Discrete Mathematics 75 (1989) 319–325
North-Holland

AN ON-LINE GRAPH COLORING ALGORITHM WITH SUBLINEAR PERFORMANCE RATIO

László LOVÁSZ
Department of Computer Science, Eötvös University, Budapest, Hungary and Department of Computer Science, Princeton University, Princeton, NJ 08544, U.S.A.

Michael SAKS
Department of Mathematics and RUTCOR, Rutgers University, New Brunswick, NJ 08903 and Bell Communications Research, Morristown, NJ 07960. Supported in part by NSF grant DMS87-03541 and Air Force Office of Scientific Research grant AFOSR-0271.

W.T. TROTTER
Department of Mathematics, Arizona State University, Tempe, Arizona 85287. Research supported in part by NSF grant DMS 87–13994.

One of the simplest heuristics for obtaining a proper coloring of a graph is the **First-Fit** algorithm: Fix an arbitrary ordering of the vertices and, using the positive integers as the color set, assign to each successive vertex the least integer possible (keeping the coloring proper). This is an example of an *on-line* algorithm for graph coloring. In the on-line model, a graph is presented one vertex at a time. Each new vertex is given together with all edges joining it to previous vertices. An on-line coloring algorithm assigns a color to each vertex as it is received and once assigned, the color cannot be changed. The *performance function*, $\rho_A(n)$, of an on-line algorithm A is the maximum over all graphs G on n vertices of the ratio of the number of colors used by A to color G to the chromatic numbers of G. The **First-Fit** algorithm has performance function $n/4$. We exhibit an algorithm with sublinear performance function.

1. Introduction

One of the simplest heuristics for obtaining a proper coloring of a graph is the **First-Fit** algorithm: Fix an arbitrary ordering of the vertices and, using the positive integers as the color set, assign to each successive vertex the least integer possible, subject to maintaining a proper coloring.

This is an example of an *on-line* algorithm for graph coloring. In the on-line model, a graph is presented one vertex at a time. Each new vertex is given together with all edges joining it to previous vertices. An on-line coloring algorithm assigns a color to each vertex as it is received and, once assigned, the color cannot be changed.

If A is any graph coloring algorithm then, for a graph G, $\chi_A(G)$ denotes the number of colors that A uses to color G. The *performance ratio* of A on G, denoted $\rho_A(G)$, is $\chi_A(G)/\chi(G)$, i.e. the ratio of the number of colors used by A to the number of colors in an optimal coloring of G. The *performance function* of

0012-365X/89/$3.50 © 1989, Elsevier Science Publishers B.V. (North-Holland)

A, denoted $\rho_A(n)$, is defined for each integer n to be the maximum of $\rho_A(G)$ over all n vertex graphs.

Various researchers (see e.g. [1, 7]) have shown that no on-line algorithm A has bounded performance function on the class of all graphs; indeed, for any on-line algorithm A, there are trees on n vertices for which that algorithm requires at least $1+\log_2 n$ colors. On the other hand, $\rho_A(n) \leq n$ for any algorithm. **First-Fit** does not do much better than this trivial bound even on the class of bipartite graphs since for any integer k, there exists an on-line bipartite graph on $2k$ vertices for which **First-Fit** requires k colors (the graph is the bipartite complement of a perfect matching). Recently, Szegedy [19] showed that for any on-line algorithm A and integer k, there is a graph on at most $k(2^k-1)$ vertices having chromatic number k, but for which the algorithm A requires 2^k-1 colors. Thus the performance function for any on-line algorithm A grows at least as fast as $n/(\log n)^2$. It is natural to ask whether there is any on-line algorithm that has a sublinear performance function. In this note we settle this question in the affirmative by proving:

Theorem 1. *There exists an on-line coloring algorithm* Color *with* $\rho_{\text{Color}}(n) = (2n/\log^* n)(1+o(1))$.

Note that in this paper all logarithms are taken to the base 2, and, as usual, $\log^* n$ is the smallest k for which the k times iterated logarithm, $\log^{(k)} n = \log\cdots\log n$ is at most 1. The algorithm **Color** is constructed recursively from an algorithm **Partition*** that partitions the vertex set into subsets each having clique number strictly smaller than the input graph. This kind of recursive construction was used by Wigderson ([21]) to obtain a polynomial (but not on-line) approximate coloring algorithm with performance ratio $n(\log\log n)^2/(\log n)^2$.

Previous researchers have considered the behavior of on-line coloring algorithms on restricted classes of graphs. The performance function of an algorithm A with respect to a class $\boldsymbol{G}$ of graphs $\rho_A(n; \boldsymbol{G})$, is the maximum of $\rho_A(G)$ over all n vertex graphs in the class $\boldsymbol{G}$. It is an easy exercise to construct an on-line algorithm that achieves a performance function of $O(\log_2 n)$ on the class of bipartite graphs, which is optimal by the lower bound for trees mentioned above. For the class of interval graphs, Kierstead and Trotter [14] showed that there is an on-line algorithm A with performance ratio 3 and this is best possible. Recently, Kierstead [12] showed that **First-Fit** has bounded performance ratio on interval graphs, solving a question posed by Woodall ([22]) and Chrobak and Slusarek. Gyárfás and Lehel ([7]) showed that **First-Fit** achieves bounded performance ratio on split graphs, complements of bipartite graphs and complements of chordal graphs.

On-line algorithms have been investigated in the context of several combinatorial optimization problems, including various problems related to dynamic data structures [2, 3, 8, 10, 16, 18, 20], task systems and server problems

[4, 15, 5, 6], bin packing [9], partitioning a partial order into chains [11], and representing a partial order by vectors in d-space [13]. We believe that many other problems can and should be analyzed from the perspective of on-line algorithms.

2. An on-line graph coloring algorithm

An *on-line graph* is an undirected graph G on a totally ordered vertex set V. For any vertex v, let $[v]$ denote the set of vertices preceding or equal to v. The *pre-neighborhood* of a vertex v, $N^-(v)$ is its neighborhood in $[v]$. The *pre-degree* of v, $d^-(v)$ is the size of its pre-neighborhood.

An *on-line graph partitioning algorithm* is a procedure that constructs a partition Π of the vertex set of G by considering each vertex of G in order and assigning it to one of the previous blocks or creating a new singleton block, without reassigning any of the previously assigned vertices. Such an algorithm is a coloring algorithm if it partitions the vertex set into independent sets. More formally, an on-line partitioning algorithm is a map which associates to each graph G in a class $\boldsymbol{G}$ a partition Π_G of the vertex set, in such a way that for each graph $G \in \boldsymbol{G}$ and each vertex v, the partition of $[v]$ induced by Π_G depends only on the graph induced by G on $[v]$.

The on-line coloring algorithm given here is constructed recursively from an on-line algorithm called **Partition**(n, d), where n is an upper bound on the total number of vertices and d is a positive integer bounded above by n. This algorithm partitions the vertex set V of the on-line graph G into sets $D_1, D_2, \ldots, D_d, C_1, C_2, \ldots, C_r$. Each set D_i is independent and is called a *first-fit set,* and each set C_i is contained in the neighborhood of some vertex and is called a *residual set.*

Let $\epsilon_1 = d/n$ and for $i > 1$, let $\epsilon_i = \epsilon_{i-1}^2/2$. Thus

$$\epsilon_i = 2\left(\frac{d}{2n}\right)^{2^{i-1}}.$$

Say that a subset S of vertices of size s is *legal* if the intersection of the pre-neighborhoods of its members has size at least $\epsilon_s n$. **Partition**(n, d) is defined as follows. Initially $r = 0$ and $D_1, \ldots, D_d$ are each empty. For each arriving vertex v: if $v \cup D_i$ is independent for some i then add v to such D_i. Otherwise if $C_j \cup v$ is legal for some residual set C_j, add v to such a set having maximum size. If there is no such set, increase r by 1 and let $C_r = \{v\}$.

The key property of the algorithm is:

Lemma 2. *For $n \geq 4$ and $d \geq n/\log\log n$, at most $4n/\log\log n$ residual sets are created by* **Partition**(n, d) *on input of any graph having n or fewer vertices.*

We begin with a simple combinatorial lemma.

Lemma 3. *Let δ be a constant with $0<\delta<\frac{1}{2}$. Let $S_1, S_2, \ldots, S_q$ be subsets of a set S such that $|S_i| \geqslant \delta\,|S|$ for all i and $|S_i \cap S_j| < \delta^2\,|S|/2$, for $i \neq j$. Then $q<2/\delta$.*

Proof. Suppose, to the contrary that $q \geqslant 2/\delta$ and let $j = \lceil 2/\delta \rceil$. For $1 \leqslant i \leqslant j$, let $T_i = S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})$. Then the sets $T_1, T_2, \ldots, T_j$ are disjoint and $|T_i| \geqslant |S_i| - |S_i \cap S_1| - |S_i \cap S_2| - \cdots - |S_i \cap S_{i-1}| > \delta(1-\delta(i-1)/2)\,|S|$. Hence

$$|S| \geqslant |T_1 \cup T_2 \cup \cdots \cup T_j| = |T_1| + |T_2| + \cdots + |T_j| > \delta j(1-\delta(j-1)/4)\,|S| > |S|,$$

a contradiction, establishing the lemma. □

Lemma 4. *Let G be an on-line graph with at most n vertices. Then when* **Partition**(n, d) *terminates, at most $2/\epsilon_t$ residual sets have size t.*

Proof. For each $j \in \{1, \ldots, r\}$ let A_j be the intersection of the pre-neighborhoods of all of the vertices in C_j. By the definition of the algorithm, if C_j has size t, A_j has size at least $\epsilon_t n$. Furthermore, we claim that if two sets C_i and C_j both have size t, $|A_i \cap A_j| < \epsilon_{t+1} n = n\epsilon_t^2/2$, which by Lemma 3, with $\delta = \epsilon_t$ finishes the proof. To prove the claim, let v be the last vertex added to either C_i or C_j and suppose it was added to C_j. Before it was added, C_i had size t and C_j had size $t-1$. Since v was added to a set having smaller cardinality than C_i, $C_i \cup v$ is not legal. This implies that the intersection of the pre-neighborhood of v with A_i has fewer than $\epsilon_{t+1} n$ elements. Since A_j is contained in the pre-neighborhood of v, the claim is established. □

Proof of Lemma 2. For any integer $k \geqslant 2$, the number of residual sets that have size at least k at most n/k. By Lemma 4, the number of residual sets of size less than k is at most $2(1/\epsilon_1 + 1/\epsilon_2 + \cdots + 1/\epsilon_{k-1})$ which is bounded above by $1/\epsilon_k$. Hence the number of residual sets is at most $n/k + 1/\epsilon_k$. Taking $k = \log\log n/2$ yields $n/k + 1/\epsilon_k \leqslant 2n/\log\log n + 2(\log\log n)^{\sqrt{\log n}} \leqslant 4n/\log\log n$. □

Next, **Partition** is used to construct a second partitioning algorithm called **Partition***. For $k \geqslant 2$, let n_k be the largest integer such that $n_k/\log_2\log_2 n_k \leqslant 2^k$ (e.g. $n_2 = 4$ and $n_3 = 16$). The on-line algorithm **Partition*** takes as input any graph and produces a union of disjoint partitions as follows: Place incoming vertices in a single class until the first vertex is received that has a neighbor in that class. Starting with that vertex, apply **Partition**$(n_2, 4)$ to the first n_2 vertices. Apply **Partition**$(n_3, 8)$ to the next $n_3 - n_2$ vertices, and in general, apply **Partition**$(n_k, 2^k)$ to vertices $\{n_{k-1}+1, \ldots, n_k\}$.

Lemma 5. *Suppose* **Partition*** *is applied to an on-line graph G on n vertices. If G is an independent set then* **Partition*** *produces a single class. Otherwise it produces* *at most* $20n/\log\log n$ *sets, and each has chromatic number strictly less than that of* G.

Proof. The behavior of **Partition*** on an independent set is apparent from the definition. On a general graph, every set produced is either a first-fit set (which is independent) or a residual set, which lies in the neighborhood of some vertex of G, and thus has chromatic number strictly less than that of G.

It remains to bound the number of sets created. Let k be the least index such that $2^{k-1} < n/\log\log n \leqslant 2^k$. For any i, at most $n_i - n_{i-1}$ vertices are partitioned by **Partition**$(n_i, 2^i)$. This results in at most 2^i first-fit sets and, by Lemma 2, at most $4n_i/\log\log n_i$ residual sets, for a total of at most $5(2^i)$ sets. Thus, the total number of sets created by **Partition*** is at most $(1 + 5(2^2) + 5(2^3) + 5(2^4) + \cdots + 5(2^k)) \leqslant 10(2^k) \leqslant 20n/\log\log n$ sets. □

Finally the algorithm **Color** is defined recursively from **Partition***: Run **Partition*** on G. For each class besides the first (independent) class produced, color it by a recursive call to **Color**.

It is easily shown by induction on the number of vertices of G, that **Color** partitions any input graph into independent sets. Define $c(n, k)$ to be the maximum number of colors used by **Color** to color an input graph on n fewer vertices and chromatic number at most k. Obviously $c(n, 1) = 1$ and $c(n, k) \leqslant n$. Define $h^{(0)}(n) = n$, $h(n) = h^{(1)}(n) = \max\{1, \log\log n/20\}$, and for $k \geqslant 2$, $h^{(k)}(n) = h^{(k-1)}(h(n))$. Note that for all k and n positive, $h^{(k)}(n)$ is a concave function of n.

Theorem 6. $c(n, k) \leqslant n/h^{(k-1)}(n)$ *for* $k \geqslant 1$ *and* $n \geqslant 1$.

Proof. We prove the result by induction on k; the case $k = 1$ is trivial. Suppose **Color** is applied to a graph on n vertices having chromatic number k. By Lemma 5, the main call **Partition*** produces at most $n/h(n)$ classes each having chromatic number at most $k - 1$. Each of these is recursively colored using **Color**. Thus the number of color classes created can be bounded above by:

$$
\begin{aligned}
c(n, k) &\leqslant \max_{\substack{t \leqslant n/h(n)\\ n_1+n_2+\cdots+n_t=n\\ n_i \geqslant 1}} \sum_{i=1}^{t} c(n_i, k-1)\\
&\leqslant \max_{\substack{t \leqslant n/h(n)\\ n_1+n_2+\cdots+n_t=n\\ n_i \geqslant 1}} \sum_{i=1}^{t} n_i/h^{(k-2)}(n_i),
\end{aligned}
$$

by the induction hypothesis. Since $n/k^{(k-2)}(n)$ is a convex function of n, the right hand side is bounded above by taking all of the n_i's to be equal and t to be as large as possible, yielding an upper bound of $n/h(n)\{h(n)/h^{(k-2)}(h(n))\} = n/h^{(k-1)}(n)$. □

Corollary 7. *The algorithm* **Color** *has a worst case performance ratio of at most* $(2n/\log^* n)(1 + o(1))$.

Proof. By Theorem 6, the performance ratio of color on a graph on n vertices with chromatic number k is most $n/kh^{(k-1)}(n)$, which is maximized for fixed n when k is the least index with $h^{(k-1)}(n) = 1$ (or, possibly 1 less than that) and this index is asymptotically equal to $\log^* n/2$. □

Note that the performance ratio of $O(n/\log^{(2k-2)}(n))$ for graphs of chromatic number bounded by k can be improved(!) to $O(n \log^{(2k-3)}(n)/\log^{(2k-4)}(n))$ for $k \geq 3$ by modifying **Partition*** as follows. Initially, instead of constructing a single independent set, apply a bipartite graph coloring algorithm using at most $O(\log |V|)$ colors, switching to **Partition**$(n_2, 4)$ only when a vertex is received that cannot be processed by that algorithm. Of course this does not improve the worst case performance ratio over general graphs.

Thus for graphs of chromatic number at most 3, there is an algorithm that achieves performance ratio $O(n/\log \log n)$. On the other hand, the only known lower bound is the $\Omega(\log n)$ lower bound for trees mentioned in the introduction (note that Szegedy's lower bound is not useful for graphs of bounded chromatic number). It would be interesting to close the gap.

References

[1] D. Bean, Effective coloration, J. Symbolic Logic 41 (1976) 469–480.

[2] J.L. Bentley and C.C. McGeoch, Worst-case analyses of self-organizing sequential search heuristics, Communications of ACM, to appear.

[3] J.R. Bitner, Heuristics that dynamically organize data structures, SIAM J. Comp. 8 (1979) 82–110.

[4] A. Borodin, N. Linial and M. Saks, An online algorithm for metrical task systems, Proc. 19th Annual ACM Symp. on Theory of Computing (1987) 373–382.

[5] R.R.K. Chung, R.L. Graham and M. Saks, Dynamic search in graphs, in Discrete Algorithms and Complexity (Academic Press, 1987).

[6] F.R.K. Chung, R.L. Graham and M. Saks, A dynamic location problem for graphs, Preprint.

[7] (a) Gyárfás and J. Lehel, On-line and first-fit colorings of graphs, J. Graph Theory, to appear.

[8] C. Gonnet, J.I. Munro and H. Suwanda, Toward self-organizing search heuristics, Proc. 20th IEEE Symp. Foundations of Comp. Sci. (1979) 169–174.

[9] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey and R.L. Graham, Worst case performance bounds for simple one-dimensional bin packing algorithms, SIAM J. Computing 3 (1974) 299–325.

[10] A.R. Karlin, M.S. Manasse, L. Rudolph and D.D. Sleator, Competitive snoopy catching, Proc. 27th IEEE Symp. Foundations of Comp. Sci. (1986) 244–254.

[11] H.A. Kierstead, An effective version of Dilworth's theorem, Trans. Amer. Math. Soc. 268 (1981) 63–77.

[12] H.A. Kierstead, The linearity of first-fit colorings of interval graphs, SIAM J. on Discrete Math., to appear.

[13] H.A. Kierstead, G.F. McNulty and W.T. Trotter, A theory of recursive dimension for ordered sets, Order 1, 67–82.

[14] H.A. Kierstead and W.T. Trotter, An extremal problem in recursive combinatorics, Congressus Numerantium 33 (1981) 143–153.
[15] M. Manasse, L. McGeoch and D. Sleator, Competitive algorithms for on-line problems, Proc. 20th Annual Symp. on Theory of Computing (1988).
[16] R. Rivest, On self-organizing sequential search heuristics, CACM 19 (1976) 63–67.
[17] J.H. Schmerl, Recursion theoretic aspects of graphs and orders, in Graphs and Order, I. Rival, ed. (D. Reidel, 1984), 467–484.
[18] D. Sleator and R. Tarjan, Amortized efficiency of list update and paging rules, CACM 23 (1985) 202–208.
[19] M. Szegedy, personal communication.
[20] R.E. Tarjan, Amortized computational complexity, SIAM J. Alg. Disc. Methods 6 (1985) 306–318.
[21] A. Wigderson, Improving the performance guarantee for approximate graph coloring, J. ACM 30 (1983) 729–735.
[22] D.R. Woodall, Problem no. 4, Combinatorics (Proc. British Combinatorial Conference), London Math. Soc. Lecture Note Series 13, T.P. McDonough and V.C. Marvon, Eds. (Cambridge University Press, 1974) 202.