JOURNAL OF ALGORITHMS 13, 657–669 (1992)

Randomized Online Graph Coloring*

SUNDAR VISHWANATHAN

Department of Computer Science, University of Chicago, Chicago, Illinois 60637

Received November 1990; revised July 1991

We study the problem of coloring graphs in an online manner. The only known deterministic online graph coloring algorithm with a sublinear performance function was found by Lovász, Saks, and Trotter, (*Discrete Math.* **75** (1989), 319–325). Their algorithm colors graphs of chromatic number χ with no more than $(2\chi n)/\log^* n$ colors, where n is the number of vertices. They point out that the performance can be improved slightly for graphs with bounded chromatic number. For three-chromatic graphs the number of colors used, for example, is $O(n \log\log\log n/\log\log n)$. We show that randomization helps in coloring graphs online. We present a simple randomized online algorithm to color graphs with expected number of colors $O(2^\chi \chi^2 n^{(\chi-2)/(\chi-1)}(\log n)^{1/(\chi-1)})$. For three-colorable graphs the expected number of colors our algorithm uses is $O(\sqrt{n \log n})$. All our algorithms run in polynomial time. It is interesting to note that our algorithm compares well with the best known polynomial time *offline* algorithms. For instance, the best polynomial time algorithm known for three-colorable graphs, due to Avrim Blum, uses $O(n^{3/8}$ poly-log$(n))$ colors (*in* "Proceedings, 31st Annual Sympos. Found. Comput. Sci., 1990," pp. 554–562). We also prove a lower bound of $\Omega((1/(\chi-1))((\log n/(12(\chi+1))) - 1)^{\chi-1})$ for the randomized model. No lower bound for the randomized model was previously known. For bounded χ, our result improves even the best known lower bound for the deterministic case: $\Omega((\log n/\log\log n)^{\chi-1})$, due to Noga Alon (personal communication, September 1989). © 1992 Academic Press, Inc.

1. INTRODUCTION

In this paper we consider the problem of coloring a graph online. In this model a graph is presented a vertex at a time. When a new vertex is presented only the edges from this vertex to vertices already presented are revealed. An online algorithm has to irrevocably assign a color to a vertex as soon as it is presented. The objective of an online algorithm is to color

*A preliminary version of this paper appeared in the "Proceedings of the 31st Symposium on the Foundations of Computer Science."

0196-6774/92 $5.00
Copyright © 1992 by Academic Press, Inc.
All rights of reproduction in any form reserved.

the graph so presented with as few colors as possible. For an online graph coloring algorithm $\mathcal{A}$ and a graph G with an ordering on its vertices (the order in which the vertices will be presented to the algorithm), let $\chi_{\mathcal{A}}(G)$ denote the number of colors used by $\mathcal{A}$ to color G. Let $\chi(G)$ denote the chromatic number of the graph, i.e., the minimum number of colors needed to color G. The performance ratio denoted by $\rho_{\mathcal{A}}(G)$ is defined to be $\chi_{\mathcal{A}}(G)/\chi(G)$. The performance function, denoted $\rho_{\mathcal{A}}(n)$, is defined for each integer n to be the maximum of $\rho_{\mathcal{A}}(G)$ over all n vertex ordered graphs G.

The problem of finding an online algorithm with small performance function has been considered by various authors. Recently Lovász, Saks, and Trotter [9] presented an algorithm with performance function $O(n/\log^* n)$. They observed that their algorithm could be modified to color three-colorable graphs with $O(n \log\log\log n/\log\log n)$ colors. This is the best deterministic upper bound known today.

There has been work done on proving lower bounds on the performance function of any online graph-coloring algorithm. Szegedy [10] showed that for any online algorithm $\mathcal{A}$ and integer k there is a graph on at most $k(2^k - 1)$ vertices with chromatic number k that requires at least $2^k - 1$ colors. Thus the performance function of any online deterministic algorithm grows at least as fast as $n/(\log n)^2$. For bounded χ the lower bounds known are worse. It was proved in [2] that there exists a family of trees on which any algorithm would require at least $\log n$ colors. Márió Szegedy and the author found a $\Omega((\log n/\log\log n)^{\chi/2})$ lower bound. Independently, using similar techniques, Noga Alon [1] found a $\Omega((\log n/\log\log n)^{\chi-1})$ lower bound. It has been an open question to shrink the gap between the lower and upper bounds for graphs of bounded chromatic number. We take a step in both directions, improving both the upper and lower bounds for graphs of bounded chromatic number.

In this paper we first prove, in Section 2, better upper bounds in a randomized setting. For instance, our randomized online algorithm colors three-colorable graphs with $O(\sqrt{n \log n})$ expected number of colors. Further, this algorithm only takes a polynomial amount of time to decide on the color of each vertex. It is interesting to see that this compares well with the best known polynomial-time *offline* coloring algorithm for three-colorable graphs which uses $O(n^{3/8}$ poly-log $n)$ colors (Blum, [4]). Though our algorithm was primarily designed to work well for graphs of bounded chromatic number, it also performs better than the algorithm of Lovász, Saks, and Trotter on graphs whose chromatic number is not bounded. The performance function our algorithm achieves is $O(n/\sqrt{\log n})$. Recently, Halldórsson [6], with a more careful design of the recursive step, has shown how to improve this to $O(n/\log n)$. This, however, only leads to constant factor improvements for the bounded chromatic number case.

Halldórsson and Szegedy [5] also prove a lower bound of $n/(\log n)^3$ on the performance function of randomized algorithms. We then prove, in Section 3, a $\Omega((1/(\chi - 1))((\log n/(12(\chi + 1))) - 1)^{\chi-1})$ lower bound for the randomized model and obtain, as a corollary, improved lower bounds for the deterministic case.

2. A Randomized Algorithm for Online Coloring

2.1. *Preliminaries*

The following is a description of our model. An adversary determines the graph and the ordering of the vertices and this is fixed for the rest of the game. He presents one vertex at a time. We can take as much time as we want (possibly an exponential amount) but we have to assign a color to this vertex before he lets us see the next vertex. We have access to a perfect random bit generator. The goal is to color the graph with a small expected number of colors. Another way to look at this model is that we have a family $\mathcal{F}$ of deterministic online algorithms and we pick one at random according to some probability distribution over $\mathcal{F}$. This is because we can toss the coins before beginning actual execution and each sequence of coin tosses determines one deterministic algorithm.

An obvious way to color online is to use the greedy algorithm. A color c would be considered admissible for an incoming vertex v if none of its current neighbors have been colored c. Let the color set be the set of positive integers. The greedy algorithm assigns to each vertex the smallest admissible color. This method does not work well on general graphs. In fact it is an easy exercise to construct bipartite graphs on which the greedy algorithm uses $n/2$ colors. But the greedy algorithm works reasonably well on sparse graphs and we leave it as an easy exercise to show that on graphs with m edges the greedy algorithm uses at most $2\sqrt{m} + 1$ colors. The power of the greedy algorithm for coloring sparse graphs online is evident in the works of Sandy Irani [7] and Howard Karloff [8].

We assume initially that we know both χ and n in advance. We then present simple modifications to dispose of both these assumptions. We need the following theorem, posed as "an easy exercise" in the paper by Lovász *et al.* [9].

Theorem 2.1. *There is an algorithm that colors bipartite graphs on $n \geq 2$ vertices with at most $4 \log n$ colors.*

2.2. *The Algorithm*

In this section we exhibit a randomized algorithm to color graphs online. We prove the following theorem:

THEOREM 2.2. *There exists a randomized polynomial-time online graph coloring algorithm $\mathcal{A}$ such that for every graph G of chromatic number χ, and every ordering on the vertices of G, the expected number of colors used by $\mathcal{A}$ is $O(\chi 2^{\chi} n^{(\chi-2)/(\chi-1)}(\log n)^{1/(\chi-1)})$.*

No a priori knowledge of χ or the number of vertices is made in these results. Let us fix a graph G and an ordering on the vertices. For the remainder of this paper by a graph we will mean a graph with an order $\prec$. By the phrase "greedy coloring" of a subset of the vertex set of G we will mean that the vertices, say $v_1, \ldots, v_k$, are considered in order $v_1 \prec v_2 \prec \cdots \prec v_k$ and at the i^{th} step we assign the smallest admissible color to v_i.

DEFINITION 1. By a *partial greedy s-coloring* of G we mean that we greedily color G with s colors, leaving out vertices that cannot be colored. We call the set of vertices left uncolored the *residue set* and denote it by R. We will call a set of vertices assigned the same color by a partial greedy coloring a greedy color class.

An important property of the partial greedy coloring is that each vertex in the residue is adjacent to some vertex of each greedy color class. (This "maximality" of the partial greedy coloring is the property that we use—in fact one could use any partial coloring satisfying this criterion.) The essence of the algorithm is to color the residue efficiently.

We now introduce the notion of partitioning the residue with respect to a greedy color class. Consider a partial coloring of G with s colors. Let the greedy color classes be $C_1, \ldots, C_s$. Consider a color class C_i. Let the vertices in this color class be $v_1, \ldots, v_l$, where $v_1 \prec v_2 \prec \cdots \prec v_l$. With every vertex in R we associate the first vertex in C_i to which it is adjacent. This partitions R into $B_1, \ldots, B_l$. Each B_j will be called a *block* of the partition. Observe that once C_i is fixed this partition does not depend on the distribution of vertices in the other C's. Then

- Each B_j has chromatic number $\leq \chi - 1$.
- Once we have picked a color i we can determine online the partition of R with respect to color class C_i.
- The average number of vertices in a color class is $\leq n/s$ where s is the number of greedy colors.

Since the algorithm and its analysis are particularly simple as well as illuminating for the case $\chi = 3$, we begin with an informal discussion of this case. To simplify the presentation, we will describe the offline version of the algorithm first and then observe that this algorithm can in fact be implemented in an online fashion. Let the vertex set of the input graph

G be $\{v_1, \ldots, v_n\}$. Let the greedy color set be the set of integers $C = \{1, \ldots, 2\sqrt{n \log n}\}$. Here is the algorithm.

1. Pick uniformly at random an integer r from C.
2. Considering the vertices in order, color them greedily with colors from C leaving uncolored, vertices that could not be colored. (Partially $|C|$-color G.)
3. Let the vertices colored r be $u_1, \ldots, u_k$. Let B_i be the set of vertices adjacent to u_i but not adjacent to the vertices u_1 through u_{i-1}. (The B_i's are the blocks of the partition, with respect to the color class r.) Color the B_i's using disjoint sets of colors (these color sets are also disjoint from C).

It is clear that step 2 can be done online. Now note that the graphs induced by each B_i are bipartite. Hence the algorithm from [9] can be used in step 3, to color the B_i's, using at most $4 \log n$ colors per B_i. So the total expected number of colors used is $|C| + 4 \log n *$ (expected value of k). Since r was picked uniformly at random, the expected value of k is $n/|C|$. Simplifying, the total expected number of colors $= 4\sqrt{n \log n}$.

We now describe the algorithm for graphs of arbitrary chromatic number. Our algorithm is recursive, recursing down to $\chi - 1$ levels. First, we unfold the recursion (in the form of a recursion tree) and informally describe the working of the algorithm.

Consider a rooted tree (an auxiliary structure that models the recursion and helps us define the algorithm) of height $\chi - 2$. The exact structure of the tree will be apparent at the end of this discussion. So as not to confuse vertices of the tree with the vertices of the input graph, we will call the vertices of the tree *nodes*.

The root will be said to be at level χ, its children at level $\chi - 1$, etc. The leaves will be at "level 2." We associate with each node on level t an online graph-coloring algorithm that obtains t-chromatic graphs as input.

When a vertex is input, it is first input to the coloring algorithm at the root. This algorithm either colors it or passes the vertex on to one of its children. So, the vertex passes down the tree until it encounters an algorithm that can color it, at which point it becomes colored.

Consider now an algorithm $\mathscr{A}_u$ at some node u at level t. The algorithm has, associated with it, a set of greedy colors of size $s = s(t)$. These color sets, corresponding to different nodes in the tree, are disjoint. Since the number of colors that $\mathscr{A}_u$ uses is limited, it may not be able to color all of the vertices it receives, and hence it will be left with a residue set. These it will pass on to its children. The algorithm $\mathscr{A}_u$ decides in advance which color class to use to partition its residue set by choosing a color class uniformly at random. This random choice is made independently for each

node in the tree. This is the only place in the algorithm where randomization is used. Each child corresponds to one block of the residue set. So the algorithm $\mathscr{A}_u$ has two parts. The partial greedy coloring part (called the PGC part, which may color an input vertex) and a residue set partitioning part (called the RSP part). When a new vertex v is presented to this algorithm by its parent, it first checks to see if v can be colored using its set of greedy colors. If yes, the PGC part colors it. If not then the RSP part determines the block B of the partition into which the vertex falls. It then passes this vertex on to the algorithm at the child of u which corresponds to the block B. $\mathscr{A}_u$ has one child for each block in the partition of the residue at u.

The chromatic number of the graphs passed down from a vertex is at most one less than the chromatic number of the graph input to that vertex. Hence the chromatic number of a graph passed to a vertex at level l is at most l. The graphs input to level two we can color deterministically (see Theorem 2.1). Thus every input vertex gets colored somewhere in the tree.

The number of colors used at each node is a function of both χ and n. Hence we need a good estimate of both χ and n at each node of the tree. Let us now look at one of the sons of the root, say w, and the graph H induced by the vertices input to the coloring algorithm at node w. Clearly $\chi(H) \leq \chi(G) - 1$. So we do have a good estimate of the chromatic number of H (viz. $\chi - 1$), but n seems to be the best estimate we have on the number of vertices. It turns out that we do much better by limiting the size of each block to s. Thus we split each block B_i into at most $\lceil \text{size}(B_i)/s \rceil$ blocks of size at most s each. By doing this we lose, in that the number of blocks (and hence the number of sons of a node) may increase (but not by more than n/s), but we gain more in reducing the estimate of the number of vertices. Moreover, this refinement of the partition can be done online.

To describe the algorithm let us define a function that determines the number of greedy color classes that we use: $S(n, \chi) = \min\{\lceil 2^{\chi} n^{(\chi-2)/(\chi-1)} (\log n)^{1/(\chi-1)} \rceil, n\}$. Here is the algorithm.

ALGORITHM Online_color(n, χ).

1. If $\chi \leq 2$ then use a bipartite coloring algorithm that uses at most $4 \log n$ colors (see Theorem 2.1).
2. Set $s := S(n, \chi)$.
3. Choose a random integer r uniformly from $\{1, \ldots, s\}$.
4. Repeat until there are no more vertices.
   (a) Get the next vertex v.
   (b) If v can be colored using the greedy set of colors $1, \ldots, s$: Color the vertex greedily. For every new vertex colored r invoke a copy of Online_color$(s, \chi - 1)$. Skip (c).

(c) Otherwise (v is in the residual set R): Determine which block B of the partition the vertex falls into. If the number of vertices in that partition exceeds s invoke a new copy of Online_color($s, \chi - 1$). In any case, input this vertex to the copy of Online_color corresponding to B.

Recall that the colors used in different copies of the algorithm are disjoint. Denote by $A(n, \chi)$ the maximum, over all n node ordered graphs of chromatic number χ, the expected number of colors used by the algorithm. On a graph of chromatic number χ, the depth of recursion is $\chi - 1$. For each non-leaf node in the recursion tree we roll a die.

THEOREM 2.3. *For* $\chi \geq 2$, $A(n, \chi) \leq \chi 2^{\chi} n^{(\chi-2)/(\chi-1)} (\log n)^{1/(\chi-1)}$.

Proof. By induction on χ.

Basis. The basis follows from the fact that $A(n, 2) \leq 4 \log n \leq 2 \cdot 2^2 n^0 \log = 8 \log n$ colors (see Theorem 2.1).

Inductive step. Let G be a graph with n vertices and chromatic number χ, on which the performance of the algorithm is the worst. Let $s = S(n, \chi)$. The number of colors used by the algorithm is the number of greedy colors s, plus the number of colors used to color the residual set. Look at some greedy color class i. The residual set is partitioned (initially) by C_i into $B_1, \ldots, B_{n_i'}$, where $n_i' = |C_i|$. Further partitioning of the residual set by limiting the size of each block to s adds at most n/s extra blocks. Let the residual set be partitioned into $G_1^i, \ldots, G_{n^i}^i$, where $n_i \leq n_i' + n/s$. (The G_j^i's are the subgraphs induced by the blocks.) Each choice of i from $\{1, \ldots, s\}$ has probability $1/s$. Over subsequent rolls of the die, let $E(G_j^i)$ denote the expected number of colors used by the algorithm to color the graph G_j^i. We note that the chromatic number of the G_j^i's are at most $\chi - 1$ and hence we can use the inductive hypothesis. Also the size of each G_j^i is at most s. Hence $E(G_j^i) \leq A(s, \chi - 1)$ by induction. So the expected number of colors to color G is

$$\begin{aligned} A(n, \chi) &= s + \frac{1}{s} \sum_{i=1}^{s} \left(E(G_1^i) + \cdots + E(G_{n_i}^i) \right) \\ &\leq s + \frac{A(s, \chi - 1)}{s} \sum_{i=1}^{s} n_i \\ &\leq s + \frac{A(s, \chi - 1)}{s} \sum_{i=1}^{s} (n_i' + n/s) \\ &\leq s + \frac{2n}{s} A(s, \chi - 1). \end{aligned}$$

The last inequality follows because $\sum_{i=1}^{s} n_i' \leq n$.

For simplicity we let $c = 2^{\chi}$ and $s' = s/c = n^{(\chi-2)/(\chi-1)} (\log n)^{1/(\chi-1)}$. We need to prove $A(n, \chi) \leq c\chi s' = \chi s$:

$$
\begin{aligned}
A(n,\chi) &\leq s + \frac{2n}{s} A(s, \chi - 1) \\
&\leq s + \frac{2n}{s}(\chi - 1)2^{\chi-1} s^{(\chi-3)/(\chi-2)} (\log s)^{1/(\chi-2)} \\
&= s + \frac{n}{s'}(\chi - 1) s^{(\chi-3)/(\chi-2)} (\log s)^{1/(\chi-2)} \\
&\leq s + \frac{n}{s'}(\chi - 1) c^{(\chi-3)/(\chi-2)} s'^{(\chi-3)/(\chi-2)} (\log n)^{1/(\chi-2)} \\
&< s + c(\chi - 1) n s'^{-1/(\chi-2)} (\log n)^{1/(\chi-2)} \\
&\leq s + c(\chi - 1) n^{1-1/(\chi-1)} (\log n)^{1/(\chi-2)-1/(\chi-2)(\chi-1)} \\
&= s + c(\chi - 1) n^{(\chi-2)/(\chi-1)} (\log n)^{1/(\chi-1)} \\
&= s + c(\chi - 1) s' \\
&= s + (\chi - 1) s \\
&= \chi s. \qquad \square
\end{aligned}
$$

To complete the proof of the main theorem: We call Online_color(n, χ') as soon as the chromatic number of the input graph increases from $\chi' - 1$ to χ'. Computing the chromatic number of the graph seen so far can be done in exponential time. However, we can reduce this time to a polynomial amount with the following observation: The requirement that that the input graph be χ-chromatic for Algorithm Online_color(n, χ) to perform as claimed is too strong. It suffices that at the leaves of the recursion tree one obtains bipartite graphs. So, as long as the input to Online_color$(n, 2)$ is bipartite, the proofs go through. Hence, one only needs to invoke Online_color(n, χ') when the input to some algorithm coloring bipartite graphs becomes three-chromatic. This check (to see if a graph is bipartite) can be done in polynomial-time.

Until now we have assumed that the number of vertices is known in advance. This is no serious restriction, since we only perform a constant

time as badly when we run the algorithm $A(2^i, \chi)$ on vertices $2^{i-1}, \ldots, 2^i - 1$. This is because, for $\chi \geq 3$ and for $2^{k-1} \leq n < 2^k$,

$$\begin{aligned}\sum_{i=1}^{k-1} A(2^i, \chi) &= \sum_{i=1}^{k-1} \chi 2^\chi (2^i)^{(\chi-2)/(\chi-1)} (\log 2^i)^{1/(\chi-1)} \\ &\leq \sum_{i=1}^{k-1} \chi 2^\chi (2^i)^{(\chi-2)/(\chi-1)} 2(\log n)^{1/(\chi-1)} \\ &= \chi 2^\chi 2(\log n)^{1/(\chi-1)} \sum_{i=1}^{k-1} (2^i)^{(\chi-2)/(\chi-1)} \\ &\leq \chi 2^\chi 2(\log n)^{1/(\chi-1)} 3(2^k)^{(\chi-2)/(\chi-1)} \\ &\leq 6A(n, \chi).\end{aligned}$$

(*Note.* $\sum_{i=1}^{k-1} (2^i)^{(\chi-2)/(\chi-1)} \leq 3(2^k)^{(\chi-2)/(\chi-1)}$.) This technique is used implicitly in [9].

3. Lower Bounds for Randomized Online Graph Coloring

3.1. *Basic Techniques and the Intuition behind Our Construction*

We first discuss possible methods to prove lower bounds for online graph coloring. Here we discuss the deterministic case, but the basic concepts carry over to the randomized case too.

Fix a deterministic online graph coloring algorithm $\mathscr{D}$. We will see various ways of building a graph on which we force $\mathscr{D}$ to use a large number of colors. The graph is built in *stages*. At stage k we build a graph G_k so that $\mathscr{D}$ is forced to use a large number of colors on G_k.

Our goal is to force $\mathscr{D}$ to use extra colors while guaranteeing that the chromatic number of the graph does not increase.

One technique used by Noga Alon [1] is to take many disjoint copies of G_k, add a new vertex connected to a vertex of color i in the i^{th} copy of G_k. The drawback of this technique is that we need to use a large number of vertices to force extra colors.

We next sketch a technique, formalized in the next section, to reduce the number of vertices. Fix χ, the actual chromatic number of the graph. The construction proceeds in stages. Suppose that we have forced $\mathscr{D}$ to use a large number of colors on G_k. Since G_k is χ-colorable we note that on at least one of the χ color classes, say, the color class c, $\mathscr{D}$ used a large number of colors. Suppose, further, that $\mathscr{D}$ used most of its colors in this part. (Although this assumption is not justified, we shall achieve a similar

effect by a doubling trick.) We now introduce a $\chi - 1$-colorable graph H connected to all vertices in this color class c. We note that the graph stays χ-colorable. Also since H is adjacent to all vertices colored c, $\mathscr{D}$ must use a fresh set of colors in H. By induction we can force the algorithm to use a large number of colors on H.

3.2. *The Construction*

In this section we prove the following lower bound for randomized online coloring of graphs.

THEOREM 3.1. *For any randomized online coloring algorithm $\mathscr{A}$ there exists an n-vertex, χ-colorable graph G so that the expected number of colors $\mathscr{A}$ uses to color G is at least* $\Omega((1/(\chi - 1))((\log n/(12(\chi + 1))) - 1)^{\chi - 1})$.

By Yao's lemma [11] it is sufficient to prove the following theorem.

THEOREM 3.2. *There is a distribution on n-node, χ-colorable graphs such that the expected number of colors used by any online (deterministic) algorithm is at least* $\Omega((1/(\chi - 1))((\log n/(12(\chi + 1))) - 1)^{\chi - 1})$ *for* $\chi \geq 2$ *and one for* $\chi = 1$.

For every fixed $\chi \geq 2$ we actually prove the theorem for infinitely many values of n that are close enough to each other so that the $\Omega(\ )$ notation holds. The proof will be by double induction. The primary induction will be on χ. The base case ($\chi = 1$) is trivial. Let t be an integer. This is a parameter that will decide the number of vertices in the graphs that we construct.

We construct the probability distribution on these graphs inductively (the secondary induction) in *stages*. For chromatic number χ, at stage k, we construct a probability distribution on a family of graphs which we call $\mathscr{G}_k(\chi)$. $G_k(\chi)$ will denote a graph drawn from the distribution $\mathscr{G}_k(\chi)$.

Further, we define a special independent set of vertices $I_k(\chi)$, in every graph $G_k(\chi)$ drawn from $\mathscr{G}_k(\chi)$. (We will show later that when we draw a graph G_k from $\mathscr{G}_k$, any deterministic algorithm would use a large number of colors on I_k with probability at least a half.)

The Construction.

$\chi = 1$: $\mathscr{G}_k(1)$ consists of a single vertex for every integer k.

$\chi > 1$:

Stage $k = 1$; *the family* $\mathscr{G}_1(\chi + 1)$. $\mathscr{G}_1(\chi + 1) = \mathscr{G}_t(\chi)$.

Stage $k + 1$; *the family* $\mathscr{G}_{k+1}(\chi + 1)$. $G_{k+1}(\chi + 1)$ consists of 16 disjoint graphs drawn from a distribution on another family $\mathscr{F}_{k+1}$ or $\mathscr{F}$ for short; I_{k+1} is the union of their special independent sets. Let us define what a member $G_{\mathscr{F}}$ of $\mathscr{F}$ looks like. We will denote its special independent set by $I_{\mathscr{F}}$. Draw two independent copies from $\mathscr{G}_k(\chi + 1)$, say H_1 and H_2, with their respective special independent sets I_1 and I_2. Let B be a graph

drawn from $\mathcal{G}_{2t}(\chi)$ and let its special independent set be J. $G_{\mathcal{F}}$ is the union of H_1, H_2, and B with some additional edges that we will specify shortly. Vertices of H_1 come first, followed by vertices of H_2, and, finally, B. We join each vertex of B to each vertex of I_1. We now flip an unbiased coin and define the special independent set $I_{\mathcal{F}}$ to be either $I_1 \cup I_2$ or $I_2 \cup J$ with probability one-half. This completes the construction.

The Analysis. We can now fix a deterministic online algorithm $\mathcal{A}$ and consider its performance on these graphs constructed. Since we will require that these graphs we construct be colorable in a certain way, we will use the term that $\mathcal{A}$ *paints* the graph with *pigments* while we, the offline adversary, color it with colors (this is to help us prove that the graphs we construct are χ-colorable).

We will call the special independent set $I_k(\chi)$ in $G_k(\chi)$ *good* if $\mathcal{A}$ uses at least $(k/(\chi - 1))t^{\chi-2}$ pigments if $\chi \geq 2$ and one if $\chi = 1$, on I_k. We now prove the following claim by induction.

CLAIM 1. *I_k is good with probability at least $\frac{1}{2}$. Further, there is a χ-coloring of $G_k(\chi)$ such that $I_k(\chi)$ is entirely contained in one color class.*

Proof of Claim. The induction for this proof will closely follow the induction in the construction. Our primary induction is on χ and the secondary induction is on k.

The assertion is trivial for $\chi = 1$. We first look at the case when $k = 1$. For $\chi > 1$, by construction $\mathcal{G}_1(\chi + 1) = \mathcal{G}_t(\chi)$. In $G_t(\chi)$, by the primary inductive hypothesis, there is a special independent set I_1 such that with probability at least $\frac{1}{2}$, $\mathcal{A}$ uses $t^{\chi-1}/(\chi - 1) > t^{\chi-1}/\chi$ (which is one if $\chi = 1$) pigments on I_1. These graphs are trivially $(\chi + 1)$-colorable by the inductive hypothesis.

Consider now, the construction of $G_{k+1}(\chi + 1)$. We claim that if I_1, I_2, and J were good then either $I_1 \cup I_2$ or $I_2 \cup J$ is good. Note that we picked one of them to be $I_{\mathcal{F}}$ with probability one-half. First, we show that they satisfy the second condition, that there is a $\chi + 1$ coloring of the new graph so that these sets are contained in one color class:

1. $I_1 \cup I_2$. If we pick this then we $\chi + 1$ color H_1 and H_2 so that I_1 and I_2 obtain the same color, say 1, and color B with colors $2, \ldots, \chi + 1$.

2. $I_2 \cup J$. Here we color H_1 and H_2 so that I_1 and I_2 obtain different colors, and χ color B, so that the part containing J obtains the same color as I_2.

Assume that I_1, I_2, and J are good.

CLAIM 2. *On either $I_1 \cup I_2$ or $I_2 \cup J$ the online algorithm $\mathcal{A}$ uses at least $((k + 1)/\chi)t^{\chi-1}$ pigments.*

Proof. In this proof, for a set S, we will let $\chi_{\mathscr{A}}(S)$ stand for the number of pigments used by the online algorithm $\mathscr{A}$ on S. Let $l = t^{\chi-1}/\chi$. Since I_1 and I_2 are good we have $\chi_{\mathscr{A}}(I_1)$ and $\chi_{\mathscr{A}}(I_2) \geq kl$. If $\chi_{\mathscr{A}}(I_1 \cup I_2) \geq (k+1)l$ we are done, so assume otherwise. By inclusion–exclusion we obtain that the number of pigments used by $\mathscr{A}$ in I_1 which are also used in I_2 is at least $(k-1)l$. Let us call this set of colors S. Since the pigments used in B (and hence in J) are not used by $\mathscr{A}$ on I_1, we have $\chi_{\mathscr{A}}(I_2 \cup J) \geq |S| + \chi_{\mathscr{A}}(J)$. Since J was assumed to be good, $\chi_{\mathscr{A}}(J) \geq 2l$ and hence $\chi_{\mathscr{A}}(I_2 \cup J) \geq (k-1)l + 2l = (k+1)l$. □

To finish the proof of the outer claim we now estimate the probability that $I_{\mathscr{F}}$ is good. It is Prob(I_1 is good $\wedge$ I_2 is good $\wedge$ J is good $\wedge$ we picked the correct set for $I_{\mathscr{F}}$) $\geq \frac{1}{16}$, since the events are independent. As mentioned earlier, G_{k+1} is the union of 16 independent members of $\mathscr{F}$ and I_{k+1} is the union of their special independent sets. It is clear that with probability $\geq 1 - (\frac{15}{16})^{16} > \frac{1}{2}$, at least one of the independent sets chosen is good, completing the proof of correctness of the inductive claim for stage $k+1$. □

We now analyze the size of the graphs constructed. Let $m = 2^t$.

CLAIM 3. $size(G_k(\chi)) \leq m^{12\chi}(2^{6(k+1)} - 2^6)$.

Proof. By induction on χ and k. Following the construction all we need is a formula for size($G_k(\chi)$) that satisfies the following three inequalities:

1. size($G_k(1)$) ≥ 1.
2. size($G_1(\chi)$) $\geq$ size($G_{\log m}(\chi - 1)$).
3. size($G_{k+1}(\chi)$) $\geq 16(2 *$ size($G_k(\chi)$) + size($G_{2 \log m}(\chi - 1)$)).

It is easily checked that the formula in the statement of the claim does satisfy all three requirements. □

To finish the proof of the theorem we look at $G_{\log m}(\chi)$. With probability at least a $\frac{1}{2}$, the number of pigments used by $\mathscr{A}$ is $\geq \log^{\chi-1} m/(\chi - 1)$ and the number of vertices used in the construction is $\leq m^{12(\chi+1)}$. So given n, the number of vertices, we pick t such that $2^{12t(\chi+1)} \leq n < 2^{12(t+1)(\chi+1)}$. Thus, if n denotes the number of vertices in the graph then we obtain a lower bound of $\Omega((1/(\chi-1))((\log n/(12(\chi+1))) - 1)^{\chi-1})$ for the expected number of colors used. We obtain a slightly better bound for the deterministic case, since we do not need to take 16 copies from the family $\mathscr{F}$.

4. Conclusion

We have described a randomized algorithm that colors graphs of ounded chromatic number online relatively well. Our algorithm runs in olynomial time. This result compares well with the best known bound for olynomial-time offline graph coloring, $O(n^{1-1/(\chi-4/3)}\log^{8/5} n)$, due to Avrim Blum [3].

We also proved a lower bound on $\Omega((1/(\chi-1))(\log n/12(\chi+1)-)^{\chi-1})$ for randomized online coloring, improving even the known deterministic lower bounds. The gap between the lower and upper bounds is till quite large, and we hope that this paper will encourage further work n this area and shrink this gap. It is intriguing to ask whether it helps to pend an exponential time to color each vertex.

Acknowledgments

I am indebted to Márió Szegedy for generously spending time with me, helping me with his work despite being under time pressure to write his thesis. I am grateful to Laci Babai nd Howard Karloff for their interest, encouragement, and for their perceptive comments on arly drafts. Their extremely thorough comments have significantly improved the presentaion of this paper. I thank Noga Alon for letting me know of his results that actually notivated this work and, of course, Hal Kierstead for bringing this problem to my attention a ong time ago.

References

1. N. Alon, personal communication, September 1989.
2. D. Bean, Effective coloration, *J. Symbolic Logic*, June (1976), 469–480.
3. A. Blum, An $O(n^{0.4})$-approximation algorithm for 3-coloring, *in* "Proceedings, 21st Annual ACM Symposium on the Theory of Computing, 1989," pp. 535–542.
4. A. Blum, Some tools for approximate 3-coloring, *in* "Proceedings, 31st Annual Symposium on the Foundations of Computer Science, 1990," pp. 554–562.
5. M. Halldórsson and M. Szegedy, personal communication, 1991.
6. M. M. Halldórsson, "Frugal Methods for the Independent Set and Graph Coloring Problems," Ph.D. thesis, Rutgers, The State University of New Jersey, 1991.
7. S. Irani, Coloring inductive graphs on-line, *in* "Proceedings, 31st Annual Symposium on the Foundations of Computer Science, 1990," pp. 470–479.
8. H. Karloff, personal communication, May 1990.
9. L. Lovász, M. Saks, and W. T. Trotter, An online graph coloring algorithm with sublinear performance ratio, *Discrete Math.* **75** (1989), 319–325.
0. M. Szegedy, personal communication, March 1989.
1. A. C. Yao, Probabilistic computations: Towards a unified measure of complexity, *in* "Proceedings, 18th Annual Symposium on the Foundations of Computer Science, 1977," pp. 222–227.