi_{mc}(G) \geq n/2$ [6].

Our $(\frac{8}{7} - \epsilon)$-hardness result for max-coloring bipartite graphs is via a gap introducing reduction from the `PRE-COLORING EXTENSION` problem [5].

3.1 An $\frac{8}{7}$-Approximation Algorithm

First note that since bipartite graphs are 2-colorable, Lemma 1 holds and hence if an optimal max-coloring of a bipartite graph uses a large number of colors, the contribution of all but the first few color classes must be quite small. We can use this to our advantage and develop an algorithm that tries to find a *good* approximation to the weights of the first few color classes. We run three algorithms, A_2, A_3, and A_4, that use 2, 3 and 4 colors respectively. The color classes produced by algorithm A_i, $2 \leq i \leq 4$, are denoted $\{A_1^i, A_2^i, \cdots\}$, and the weights of the corresponding color classes are denoted $\{a_1^i, a_2^i, \cdots\}$. We start with a description of algorithm A_2.

Algorithm $A_2(G, w)$
1. For each connected component G_i of G do
2. Color G_i with colors 1 and 2, such that a vertex with maximum weight is colored 1.

The fact that A_2 is a 2-approximation immediately follows from the fact that $weight(A_2) \leq 2w_1$, and $w_1 \leq OPT$. We encode this result in the following lemma.

Lemma 3. $weight(A_2) \leq 2w_1$

In an optimum coloring, the weight of the first color class, w_1 is fixed. By using more colors, OPT may gain an advantage because it can then push *heavy* vertices into *lower* color classes. We now introduce algorithm A_3 which constructs a 3-coloring of G such that the weight of the second color class is minimized.

Algorithm $A_3(G, w)$
1. Let S be a *maximal* independent set of G picked by examining vertices in non-increasing weight order.
2. Use Algorithm A_2 to color $G \setminus S$.
3. Rename colors 1 and 2, as colors 2 and 3 respectively.
4. Color S with color 1.

Lemma 4. $weight(A_3) \leq w_1 + 2w_2$.

Proof. In algorithm A_3, $a_1^3 = w_1$. Since S is a maximal independent set selected in non-increasing weight order, the weight of the second color class of OPT, w_2 cannot be smaller than the weight of any vertex in $G \setminus S$. Hence, $w_2 \geq a_2^3$. Since $a_3^3 \leq a_2^3$, it follows that $weight(A_3) = a_1^3 + a_2^3 + a_3^3 \leq w_1 + w_2 + w_2 = w_1 + 2w_2$. □

The greedy strategy employed by algorithm A_3 in selecting the first color class causes a_2^3 to be no larger than w_2. However, it might cause a_3^3 to be significantly larger than w_3. We rectify this situation by introducing algorithm A_4 that uses four colors to color G.

Algorithm $A_4(G, w)$
1. For all w^* such that there is a $u \in V$, with $w(u) = w^*$ do
2. Partition the vertices of G into two parts
   $P_1 = \{v \in V \mid w(v) > w^*\}$, and
   $P_2 = \{v \in V \mid w(v) \leq w^*\}$.
3. Use algorithm A_2 to color P_2.
4. Rename colors 1 and 2 as 3 and 4 respectively.
5. Use algorithm A_2 to color P_1.
6. Return the coloring with minimum weight, over all choices of w^*.

Lemma 5. $weight(A_4) < w_1 + w_2 + 2w_3$

Proof. Since the weight of every vertex in G is used for the threshold w^*, in some iteration of A_4, $w^* = w_3$. At this point, A_4 partitions the vertex set such that $P_1 = \{v \mid w(v) > w_3\}$ and $P_2 = \{v \mid w(v) \leq w_3\}$. In this iteration, A_4 colors P_1 with weight at most $w_1 + w_2$, and colors P_2 with weight at most $2w_3$. Since A_4 returns the coloring with minimum weight, over all choices of w^*, it follows that $weight(A_4) \leq w_1 + w_2 + 2w_3$. □

The final algorithm, which we call `Bipartite Max-Color` runs A_2, A_3, A_4, and returns the minimum weight coloring.

Theorem 1. *Algorithm* `Bipartite Max-Color` *is a* $\frac{8}{7}$*-approximation for the max-coloring problem on bipartite graphs.*

Proof. Let $w(B)$ denote the weight of the coloring produced by algorithm `Bipartite Max-Color`. From Lemmas 3, 4, and 5, we know that $w(B) \leq 2w_1$, $w(B) \leq w_1 + 2w_2$, $w(B) \leq w_1 + w_2 + 2w_3$. Now, multiplying the first inequality by 1, the second inequality by 2, the third inequality by 4 and adding, we get

$$7 \cdot w(B) \leq 8 \cdot (w_1 + w_2 + w_3) \leq 8 \cdot OPT$$ □

3.2 An $(\frac{8}{7} - \epsilon)$-Hardness Reduction

We now show that the 8/7-approximation produced by the above algorithm is optimal. We do this by showing a matching hardness result via a reduction from the `PRE-COLORING EXTENSION` problem on bipartite graphs. The `PRE-COLORING EXTENSION` problem for general graphs is defined below.

`PRE-COLORING EXTENSION`

INPUT: A graph $G = (V, E)$, with $r \geq \chi(G)$, a subset $P \subseteq V$, and a proper assignment $c : P \to \{1, \cdots, r\}$ of colors to vertices in P.

QUESTION: Is there an extension of the proper vertex coloring of P to a proper vertex coloring of G, using colors from $\{1, \cdots, r\}$?

In [5], Kratochvil proved that `PRE-COLORING EXTENSION` is NP-complete for planar bipartite graphs even when the color bound $r = 3$. We now show a simple gap introducing reduction from `PRE-COLORING EXTENSION` on bipartite graphs with $r = 3$ to max-coloring on bipartite graphs.

Theorem 2. *For any* $\epsilon > 0$*, there is no* $(8/7 - \epsilon)$*-approximation algorithm for max-coloring on bipartite graphs, unless P=NP.*

Proof. The reduction is from `PRE-COLORING EXTENSION`. Let the given instance of `PRE-COLORING EXTENSION` consist of a bipartite graph $G = (V_1, V_2, E)$, a subset $P \subseteq V_1 \cup V_2$, and a proper assignment $c : P \to \{1, 2, 3\}$ of colors to vertices in P. We transform G into a vertex-weighted bipartite graph $G' = (V_1', V_2', E')$ as follows. Add four new vertices, x_1, x_2, y_1, and y_2 to G. Let $X = \{x_1, x_2\}$,

$Y = \{y_1, y_2\}$, $V_1' = V_1 \cup X$, and $V_2' = V_2 \cup Y$. To each vertex $v \in P$, assign a weight $w(v)$ using the rule: $w(v) = 2^{3-i}$ if $c(v) = i$, for each $i \in \{1, 2, 3\}$. If $v \in (V_1 \cup V_2) - P$, set $w(v) = 1$. The new vertices are assigned weights as follows: $w(x_1) = w(y_1) = 4$ and $w(x_2) = w(y_2) = 2$. The edge set E' of G' contains some additional edges between the new vertices and the old.

$$E' = E \cup \{\{x_i, y\} | y \in P \cap V_2', \text{ and } w(y) < w(x_i)\} \cup \{\{y_i, x\} | x \in P \cap V_1' \text{ and } w(x) < w(y_i)\} \cup \{\{x_1, y_2\}\} \cup \{\{x_2, y_1\}\}.$$

This completes the description of G'.

Now suppose that the coloring of P can be extended to a proper 3-coloring $c : V_1 \cup V_2 \rightarrow \{1, 2, 3\}$ of G. Start with the coloring c and extend this to a proper vertex coloring of G' by assigning colors to the new vertices as follows: $c(x_1) = c(y_1) = 1$ and $c(x_2) = x(y_2) = 2$. Observe that this indeed produces a proper coloring of G'. To see that the weight of this coloring on G' has weight at most 7, note that the weight of the coloring in G' restricted to the vertices of G is at most 7, and since the coloring above of the vertices $\{x_1, x_2, y_1, y_2\}$ does not increase this cost, we are done.

Now suppose that G does not have a pre-coloring extension. We show by contradiction that in this case G' does not have a proper vertex coloring of weight less than 8. So suppose that there is a proper vertex coloring $c' : V_1' \cup V_2' \rightarrow \{1, 2, \ldots\}$ of weight less than 8. Without loss of generality, assume that in this coloring, the color classes are labeled in non-increasing order of their weight. Therefore, all vertices of weight 4 are in color class 1. This includes vertices x_1 and y_1 and this forces all vertices of weight 2 to be excluded from color class 1. Since color class 1 has weight 4, to prevent the total weight of the coloring from reaching 8, all vertices of weight 2 have to be included in color class 2. This includes vertices x_2 and y_2, and so this color class is also non-empty. Therefore the total weight of color classes 1 and 2 is 6. Since c' is a coloring of G' of weight less than 8, it must be the case that color class k, for each $k \geq 4$, is empty. This means that c' is a 3-coloring of G'. Furthermore, it is a 3-coloring of G that respects the pre-coloring of P. This contradicts the assumption that G has no pre-coloring extension and therefore we have that any proper vertex coloring of G' has weight at least 8.

If for some $\epsilon > 0$, there were an $(\frac{8}{7} - \epsilon)$-approximation algorithm for max-coloring bipartite graphs, then using the above polynomial time transformation from G to G', we could distinguish between positive and negative instances of `PRE-COLORING EXTENSION` on bipartite graphs with $r = 3$. This is not possible unless $P = NP$. □

4 Max-coloring on Arbitrary Graphs

Let $\mathcal{G}$ be a hereditary class of graphs for which the minimum vertex coloring problem has a c-approximation. In other words, there is a polynomial time algorithm A that takes a graph $G \in \mathcal{G}$ as input and returns a proper vertex coloring

of G using at most $c \cdot \chi(G)$ colors. In this section, we present an $4c$-approximation algorithm, that we call `GeomFit`, for the max-coloring problem on the class of graphs $\mathcal{G}$. The algorithm is inspired by the algorithm of Halldórsson, et. al. [4] for sum-coloring interval and comparability graphs. `GeomFit` will repeatedly use A as a black box to obtain "good" vertex colorings of portions of the input graph. For ease of exposition, below we describe `GeomFit` assuming that $c = 1$.

`GeomFit(`G`,` w`)`
1. Let $i = 0$, $l_i = 0$
2. While $G \neq \emptyset$ do
    3. Set $c_i = 2^i$
    4. Let $G_i = mkc(G, c_i)$
    5. Color G_i optimally using colors $l_i + 1, \cdots, l_i + c_i$
    6. Set $l_{i+1} = l_i + c_i$, $i = i + 1$.
    7. Set $G = G \setminus G_i$.
8. End While

A round of the algorithm corresponds to an iteration of the while loop. Suppose that each round is labeled with the value of i at the beginning of that round. For some integer $t > 0$, suppose that the algorithm executes rounds $0, 1, \cdots, t-1$, after which the graph is entirely colored. In each round i, $0 \leq i < t$, the algorithm calls the subroutine `mkc`(G, c_i), that returns a maximal c_i-colorable subgraph of G, obtained by examining vertices in non-increasing order of weight. Here G is the subgraph of the input graph induced by the not yet colored vertices and $c_i = 2^i$. When called, the subroutine `mkc`(G, c_i) starts with an empty set S and processes each vertex v of G, in non-increasing order of weight. The subroutine tests if $G[S \cup \{v\}]$ is c_i-colorable or not and if it is, it adds v to S, and proceeds to the next vertex in G. To perform this test, `mkc`(G, c_i) calls the algorithm A that returns a minimum vertex coloring of G.

Lemma 6. *If* `GeomFit` *uses t rounds to color G, then $\chi(G) > c_{t-2}$.*

Proof. In round $t-2$, the algorithm picks a maximal c_{t-2} colorable subgraph of G. If G were c_{t-2}-colorable, then all of it would have been picked up in round $t-2$ or earlier. Since we used one more round to color G, it must mean that $\chi(G) > c_{t-2}$. □

Without loss of generality, suppose that OPT uses numbers $1, 2, \ldots$ for colors such that color classes are numbered in non-increasing order of weight. Now observe that color classes created in round i by `GeomFit` are all heavier than color classes created in round $i+1$. Without loss of generality, assume that the color classes created in each round of `GeomFit` are numbered in non-increasing order of weight. Let $color_{OPT}(v)$ denote the color assigned to vertex v in OPT, Now using the color classes of OPT we define a pairwise disjoint collection of vertex subsets of G, $\{V_0, \cdots, V_{t-1}\}$, where $V_i = \{v \in G | c_{i-1} < color_{OPT}(v) \leq c_i\}$, $i = 0, \cdots, t-1$. For the definition to make sense, we assume that $c_{-1} = 0$. Since V_{t-1} contains vertices colored $c_{t-2} + 1, c_{t-2} + 2, \ldots, c_{t-1}$ by OPT, from Lemma

6, it follows that $V_{t-1} \neq \emptyset$. Now we state and prove a critical inequality that follows from the greedy choice of a subgraph in each round of `GeomFit`. Let W_i denote the weight of color class $c_{i-1}+1$ in OPT. Note that color class $c_{i-1}+1$ is a subset of V_i and by our labeling convention, it is a heaviest color class in V_i. Similarly, let R_i denote the weight of color class l_i+1 created by `GeomFit`. Note that this is a heaviest color class created in round i be `GeomFit`. Also note that $l_i = \sum_{j=0}^{i-1} c_j = c_i - 1$ and therefore color class $l_i + 1$ is simply color class c_i.

Lemma 7. *$R_i \leq W_i$, for $i = 0, 1, \cdots, t-1$.*

Proof. Since R_0 and W_0 are equal to the maximum weight vertex in G, the lemma holds for $i = 0$. By the greedy choice employed in selecting G_0, we ensure that for any other independent set S of G, the maximum weight of a vertex in $G \setminus S$ is at least as large as the maximum weight vertex in $G \setminus G_0$. This ensures that $R_1 \leq W_1$. By the same reasoning, since in round $i-1$, we greedily select a maximal c_{i-1} colorable subgraph of OPT, and $V_1 \cup V_2 \cup \cdots V_{i-1}$ is c_{i-1} colorable, it follows that $R_i \leq W_i$. □

Theorem 3. *Let $\mathcal{G}$ be a hereditary class of graphs on which the minimum vertex coloring problem can be solved in polynomial time. Algorithm* `GeomFit` *is a 4-approximation algorithm for the max-coloring problem on $\mathcal{G}$.*

Proof. The weight of the max-coloring produced by `GeomFit` is bounded above by

$$weight(\texttt{GeomFit}) \leq \sum_{i=0}^{t-1} c_i \cdot R_i \leq \sum_{i=0}^{t-1} c_i \cdot W_i$$

The first inequality follows from the fact that in each round i, `GeomFit` uses at most c_i colors and a heaviest color class in round i has weight R_i. The second inequality follows from Lemma 7.

We obtain a lower bound on OPT as follows. The set V_0 contains one color class and this has weight W_0. Now consider a set V_i, $1 \leq i \leq t-2$. It contains one color class of weight W_i and the remaining color classes have weight at least W_{i+1}. Recall that V_i has color classes labeled $c_{i-1}+1, c_{i-1}+2, \ldots, c_i$ and therefore $weight(V_i) \geq W_i + (c_{i-1}-1)W_{i+1}$.

$$\begin{aligned} OPT \geq \sum_{i=0}^{t-1} weight(V_i) &\geq W_0 + \sum_{i=1}^{t-2}(W_i + (c_{i-1}-1)W_{i+1}) + W_{t-1} \\ &= W_0 + W_1 + \sum_{i=0}^{t-3} c_i W_{i+2}. \end{aligned}$$

Therefore, $4 \cdot OPT \geq 4W_0 + 4W_1 + \sum_{i=0}^{t-3} 4c_i W_{i+2} = 4W_0 + 4W_1 + \sum_{i=2}^{t-1} c_i W_i$. This lower bound on $4 \cdot OPT$ is larger than the upper bound on $weight(\texttt{GeomFit})$ above. Therefore, $weight(\texttt{GeomFit}) \leq 4 \cdot OPT$. □

Now suppose that $\mathcal{G}$ is a heriditary class of graphs that has a c-approximation algorithm A for the minimum vertex coloring problem. A $4c$-approximation algorithm for max-coloring on graphs in $\mathcal{G}$ is obtained by modifying `GeomFit` slightly. In Step (4), the algorithm computes a maximal $\lfloor c \cdot c_i \rfloor$-colorable subgraph. Correspondingly, in Step (5), G_i is colored using colors $l_i + 1, \cdots, l_i + \lfloor c \cdot c_i \rfloor$. The analysis of this modified `GeomFit` proceeds in a manner similar to the $c = 1$ case. For details, see [6].

Theorem 4. *Let $\mathcal{G}$ be a hereditary class of graphs on which the minimum vertex coloring problem has a c-approximation algorithm. Algorithm* `GeomFit` *is a $4c$-approximation algorithm for the max-coloring problem on $\mathcal{G}$.*

The choice of $c_i = 2^i$ in `GeomFit` gave us an approximation factor of $4c$. This approximation factor can be improved to $3.5c$ by running `GeomFit` twice, once by setting $c_i = 2^i$, and once by setting $c_i = \lfloor 1.5 \times 2^i \rfloor$ and returning the coloring with smaller weight. More generally, setting $c_i = \lfloor \alpha q^i \rfloor$, where α is chosen uniformly at random from a certain range and q is an appropriately chosen constant, may yield a furthur improvement in the approximation ratio.

Acknowledgments. We would like to thank the anonymous referees for suggestions that led to a simplified proof of Theorem 1, and also for pointing out the work done in [3].

References

1. M.C. Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, NY, 1980.
2. R. Govindarajan and S. Rengarajan. Buffer allocation in regular dataflow networks: An approach based on coloring circular-arc graphs. In *Proceedings of the 2nd International Conference on High Performance Computing*, 1996.
3. D.J. Guan and Xuding Zhu. A coloring problem for weighted graphs. *Information Processing Letters*, 61:77–81, 1997.
4. Magnús M. Halldórsson, Guy Kortsarz, and Hadas Shachnai. Sum coloring interval and k-claw free graphs with application to scheduling dependent jobs. *Algorithmica*, 37(3):187–209, 2003.
5. J. Kratochvil. Precoloring extensions with a fixed color bound. *Acta Mathematica Universitatsis Comenianae*, 62:139–153, 1993.
6. S. V. Pemmaraju and R. Raman. Approximation algorithms for the max-coloring problem. `http://www.cs.uiowa.edu/~sriram/papers/tbPerfectFull.ps`.
7. S.V. Pemmaraju, R. Raman, and K. Varadarajan. Buffer minimization using max-coloring. In *Proceedings of The ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 562–571, 2004.