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 subexponential 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 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 [11] and in digital signal processing applications [6]. 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.

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

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 [11], 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 [5]. [11] 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. Perfect graphs include many well-known subclasses of graphs such as bipartite graphs, interval graphs, chordal graphs, and permutation graphs. Thus max-coloring has a 4-approximation for all of these classes of graphs. In addition, our result also implies a constant-factor approximation algorithm for 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. 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 $P = NP$. Thus our approximation algorithm yields an optimum approximation factor. Finally, we also present an exact subexponential algorithm and a PTAS for trees.

Related problems. There are two classes of problems that seem related to max-coloring in the sense that techniques for solving those problems might be applicable to max-coloring and vice versa. One of these is the interval coloring problem, also known as the *dynamic storage allocation* problem. Like the max-coloring problem, the *interval coloring* problem takes as input a graph $G = (V, E)$ and positive integral vertex weights $w : V \to \mathbf{N}$. The problem seeks to find an assignment of an interval $I(u)$ to each vertex $u \in V$ such that two constraints are satisfied: (i) for every vertex $u \in V$, $|I(u)| = w(u)$ and (ii) for every pair of adjacent vertices u and v, $I(u) \cap I(v) = \emptyset$. The goal is to minimize the span $|\cup_v I(v)|$. The interval coloring problem has a fairly long history dating back, at least to the 70's. For example, Stockmeyer showed in 1976 that the interval coloring problem is NP-complete even when restricted to interval graphs and vertex weights in $\{1, 2\}$ (see problem SR2 in Garey and Johnson [4]). Currently, the best approximation algorithm for the interval coloring problem on interval graphs is a $(2 + \epsilon)$-approximation algorithm due to [2]. It is easy to see that any feasible solution to max-coloring is also a feasible solution to the interval coloring problem. Thus the weight of an optimal max-coloring of a graph is bounded below by the span of an optimal interval coloring of that graph. However, as pointed out in [11], even for interval graphs, the weight of an optimal max-coloring of a graph may be $\Omega(\log n)$ times the span of an optimal interval coloring. Despite this, we hope that some of the ideas in this paper, will help us devise constant-factor approximation algorithms for the interval coloring problem on perfect graphs – the current best approximation factor is $O(\log n)$.

Another class of problems that has connections to max-coloring is the class of graph multicoloring problems. The input to this class of problems, is also a graph $G = (V, E)$ with a weight function $w : V \to \mathbf{N}$ on the vertices. A *multicoloring* of G is an assignment of a set $S(v)$ of natural numbers to each vertex, where $|S(v)| = w(v)$, and adjacent vertices receive distinct colors. i.e., $S(u) \cap S(v) = \phi$, for all $\{u, v\} \in E$. The set of colors, or numbers assigned to each vertex correspond to time units when the job is run, and this assignment gives us a *pre-emptive* schedule. Let $f_\psi(v) = \max\{i \mid i \in S_\psi(v)\}$ denote the completion time of the job v in a multicoloring ψ. In the standard multicoloring problem, the goal is to minimize the makespan $\max_v f_\psi(v)$. The *non-preemptive* version of multicoloring imposes the restriction that the set $S(v)$ of colors assigned to any vertex be consecutive numbers. It is not difficult to see that the non-preemptive multicoloring problem is just the interval coloring problem.

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

The papers [1, 8] introduced the *sum multicoloring* problem which seeks to minimize the *average completion time* of the jobs. Given a multicoloring ψ, the average completion time of the schedule corresponding to this multicoloring is $\sum_{v \in V} f_\psi(v)$. The best results currently on the sum multicoloring problem are due to [3], who present (along with some other results) a 5.536-approximation algorithm for the sum multicoloring problem on perfect graphs, and a 11.273-approximation algorithm for the non-preemptive version of sum multicoloring on interval graphs.

[1] also studies the *co-scheduling* version of multicoloring. Here, the jobs are to be scheduled in batches with one batch of jobs starting only after all jobs in the previous batch have ended. Let $c_{min}(v) = \min\{i \mid i \in S(v)\}$ denote the minimum number in $S(v)$. Then, a feasible co-schedule is a multicoloring of G where for all $u, v \in V$, $S(u) \cap S(v) \neq \phi \Rightarrow c_{min}(u) = c_{min}(v)$. In other words, if there is a common time instant at which two jobs u and v execute, then it must be the case that both belong to the same batch and therefore start at the same time. In [1], the objective of minimizing the average completion time of a co-schedule has been studied and this paper presents a 16-approximation algorithm for problem of minimizing the average completion time of a co-schedule on perfect graphs. It is easy to see that the max-coloring problem has the same set of feasible solutions as the co-scheduling problem, but with the objective of minimizing the makespan rather than the average completion time. Thus, the $4c$-approximation result in this paper provides a 4-approximation algorithm for the co-scheduling problem with minimum makespan.

A note on notation. Throughout this paper all graphs are finite and simple, and vertices of these graphs usually have associated positive integral weights. We let $\Delta, \chi, \alpha, \omega$ denote the maximum degree, the chromatic number, the stability number and clique number respectively of a graph. When we use OPT, it will typically denote both an optimum max-coloring of G and the weight of an optimal max-coloring. Which of these two OPT exactly stands for will be clear from the context. Let $\chi_{mc}(G)$ denote the *max-color number* of G; i.e., the minimum k such that G has a max-coloring of weight OPT with k colors.

2 Max-Coloring Trees

The max-coloring problem has turned out to be surprisingly hard even for trees. Though we believe that the problem can be solved in polynomial time, the two best algorithms we have are (i) a subexponential 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. In any optimal max-coloring $\{C_1, C_2, \ldots, C_k\}$ of G with $w_i = weight(C_i)$ and $w_1 \geq w_2 \geq \cdots \geq w_k$, 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. In any optimal max-coloring $\{C_1, C_2, \ldots, C_k\}$ 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. We now state three upper bounds on χ_{mc} of trees, the first of which applies to arbitrary graphs as well.

Lemma 2. *Let G be a vertex-weighted graph with maximum vertex degree Δ. Then $\chi_{mc}(G) \leq \Delta + 1$.*

Proof. Let $k = \chi_{mc}(G)$ and let $C_1, C_2, \ldots, C_k$ be an optimal max-coloring of G. Let $w_i = weight(C_i)$ and without loss of generality assume that $w_1 \geq w_2 \geq \cdots \geq w_k$. If $k > \Delta + 1$, then for each vertex $v \in C_k$, there is a color class C_i, $i < k$, such that v has no neighbors in C_i. Note that since $w_i \geq w_k$, when v is moved into C_i, the weight of the coloring does not increase. Furthermore, when C_k becomes empty, the weight of the coloring decreases, contradicting the optimality of $C_1, C_2, \ldots, C_k$.

Lemma 3. *Let T be a vertex-weighted tree on n vertices. Then, $\chi_{mc}(T) \leq \lfloor \log_2 n \rfloor + 1$.*

Proof. Let $k = \chi_{mc}(G)$ and let $C_1, C_2, \ldots, C_k$ be an optimal max-coloring of G. Let $w_i = weight(C_i)$ and without loss of generality assume that $w_1 \geq w_2 \geq \cdots \geq w_k$. 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 $\chi_{mc} = k \leq \lfloor \log_2 n \rfloor + 1$.

Lemma 4. *Let T be a vertex-weighted tree on n vertices, and let W be the ratio of the weight of heaviest vertex to the weight of the least heavy vertex. Then, $\chi_{mc}(T) \leq \lceil \log_2 W \rceil + 1$.*

Proof. Let $k = \chi_{mc}(T)$ and let $C_1, C_2, \ldots, C_k$ be an optimal max-coloring of T. For $1 \leq i \leq k$, let $w_i = weight(C_i)$ and without loss of generality assume that $w_1 \geq w_2 \geq \cdots \geq w_k$. Thus w_1 is the weight of the heaviest vertex in the tree. Let $\ell = \min\{t \in \mathbf{N} \mid \text{ for all } v \in V(T), w(v) \geq w_1/2^t\}$. Therefore, $\ell = \lceil \log_2 W \rceil$.

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. If the contribution of I_0 is three or more, then it must be the case that we can construct a 2-coloring with the same or smaller weight, compared to $\{C_1, C_2, \ldots, C_k\}$. This contradicts the fact that $k = \chi_{mc}(T)$ and $C_1, C_2, \ldots, C_k$ is an optimal max-coloring of T.

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 prove the following claim:

> **Claim:** For any pair of consecutive intervals I_p, $p = i_j$ and I_q, $q = i_{j+1}$, where $j < t$, it is the case that 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. This implies that the contributing vertices in all intervals except I_t can be reassigned to a distinct interval. Since there are ℓ intervals and since the contribution of 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.

The upper bounds in the three preceding lemmas are all tight, as the following example will show. Let $T_0, T_1, T_2, \ldots$ be a sequence of trees where T_0 is a single vertex, with weight 1, and T_i, $i > 0$, is constructed from T_{i-1} as follows. Let $V(T_{i-1}) = \{u_1, u_2, \ldots, u_k\}$. To construct T_i, start with T_{i-1} and add a set of new vertices $\{v_1, v_2, \ldots, v_k\}$, each with weight 2^i, and edges $\{u_i, v_i\}$ for all $i = 1, 2, \ldots, k$. Thus the leaves of T_i are $\{v_1, v_2, \ldots, v_k\}$ and every other vertex in T_i has a neighbor v_j for some j. Now consider a tree T_n in this sequence. Clearly, $|V(T_n)| = 2^n$ and the maximum vertex degree of T_n, $\Delta(T_n) = n - 1$. See Figure 1 for T_0, T_1, T_2, and T_3. Now consider the coloring of T_n defined by $C_i = \{$ leaves of $T_{n+1-i}\}$, $1 \leq i \leq n+1$. Note that $weight(C_i) = 2^{n+1-i}$ and therefore the weight of the coloring is $2^{n+1} - 1$. It is not hard to see that any coloring using fewer than $n+1$ colors has weight at least 2^{n+1}. Therefore,

$$\chi_{mc} = n + 1 = \log_2 |V(T_n)| + 1 = \Delta + 1 = \log_2 \left(\frac{2^n}{1}\right) + 1.$$

Since the number of colors are at most $\lfloor \log n \rfloor + 1$, this immediately gives a simple sub-exponential time

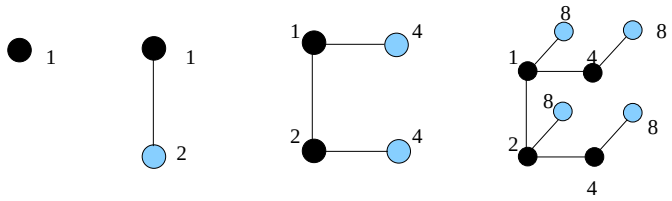

Fig. 1. Sequence of trees which show that the upper bounds of Lemmas 2, 3 and 4 are all tight. The figure above shows the trees $T_0, \cdots, T_3$.

algorithm. Try all $\lfloor \log n \rfloor + 1$ 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 tree has a constant number of distinct weights, we can find an optimal max-coloring in polynomial time. Later this will turn out to be critical for our PTAS. 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 $w : V \to \mathbf{N}$, and a 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.

Here is a simple dynamic programming algorithm for solving `FEASIBLE` k-`COLORING` on trees in $O(nk)$ time. Let T be rooted at an arbitrary vertex r. Let $ch(v)$ denote the set of children of v, and

let $parent(v)$ denote the parent of v. For a vertex v, let $T(v)$ denote the sub-tree of T rooted at v. Let $[k]$ denote the set of colors $\{1, \cdots, k\}$. For a vertex v, let $F(v) = \{j \mid w(v) > W_j\}$ be the set of forbidden colors, and let $S(v) = \{i \mid$ there exists a feasible coloring of $T(v)$ with $color(v) = i\}$. The algorithm to compute a feasible coloring, if one exists is as follows. The correctness of the algorithm and the running time are easy to established and this is summarized in the lemma below.

Algorithm FKC

```
1. Let S(v) = φ, ∀ v ∈ T.
2. For each vertex v in a post-order traversal of T do
3.  For each color i ∈ [k] − F(v)
4.          If S(u) − {i} ≠ φ ,∀u ∈ ch(v)
5.                  Set S(v) = S(v) ∪ {i}.
6. If S(r) = φ, return NULL.
7. Pick an arbitrary i ∈ S(r) and set color(r) = i.
8. For each v in a pre-order traversal of T
9.          Pick an arbitrary j ∈ S(v) − color(parent(v)) and set color(v) = j
```

Lemma 5. *Algorithm FKC solves the* FEASIBLE k-COLORING *problem in* $O(nk)$ *time.*

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 $w : V \to \mathbf{N}$ and an $\epsilon > 0$. Let $c > 0$ be an integer such that $(2 \log c + 3)/c \leq \epsilon$, and let $\alpha = (w_1 - 1)/c$. Let $I_1, I_2, \cdots, I_c$ be a partition of the range $[1, w_1)$, 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 $w' : V \to \mathbf{N}$ 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 $w(v) = w_1$, then $w'(v) = w_1$. In other words, except for vertices with maximum weight w_1, 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 $(w_1 - 1)/c$ for α and noting that $w_1 \leq OPT'$, 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', $k = O(\log c)$. 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

$$1 + \alpha \leq w(C_{k-2}) \leq \frac{w_1}{2^{\lfloor (k-3)/2 \rfloor}}.$$

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 OPT' used as a coloring for T is at most $(1 + \epsilon)OPT$.

To construct OPT' in polynomial time, for each $k = 1, \ldots, 2 \log c + 3$, 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 $k = 1, \ldots, 2 \log c + 3$. 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 a bipartite graph may need an unbounded number of colors. 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 $k \geq 3$ [10]. This has forced us to use a different approach for bipartite graphs. Another difference 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 $\chi_{mc}(G) \geq n/2$. One such an example is shown in Figure 2.

Our $(\frac{8}{7} - \epsilon)$-hardness result for max-coloring bipartite graphs is via a gap introducing reduction from the `PRE-COLORING EXTENSION` problem [10].

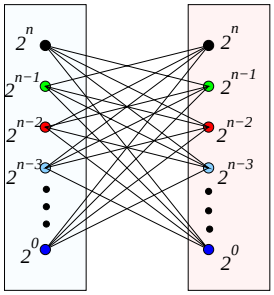

Fig. 2. An instance of max-coloring of a bipartite graph on $2n$ vertices that requires n colors in an optimal max-coloring. The weights of the vertices are powers of 2 and are shown next to the vertices. A k-coloring for any $k < n$ has weight at least 2^n, while a coloring with n colors has weight $2^n - 1$.

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 6. $weight(A_2) \leq 2w_1$

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 7. *$weight(A_3) \leq w_1 + 2w_2$. If $w_2 \leq \frac{1}{2}w_1$, then $weight(A_3) \leq \frac{3}{2}w_1 + w_2$.*

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 $weight(A_3) = a_1^3 + a_2^3 + a_3^3 \leq w_1 + w_2 + w_2 = w_1 + 2w_2$. If $w_2 \leq \frac{1}{2}w_1$, then by plugging this inequality into the above upper bound for $weight(A_3)$, we get the second inequality.

As a warm-up to our main result, we now show that running A_2 and A_3 together and selecting a coloring with smaller weight gives a 4/3-approximation.

Theorem 1. *Let A be the algorithm that runs A_2 and A_3 and returns, from among the two colorings produced, a coloring with smaller weight. Algorithm A is a 4/3-approximation algorithm for max-coloring bipartite graphs.*

Proof. There are two cases depending on the relative values of $weight(A_2)$ and $w_1 + 2w_2$. If $weight(A_2) \leq w_1 + 2w_2$, then combining this with the bound from Lemma 6 we get that $weight(A_2) \leq min\{2w_1, w_1 + 2w_2\}$. Since $OPT \geq w_1 + w_2$, we get

$$\frac{weight(A_2)}{OPT} \leq \min\left\{\frac{2w_1}{w_1 + w_2}, \frac{w_1 + 2w_2}{w_1 + w_2}\right\}.$$

The function on the right attains a maximum of $\frac{4}{3}$ when the two functions inside the min operator are equal. This happens at $w_1 = 2w_2$.

If $weight(A_2) > w_1 + 2w_2$, then using Lemma 7, we derive the inequality $weight(A_3) \leq w_1 + 2w_2 < weight(A_2) \leq 2w_1$. Thus $w_2 \leq w_1/2$. Hence from Lemma 7 we get $weight(A_3) \leq min\{w_1 + 2w_2, \frac{3}{2}w_1 + w_2\}$. Using the lower bound of $w_1 + w_2$ on OPT, we get

$$\frac{weight(A_3)}{OPT} \leq \min\left\{\frac{w_1 + 2w_2}{w_1 + w_2}, \frac{3w_1/2 + w_2}{w_1 + w_2}\right\}.$$

Again, the function on the right, attains a maximum at $\frac{4}{3}$ and this happens when $w_1 = 2w_2$.

We can improve this ratio, by using a fourth color. 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 8. *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.

```
Bipartite Max-Color(G, w)
    1. Run algorithms A_2, A_3, A_4.
    2. Return the coloring of minimum weight.
```

Theorem 2. *Algorithm* `Bipartite Max-Color` *is a* $\frac{8}{7}$*-approximation for the max-coloring problem on bipartite graphs.*

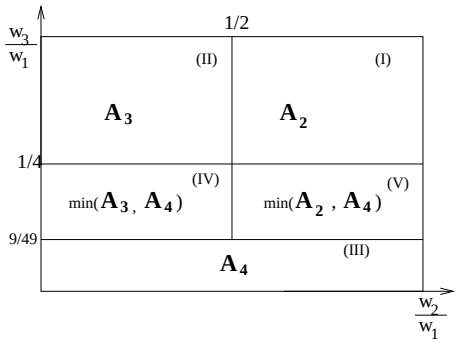

Fig. 3. Figure showing the different cases for the proof based on values of w_2 and w_3 as fractions of w_1. The x-axis shows possible values of the ratio w_2/w_1 and the y-axis shows possible values of the ratio w_3/w_1. Note that the former ratio is in the range $(0, 1]$ and the latter is in the range $(0, 1/2]$. Each box in the diagram is labeled with a case in the proof that it corresponds to and also with the subset of algorithms that lead to the 8/7 factor in that case.

Proof. The proof is split into several cases depending on the weights of first 3 color classes of OPT. The different cases in the proof are shown in the Figure 3

Case 1 : $w_3 \geq \frac{1}{4}w_1$**, and** $w_2 \geq \frac{1}{2}w_1$**.** Thus, $OPT = \sum_{i=1}^{k} w_i \geq w_1 + w_2 + w_3 \geq \left(1 + \frac{1}{2} + \frac{1}{4}\right) \geq \frac{7}{4}w_1$. Using this lower bound with the upper bound on A_2 from Lemma 6, we get $weight(A_2) \leq 2 \cdot w_1 \leq 2 \cdot \frac{7}{4} \cdot OPT = \frac{8}{7}OPT$.

Case 2 : $w_3 \geq \frac{1}{4}$ **and** $w_2 \leq \frac{1}{2}w_1$ In this case, we have the following lower bound on OPT: $OPT \geq w_1 + w_2 + w_3 \geq \left(1 + \frac{1}{4}\right)w_1 + w_2 = \frac{5}{4}w_1 + w_2$. Using the upper bound on A_3 from Lemma 7, we get

$$\frac{weight(A_3)}{OPT} \leq \frac{w_1 + 2w_2}{\frac{5}{4}w_1 + w_2} \tag{1}$$

Since the function on the right-hand side is monotonically increasing with w_2, it attains a maximum at $w_2 = \frac{1}{2}w_1$. Substituting this value for w_2 in Equation (1), we get that $weight(A_3) \leq \frac{8}{7} \cdot OPT$.

Case 3: $w_3 \leq \frac{9}{49}w_1$**.** In this case, algorithm A_4 itself gives us the desired bounds. Let $\alpha \leq 9/49$, be such that $w_3 = \alpha \cdot w_1$. From the definition of OPT, we get the following lower bound on OPT: $OPT = \sum_{i=1}^{k} w_i \geq w_1 + w_2 + w_3 \geq w_1 + 2 \cdot w_3 = (1 + 2\alpha)w_1$. We now try to determine the worst case bounds for w_3 in terms of OPT. First, since $w_1 \leq OPT$, we get that $w_3 \leq \alpha \cdot OPT$. Also, because we assume that $w_3 \leq \frac{9}{49}w_1$ in this case, from the above lower bound on OPT, we get $w_3 \leq \frac{9}{49} \cdot \frac{OPT}{(1+2\alpha)}$. Combining the two upper bounds on w_3 we get

$$w_3 \leq \min\left\{\alpha, \frac{9}{49(1+2\alpha)}\right\} \cdot OPT \tag{2}$$

Now, using Lemma 8 for an upper bound on $weight(A_4)$ and Equation (2) for an upper bound on w_3, we get the following:

$$weight(A_4) \leq w_1 + w_2 + 2 \cdot w_3 \leq OPT + w_3 \leq \left(1 + \min\left\{\alpha, \frac{9}{49(1+2\alpha)}\right\}\right) \cdot OPT$$

The right hand side of the above inequality is maximized when the two functions inside the min operator are equal. This happens at $\alpha = \frac{1}{7}$ and yields $weight(A_4) \leq \frac{8}{7}OPT$.

Now we are left with the case when w_3 is in the band $(9/49, 1/4)$. In this range, the color classes $w_4, \cdots, w_k$ can make a significant contribution to OPT, making our approximation harder. However, we show that in this case a combination of the three algorithms will give us the desired bounds.

Case 4 : $w_2 \leq \frac{1}{2}w_1$ **and** $w_3 \in \left(\frac{9}{49}, \frac{1}{4}\right)$**.** Let A denote the algorithm that returns the minimum of the solutions produced by A_3 and A_4, and let $weight(A)$ denote the weight of the solution produced by algorithm A. Since the upper bounds of algorithms A_3 and A_4 simultaneously apply to A, using Lemma 7, and 8 and the lower bound on OPT, we get

$$\frac{weight(A)}{OPT} \leq \min\left\{\frac{w_1 + 2w_2}{w_1 + w_2 + w_3}, \frac{w_1 + w_2 + 2w_3}{w_1 + w_2 + w_3}\right\}$$

The right-hand side is maximized when the two functions inside the min operator are equal. This happens at $w_2 = 2w_3$. Using this to substitute for w_3, we get

$$\frac{weight(A)}{OPT} \leq \frac{w_1 + 2w_2}{w_1 + 3w_2/2}.$$

The right-hand side of this function monotonically increases with w_2 and attains a maximum at $w_1 = w_1/2$. Substituting this into the right-hand side above, we get that $weight(A) \leq \frac{8}{7}OPT$.

Case 5 : $w_2 \geq \frac{1}{2}w_1$ **and** $w_3 \in \left(\frac{9}{49}, \frac{1}{4}\right)$**.** Similar to the previous case, let B be the algorithm that returns a coloring of minimum weight from among the two obtained by running A_2 and A_4. Let $weight(B)$ denote

the weight of the solution returned by B. Then, using the upper bounds on A_2 and A_4 from Lemmas 6, and 8 respectively, and the known lower bounds on OPT, we get

$$\frac{weight(B)}{OPT} \leq \min\left\{\frac{2w_1}{w_1+w_2+w_3}, \frac{w_1+w_2+2w_3}{w_1+w_2+w_3}\right\}$$

Again, the right-hand side is maximized when the two functions inside the min operator are equal. This happens when $w_1 = w_2 + 2w_3$. We use this equation to substitute out w_3 from the above inequality and obtain

$$\frac{weight(B)}{OPT} \leq \frac{2w_1}{3w_1/2 + w_2/2}.$$

The above right-hand side decreases monotonically with increasing w_2 and therefore it attains a maximum when w_2 is smallest, that is, $w_2 = w_1/2$. Substituting this for w_2, we get that $weight(B) \leq \frac{8}{7}OPT$.

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 chromatic number $\chi(G) = r$, 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 [10], 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 3. *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 on bipartite graphs. 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 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'. Figure 4 illustrates this construction.

Now suppose that the coloring of P can be extended to a proper 3-coloring $c : V_1 \cup V_2 \to \{1, 2, 3\}$ 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 $c(x_2) = x(y_2) = 2$. To see that this coloring of G' is proper, observe that all neighbors of x_1 have weights 1 or 2 and are in $P \cup \{y_2\}$. By our construction of G' from G, all such neighbors, with the exception of y_2, were colored in the given pre-coloring of P with some color distinct from 1. Furthermore, $c(y_2) = 2 \neq c(x_1)$. A similar argument shows that the colors assigned to y_1, y_2, and x_2 are all proper.

Now we show that the coloring c has weight at most 7. Since the maximum weight of any vertex in G' is 4, the weight of color class 1 is at most 4. Also, no vertex with weight 4 is in color class 2. This is

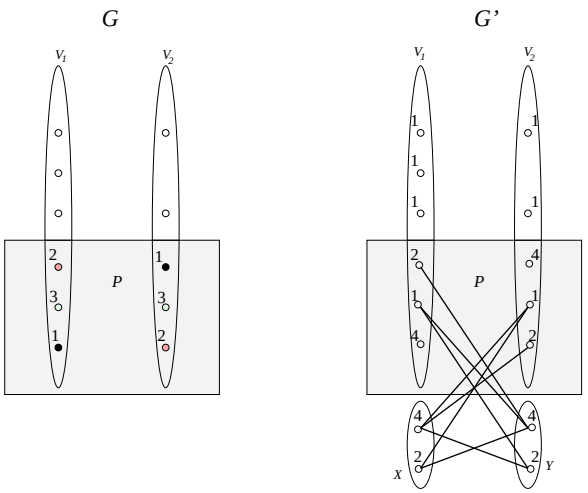

Fig. 4. On the left is an instance G of PRE-COLORING EXTENSION for bipartite graphs, with $r = 3$. Vertices in the set P are "pre-colored" with colors from $\{1, 2, 3\}$. On the right is the bipartite instance G' of max-coloring, constructed from G. The new vertices in $X \cup Y$, the assignment of weights to vertices, and the edges from the new vertices to the old vertices, are all shown.

because our construction was such that any vertex in P assigned weight 4 has a pre-coloring of 1. The only other vertices with weight 4 are x_1 and y_1. These have been explicitly colored 1. Therefore, the maximum weight of a vertex in color class 2 is 2. A similar argument shows that no vertex with weight 2 or more is in color class 3, thereby showing that the weight of color class 3 is at most 1. This shows that the coloring c has weight at most 7.

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 $c' : V_1' \cup V_2' \to \{1, 2, \ldots\}$ 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. This is not possible unless $P = NP$.

Note that PRE-COLORING EXTENSION was proved NP-complete for bipartite, *planar* graphs for $r = 3$. We can modify the above reduction to maintain planarity of the input bipartite graph, by introducing a pair of vertices, one of weight 4 and one of weight 2, to connect to each vertex in P. This shows that max-coloring is impossible to approximate to a factor better than 8/7, even on *planar* bipartite graphs.

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 $\mathcal{G}$. `GeomFit` will repeatedly use A as a black box to obtain "good" vertex colorings of portions of the input graph.

A graph G is *perfect* if for every induced subgraph A of G, $\chi(A) = \omega(A)$. The class of perfect graphs is a well-known example of a class of graphs that can be optimally colored in polynomial time via the ellipsoid algorithm of Grötschel, Lovász and Schirjver [7]. Therefore, for the class of perfect graphs, `GeomFit` provides a 4-approximation algorithm for max-coloring. This is the first known constant-factor approximation algorithm for perfect graphs. The class of perfect graphs includes many well-known subclasses such as bipartite graphs, interval graphs, chordal graphs, and permutation graphs. For bipartite graphs we have presented a better approximation algorithm earlier in the paper, for interval graphs [11] provide a 2-approximation, but for chordal graphs and permutations graphs `GeomFit` is the first constant-factor approximation algorithm for max-coloring. In [11], the authors showed that max-coloring on interval graphs is NP-complete and from this, it follows that max-coloring on perfect graphs is NP-complete as well. In addition, there are well-known classes of graphs that are not perfect, but have constant-factor approximation algorithms for solving the minimum vertex coloring problem. These classes include *circle graphs*, *circular arc graphs*, and *unit disk graphs*. Thus `GeomFit` provides an $O(1)$-approximation for all of these classes of graphs. Our algorithm uses ideas from [9], in which the authors presented a constant-factor approximation algorithm for the sum-coloring problem on interval graphs.

For ease of exposition, below we first describe `GeomFit` assuming that $c = 1$. In other words, we assume that a minimum vertex coloring of the input graph can be computed in polynomial time. To obtain a $4c$-approximation for arbitrary $c \geq 1$, `GeomFit` needs to be modified very slightly and the analysis that shows the $4c$ approximation factor is quite similar to the analysis in the $c = 1$ case. The modified `GeomFit` and the modified analysis are presented subsequently.

GeomFit(G, w)
1. Let $i = 0$, $l_i = 0$
2. While $G \neq \phi$ 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. Assuming that A runs in polynomial time, each call to the subroutine `mkc`(G, c_i) also runs in polynomial time. Step (5) of the above algorithm is also executed in polynomial time by calling the algorithm A. Since the number of rounds $t = O(\log(n))$, the entire algorithm runs in polynomial time. We start our analysis of with a simple observation.

Lemma 9. *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 9, it follows that $V_{t-1} \neq \phi$. Now we state and prove a critical observation 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 10. *$R_i \leq W_i$, 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 4. *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 10.

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, $weight(\texttt{GeomFit}) \leq 4 \cdot OPT$.

It is worth pointing out a slight strengthening of the above analysis. In the above proof, the upper bound on $weight(\texttt{GeomFit})$ and the lower bound on $4 \cdot OPT$, can be combined to yield

$$4 \cdot OPT \geq 3W_0 + 2W_1 + weight(\texttt{GeomFit}).$$

Let k denote the chromatic number of the input graph G. Then $k \cdot W_0 \geq OPT$ and therefore we have $weight(\texttt{GeomFit}) \leq (4 - \frac{3}{k})OPT$.

We now assume that $\mathcal{G}$ is a hereditary class of graphs that has a c-approximation algorithm A for the minimum vertex color problem. We modify `GeomFit` so that in round i, 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 $l_i + 1, l_i + 2, \ldots, l_i + \lfloor c \cdot c_i \rfloor$.

The analysis proceeds in a manner similar to the analysis for the $c = 1$ case. Suppose that `GeomFit` finishes coloring G in t rounds, $0, 1, \ldots, t-1$. In round $(t-2)$, `GeomFit` finds a maximal $\lfloor c \cdot c_{t-2} \rfloor$-colorable subgraph and there is at least one uncolored vertex left over for round $t-1$. This implies that algorithm A needs at least $\lfloor c \cdot c_{t-2} \rfloor + 1$ colors for the input graph G. Since A is a c-approximation for the minimum vertex coloring problem, $\chi(G) > c_{t-2}$. So OPT has to use more than c_{t-2} colors for G. Partition the vertex set V of G according to the coloring used by OPT, exactly as before. For $i = 0, 1, \ldots$, $V_i = \{v \mid c_{i-1} < color_{OPT}(v) \leq c_i\}$. Then since $\chi(G) > c_{t-2}$, $V_{t-1} \neq \emptyset$. As before, let W_i be the weight of color class $c_{i-1} + 1$ in OPT. Recall our assumption that OPT numbers color classes $1, 2, \ldots$ in non-increasing order of weight. Similarly, let R_i denote the weight of a heaviest color class created in round i, by `GeomFit`. Using the same reasoning as in Lemma 10, we obtain that $R_i \leq W_i$ for all $i = 0, 1, \ldots, t-1$. As before, a lower bound on OPT is

$$OPT \geq W_0 + W_1 + \sum_{i=0}^{t-3} c_i \cdot W_{i+2}.$$

As upper bound on $weight(\texttt{GeomFit})$ is

$$weight(\texttt{GeomFit}) \leq \sum_{i=0}^{t-1} \lfloor 2^i c \rfloor R_i.$$

It follows that $weight(\texttt{GeomFit}) \leq 4c \cdot OPT$ and we obtain the following theorem.

Theorem 5. *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 $\mathcal{G}$.*

5 Open Questions

While max-coloring on bipartite graphs has been satisfactorily dealt with by the results in this paper, the situation for arbitrary graphs is not as clear. We believe that our $4c$-approximation algorithm can be improved to yield a better approximation factor. Furthermore, the strongest hardness of approximation result we have is the $(\frac{8}{7} - \epsilon)$-hardness of approximation of max-coloring on bipartite graphs. It is quite possible that for classes of graphs such as perfect graphs, this result can be strengthened. Finally, we believe that max-coloring on trees can be solved in polynomial time, but the best algorithms we currently have are (i) an exact algorithm that runs in $O(n^{\log n})$ time and (ii) a $(1 + \epsilon)$-approximation algorithm.

References

1. A. Bar-Noy, M. Halldorsson, G. Kortsarz, R. Salman, and H. Shachnai. Minimum sum multi-coloring of graphs. *Journal of Algorithms*, 37:422–450, 2000.

2. A.L. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold, and M. Thorup. OPT versus LOAD in dynamic storage allocation. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, 2003.
3. R. Gandhi, M. M. Halldorsson, G. Kortsarz, and H. Shachnai. Improved bounds for sum multicoloring and scheduling dependent jobs with minsum criteria. In *Proc. of the Second Workshop on Approximation and Online Algorithms (WAOA 2004)*, pages 68–82, 2004.
4. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the theory of NP-completeness.* W.H. Freeman and Company, San Fransisco, 1979.
5. M.C. Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, NY, 1980.
6. 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.
7. M. Grotschel, L. Lovasz, and A. Schrijver. *Geometric algorithms and Combinatorial Optimization.* Springer Verlag, 1993.
8. M. Halldorsson, G. Kortsarz, A. Prokurowski, R. Salman, H. Shachnai, and J.A.Telle. Sum multi-coloring trees. In *Proceedings of 5th Annual International Computing and Combinatorics Conference (COCOON)*, pages 171–180, 1999.
9. 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.
10. J. Kratochvil. Precoloring extensions with a fixed color bound. *Acta Mathematica Universitatsis Comenianae*, 62:139–153, 1993.
11. 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.