Online Coloring Known Graphs

Magnús M. Halldórsson
Science Institute – University of Iceland
IS-107 Reykjavik, Iceland
mmh@hi.is, www.hi.is/~mmh.

Submitted: September 13, 1999; Accepted: February 24, 2000.

Subject Classification: 05C15, 68W25.
Keywords: graph coloring, online algorithms.

Abstract

The problem of online coloring an unknown graph is known to be hard. Here we consider the problem of online coloring in the relaxed situation where the input must be isomorphic to a given known graph. All that foils a computationally powerful player is that it is not known to which sections of the graph the vertices to be colored belong. We show that the performance ratio of any online coloring algorithm with *advance knowledge of the input graph* is at least $\Omega(N/\log^2 N)$, where N is the number of vertices. This matches and generalizes the bound for the case of an unknown input graph. We also show that any online independent set algorithm has a performance ratio at least $N/8$.

1 Online Graph Coloring

Graph coloring is the problem of assigning the fewest possible colors to the vertices of a graph so that adjacent vertices receive different colors. In the *online* version, the input graph is given only one vertex at a time, along with the edges to previous vertices, and the algorithm must irrevocably color the given vertex before receiving later ones.

The Known-Graph Online Model Online problems receive much of their difficulty from the fact that the input is hidden. Clearly, if the whole input sequence was given, the problem reduces to the offline version. But, what if the input was known, but not the sequence in which it was given? In the case of graphs, suppose the input had to induce a graph isomorphic to a graph known in advance, but the identification of the presented vertices was not given. Can we then color the graph relatively easily?

Formally, consider the following game for two players, Alice and Bob.

1. Alice presents Bob with a graph G on N vertices.

2. For $i = 1, \dots, n$
   a) Alice presents a vertex v_i with all edges from v_i to $v_1, \dots, v_{i-1}$. The graph induced by vertices $v_1, \dots, v_i$ is isomorphic to a subgraph of G.
   b) Bob responds with a color $c(v_i)$ that is distinct from the color of any vertex v_j adjacent to v_i, $j < i$.

In ordinary parlance, Bob an algorithm, while Alice is the adversary. Time complexity, however, is not the main concern in the online scenario, and since the total number of possible game sequences is bounded by $2^{poly(N)}$, we may as well consider Bob to be as Alice's equal.

The input graph G is referred to as the *underlying graph*, while the subgraph that is incrementally given to the algorithm is the *presented graph*. The former has N vertices, while the latter contains n vertices.

The value of the game is the number of colors used by the algorithm (Bob) divided by the chromatic number of G (the number of colors in an optimal offline coloring). The worst-case game value, over all possible adversaries, is referred to as the *performance ratio*, or the *competitive ratio* of the algorithm.

Results We show that the competitive ratio of any coloring algorithm, even with full knowledge of the input graph G, is $\Omega(N/\log^2 N)$, where N is the number of vertices of G. This generalizes the results of [6] for the known graph case. In fact, for each N, we construct *one particular graph* that is hard to color that contains the whole class of graphs given in [6] as induced subgraphs (each with about $N/\log N$ vertices).

The same lower bound holds also for randomized algorithm against an *oblivious adversary*; in this version, Bob is allowed to base his color decisions on coin flips which are hidden from Alice. It also holds for a further restricted model, where Alice immediately commits a presented vertex to be a particular vertex in the underlying graph as soon as Bob decides on his color. This implies that Alice is effectively coloring the vertices online as well, and revealing those choices.

Related results Our results are an immediate generalization of those of Halldórsson and Szegedy [6] for the ordinary online coloring of an unknown graph. They constructed a class of $\log n$-colorable graphs that require at least $n/\log n$ colors online in the worst case [6]. The best performance ratios known are $O(n \log\log\log n/\log\log n)$ by a deterministic algorithm [7], and $O(n/\log n)$ by a randomized algorithm against an oblivious adversary [5]. Even for the case of trees, any online algorithm requires $\log n$ colors in the worst case [3, 4]. In fact, the construction of Gyárfás and Lehel [4] holds also for our model of a known input graph.

Bartal et al. [2] considered a different version of online coloring a known graph. In their model, each presented vertex is identified on arrival with one of the underlying vertices. The difficulty then lies in the fact that the algorithm does not know which subset of the vertices will be presented. Also, the adversary is charged only for the chromatic number of the presented subgraph, as opposed to that of the underlying graph in our model. They prove a lower bound of $\Omega(n^{1-\log_4 3}) \approx \Omega(n^{.2})$ and an upper bound of $O(\sqrt{n})$.

Online Independent Set We consider also the problem of finding an *independent set* (or a set of mutually nonadjacent vertices) of maximum cardinality in the same online model with a known graph. We relax the problem by allowing the algorithm to color the vertices online and selecting at the end of the game the largest color class as the independent set solution.

In spite of this relaxation, we can prove the worst possible bound of $\Omega(n)$ for the performance ratio of any deterministic algorithm. However, if we allow the algorithm randomization against an oblivious adversary, the best possible performance ratio is $\theta(n/\log n)$.

2 Lower bound constructions for coloring

We shall focus on the following restriction to the game between Alice and Bob. Note that only Alice's moves are being restricted, while Bob is supplied with more information. Hence, the lower bounds proved here immediately carry over to the basic game of Section 1.

Let $[k] = \{1, 2, 3, \ldots, k\}$ denote the set of colors used by the adversary, and $[k]^t$ denote the collection of all t-element subsets of $[k]$.

1. Alice announces k, the number of colors she will use, x, a parameter, and n, the number of vertices to be presented. She then presents a collection of subsets $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ of $[k]^{k/2}$ in some order.[1] This specifies an *underlying graph* G on $N = t \cdot k/2$ vertices and a coloring $c : V(G) \mapsto [k]$ as follows. The graph consists of t blocks of $k/2$ vertices each, with each block B_i arranged in a clique and colored with the colors in C_i. Further, all vertices of B_i are identically adjacent to exactly those vertices in previous blocks whose colors are not in C_i.

   Formally, let $C_i = \{c_{i,1}, \ldots, c_{i,k/2}\}$. Then,

[1] Note that the ordering of the blocks is not important, and in fact Alice could allow Bob to choose the ordering. This implies that we implicitly give an exponential-size family of hard graphs.

$$V(G) = \{v_{i,\ell} : i = 1, \ldots, t, \ell = 1, \ldots, k/2\},$$
$$c(v_{i,\ell}) = c_{i,\ell}, \quad \text{for } i = 1, \ldots, t, \ell = 1, \ldots, k/2,$$
$$E(G) = \{(v_{i,\ell}, v_{j,q}) : i < j \text{ and } c_{i,\ell} \notin C_j\} \cup \{(v_{i,\ell}, v_{j,q}) : i = j\}.$$

2. For $i \leftarrow 1$ to t

   Alice presents x (identically looking) vertices from block B_i.
   Bob colors these x vertices, consistent with his previous coloring.
   Alice maps each of the x vertices to a different vertex of B_i.

The *oblivious adversary model* modifies step 2 as follows:

Alice randomly picks subsets S_i from each C_i, such that $|S_i| = x$.
for $i \leftarrow 1$ to t
  Bob gives colors to x (identical) vertices from B_i.
  The set S_i is revealed to Bob.

Thus Alice makes all her decisions before Bob starts coloring. This is an oblivious strategy, in that it does not depend on Bob's moves; thus it is also necessarily a *mixed* strategy, meaning that it uses randomization. Here, it is essential that Bob does not have knowledge of S_j, $j > i$, during round i. The above setup borrows many of the ideas of [6].

It is clear from the edge description that the graph G is k-colorable. Further, since at each round Alice selects a subset of the vertices of a block, the graph presented is an induced subgraph of G. The number of presented vertices is $n = t \cdot x = N \cdot 2x/k$.

The setup above is a restriction of the game described in section 1 in in several ways. The most significant is that the mapping of the presented vertex to one in the underlying graph, is indicated to Bob at the end of each round. This implicitly informs Bob of the color that Alice used for the vertex. In [6], this was referred to as a *transparent* model and shown that a matching $O(n/\log^2 n)$ performance ratio is achievable in this case. The global structure of the presented graph, including the ordering of the blocks, is also given in advance. The only uncertainty is *which x out of $k/2$ block-vertices* are being presented in each round.

We shall describe three strategies for Alice that force Bob to use many colors. A strategy involves specifying the set collection $\mathcal{C}$ and a rule for mapping presented vertices to particular vertices in the underlying graph. For the latter, we refer to underlying vertices by their preassigned colors.

Terminology We say that Bob assigns vertices to *bins* while Alice *colors*. The *hue* of a bin is the set of colors among the vertices in the bin. We say that *progress is made* on a vertex if the color assigned to that vertex increases the hue of the bin to which the vertex was assigned.

The goal of the constructions is to make progress with each vertex, or at least with a constant fraction of the vertices. Observe that if that goal is achieved, Bob is forced to use $\Omega(n/k)$ bins, since at most $k/2$ vertices in the same bin can make progress. That yields a $\Omega(n/\log^2 n)$ approximation ratio. If x is additionally $\Omega(k)$, the ratio is also $\Omega(N/\log^2 N)$.

Simple lower bound Let $\mathcal{C} = [k]^{k/2}$, and let $x = 1$. Namely, the set collection consists of all subsets of size $k/2$. From each block, Alice presents a single vertex, whose color she chooses so as to increase the hue of the bin used by Bob. We now show that such a choice is always possible.

Lemma 1 *Alice can make progress with every presented vertex.*

Proof. Consider a vertex presented from block i, and let H be the hue of the bin used. By the definition of the edges of the graph, H must be a subset of C_i. Further, H must also be a subset of the admissible colors for the last vertex added to the bin, which is the set C_j, for some $j < i$. Since the intersection of C_j and C_i contains at most $k-1$ vertices, H is a proper subset of C_i. Thus, there always exists an element in $C_i - H$, and Alice makes progress by assigning the vertex such a color. $\square$

We observe that the subgraph produced is precisely the lower bound construction of [6]. A lower bound of $\Omega(N/\log^3 N)$ on the performance ratio of any deterministic online coloring algorithm now follows.

This construction has the elegant property that the presented vertices appear in the same order as they occur in the underlying graph. Thus, the presented vertices form a subsequence of the ordered set of vertices of the underlying graph.

Optimal lower bound We want the set collection to have certain desirable properties. We call a collection *dispersed* if any pair of sets have at most $3k/8$ elements in common. Thus, we seek a binary code of length k with Hamming distance $k/4$. As the following lemma shows, there exists a dispersed collection of exponential size.

Lemma 2 *There is a dispersed collection of sets from* $[k]^{k/2}$ *with at least* 1.1^k *elements.*

Proof. Consider the following greedy algorithm. Pick any set S from $[k]^{k/2}$ and add to collection. Eliminate from further consideration S and all sets that agree with S in at least $3k/8$ elements. Repeat until no sets remain. The number of sets that differ from S in $2i$ elements equals the number of ways that i elements can be removed from S, times the number of ways that i elements outside S can be added, or $\binom{k/2}{i}^2$. Hence, the total number of sets eliminated from consideration in each round is at most $\frac{k}{2}\binom{k/2}{k/8}^2$, and the number of rounds is at least

$$\frac{\binom{k}{k/2}}{\frac{k}{2}\binom{k/2}{k/8}^2} \geq 1.1^k. \qquad \square$$

Alice chooses $\mathcal{C}$ to be a dispersed collection as above, and x to be $k/8$. In the adaptive construction, the colors of the vertices are chosen so that she makes progress on all of them (which we show possible below). In the oblivious construction, each S_i is chosen uniformly randomly among the subsets of C_i of cardinality $k/8$.

Lemma 3 *Each vertex is assigned to a bin with a hue of at most $3k/8$ elements.*

Proof. The vertices of a block B_i must be assigned to different bins since they form a clique. Let H be the hue of a bin to which a vertex is to be assigned. H must be a subset of C_i. H must also be a subset of the set of admissible colors the admissible colors of the last vertex added to that bin, which corresponds to C_j, for some $j < i$. Since $\mathcal{C}$ is a dispersed collection, $|H| \leq |C_i \cap C_j| \leq 3k/8$. □

Lemma 4 *There is an adaptive strategy for Alice that makes progress on every vertex.*

Proof. Each vertex gets assigned to a bin whose hue contains at most $3k/8$ elements, in which case the remaining at least $k/8$ colors make progress. Thus, regardless of how the other $k/8 - 1$ vertices of the block are colored, there is always a color available for v that makes progress. □

Lemma 5 *There is an oblivious strategy for Alice that makes progress on at least one fourth of the vertices.*

Proof. All the $k/2$ colors of C_i are equally likely to be assigned to a particular vertex v of B_i. There are at least $k/8$ out of $k/2$ colors that increase the hue of the bin to which v is assigned. Thus, the probability of making progress with a particular vertex is at least $1/4$. By linearity of expectation, the expected number of vertices making progress is thus at least $n/4$. □

Lemmas 4 and 5 immediately lead to strong lower bounds on Bob's performance.

Theorem 6 *$\Omega(N/\log^2 N)$ is a lower bound on the performance ratio of any online coloring algorithm, even when the input graph is given in advance. It holds also for randomized algorithms against an oblivious adversary.*

Proof. In the example constructed, $N \geq \frac{k}{2}1.1^k$ and $n = N/2$. The algorithm uses at least expected $(n/4)/(k/2) = n/k = \Omega(N/k)$ colors, while the adversary uses k. Since $k = O(\log N)$, the performance ratio is at $\Omega(N/\log^2 N)$. □

Remark: The deterministic construction has the powerful property that every color class of any coloring found by an online algorithm contains at most $k/2 = \Omega(\log n)$ vertices. This implies e.g. that the lower bound holds also for the online version of the *Sum Coloring* problem studied in [1], where the objective is to find a coloring with the positive integers so that the sum of the values assigned to the vertices is minimized. Also, note that Alice's strategy is efficiently computable in time $O(n)$ per vertex.

3 Online Independent Set Problem

We now consider the problem of finding a large independent set online, when the original graph is given in advance. The algorithm (Bob) is allowed to form a coloring of the vertices, and choose the largest color class at the end as the final solution. Clearly, this allows for a considerable flexibility over requiring Bob to irrevocably assign vertices to a single set. This version faithfully represent “onlineness” in that the algorithm must irrevocably decide for every pair of nodes if they are to be in the same independent set, and models somewhat more intelligent online strategies.

In this problem, the adversary is also evaluated in terms of the presented subgraph, i.e. the independence number of the subgraph. However, we are interested in bounding the performance ratio as a function of the size of the original graph. From the remark at the end of the last section, it is clear that the $\Omega(N/\log^2 N)$ lower bound holds also for the independent set problem. However, we can actually show the tightest possible $\Omega(N)$ lower bound.

Adaptive construction The underlying graph contains $N = 2n$ vertices, a_i, b_i, $i = 1, 2, \ldots, n$, with the a-vertices forming an independent set, b-vertices forming a clique, and a_i adjacent to b_j iff $i > j$. In a sense, a_i and b_i form a block of two identically looking vertices; the a vertices are *good*, belonging to the optimal independent set, while the b vertices are *bad*, excluding any further addition of vertices to the same bin.

Alice presents n vertices and, depending on the algorithm's actions, maps the i-th presented vertex either to a_i or b_i. Namely, if the vertex is placed in an empty bin, then it will be declared to be a_i; if placed in a nonempty bin, it will be b_i.

Theorem 7 *The performance ratio of any online independent set algorithm is at least $N/8$, even if the graph on N vertices is given in advance.*

Proof. If a vertex is placed in a nonempty bin, all remaining vertices will be adjacent to it. Thus, by induction, bins accumulate at most two vertices. On the other hand, at least half of the vertices will be a-vertices; hence, the size of the adversary's solution is at least $n/2$. The performance ratio is therefore at least $n/4 = N/8$. □

We can obtain a matching performance ratio using the ubiquitous First-Fit algorithm that assigns each vertex in sequence to the first bin possible. Every bin, except possibly the last one, must contain a node not in a particular optimal independent set. If First-Fit outputs t nodes, it uses at least $(n-1)/t + 1$ bins and the optimal solution contains at most $(t-1)(n-1)/t + 1$ nodes. Thus, First-Fit's performance ratio is at most $((t-1)n + 1)/t \leq n/4 + 1/2$.

Oblivious construction For the oblivious case, we again use the same underlying graph and present $n = N/2$ nodes. Here, we let the i-th presented node be *good* (i.e. a_i) with probability 1/2 and *bad* (i.e. b_i) with probability 1/2. With high probability the sequence will contain $n/2(1-o(1))$ good nodes. For a particular bin to accumulate nodes, all but possibly the last node must be good. The events of nodes being good are independent. Thus, the probability that a given bin collects more than t nodes is at most 2^{-t}. Hence, the probability that all the (at most n) bins contain $2\log n$ or fewer nodes, is at least $1 - n \cdot 2^{-2\log n} = 1 - 1/n$. Therefore, the expected size of the largest independent set found is $O(\log n)$. Thus, the performance ratio of any randomized algorithm is $\Omega(N/\log N)$, expected and with high probability.

A matching $O(n/\log n)$ performance ratio can be obtained by a simple randomized algorithm, even without knowledge of the input graph. We describe here an algorithm for the case when the number of vertices n and the independence number α are known in advance. The unknown case can be handled via a doubling technique, see [5].

Without loss of generality, $\alpha \geq n/\log n$. Let $q = (1/2)\log_{n/\alpha} n$. Before coloring, randomly partition the n vertices into $\lfloor n/q \rfloor$ blocks, each of size q, ignoring the remainder. Greedily color the vertices of each block independently. With probability $p = \alpha/n$, a random vertex is contained in the maximum independent set. Thus, with probability at least $p^q = 1/\sqrt{n}$ all the vertices of a given block form an independent set. The probability that none of the blocks are independent is thus at most

$$\left(1 - \frac{1}{\sqrt{n}}\right)^{n/q} \leq e^{-\sqrt{n}/\log n}.$$

With high probability, some block is independent, and thus the expected size of the independent set found is $\Omega(q)$. The performance ratio is then $O(\alpha/\log_{n/\alpha} n) = O(n/\log n)$.

We summarize the above with the following theorem.

Theorem 8 *The performance ratio of any randomized online independent set algorithm against an oblivious adversary is at least $\Omega(N/\log N)$, even if the graph on N vertices is given in advance. There exists a randomized online algorithm that attains this ratio, even without knowledge of the input graph.*

Observe that our constructions have both the transparency property of Section 2, as well as the property of the presented vertices forming a subsequence of the underlying vertex set.

References

[1] Amotz Bar-Noy, Mihir Bellare, Magnús M. Halldórsson, Hadas Shachnai, and Tami Tamir. On chromatic sums and distributed resource allocation. *Information and Computation*, **140**(2), February 1998.

[2] Yair Bartal, Amos Fiat, and Stefano Leonardi. Lower bounds for on-line graph problems with application to on-line circuit and optical routing. In *Proc. 28th Ann. ACM Symp. on Theory of Computing*, (1996), 531–540.

[3] Dwight Bean. Effective coloration. *J. Symbolic Logic*, **41** (1976), 469–480.

[4] A. Gyárfás and J. Lehel. On-line and first fit colorings of graphs. *J. Graph Theory*, **12**(2) (1988), 217–227.

[5] Magnús M. Halldórsson. Parallel and on-line graph coloring. *J. Algorithms*, **23** (1997), 265–280.

[6] Magnús M. Halldórsson and Márió Szegedy. Lower bounds for on-line graph coloring. *Theoretical Computer Science*, **130** (1994), 163–174.

[7] Hal Kierstead. On-line coloring k-colorable graphs. *Israel J. of Math* **105** (1998), 93–104.