

Max-Coloring and Online Coloring with Bandwidths on Interval Graphs

SRIRAM V. PEMMARAJU

University of Iowa,

RAJIV RAMAN

Max-Planck-Institut für Informatik

and KASTURI VARADARAJAN

University of Iowa

Given a graph $G = (V, E)$ and positive integral vertex weights $w : V \rightarrow \mathbf{N}$, the *max-coloring problem* seeks to find a proper vertex coloring of G whose color classes C_1, C_2, \dots, C_k , minimize $\sum_{i=1}^k \max_{v \in C_i} w(v)$. This problem, restricted to interval graphs, arises whenever there is a need to design dedicated memory managers that provide better performance than the general purpose memory management of the operating system.

Though this problem seems similar to the dynamic storage allocation problem, there are fundamental differences. We make a connection between max-coloring and on-line graph coloring and use this to devise a simple 2-approximation algorithm for max-coloring on interval graphs. We also show that a simple first-fit strategy, that is a natural choice for this problem, yields an 8-approximation algorithm. We show this result by proving that the first-fit algorithm for on-line coloring an interval graph G uses no more than $8 \cdot \chi(G)$ colors, significantly improving the bound of $26 \cdot \chi(G)$ by Kierstead and Qin (*Discrete Math.*, 144, 1995). We also show that the max-coloring problem is NP-hard.

The problem of online coloring of intervals with bandwidths is a simultaneous generalization of online interval coloring and online bin packing. The input is a set \mathcal{I} of intervals, each interval $i \in \mathcal{I}$ having an associated *bandwidth* $b(i) \in (0, 1]$. We seek an online algorithm that produces a coloring of the intervals such that for any color c and any real r , the sum of the bandwidths of intervals containing r and colored c is at most 1. Motivated by resource allocation problems, Adamy and Erlebach (*Proceedings of the First International Workshop on Online and Approximation Algorithms, 2003, LNCS 2909, pp 1–12*) consider this problem and present an algorithm that uses at most 195 times the number of colors used by an optimal off-line algorithm. Using the new analysis of first-fit coloring of interval graphs, we show that the Adamy-Erlebach algorithm is 35-competitive. Finally, we generalize the Adamy-Erlebach algorithm to a class of algorithms and show that a different instance from this class is 30-competitive.

The work was done when the second author was a graduate student at the University of Iowa. Sriram Pemmaraju and Rajiv Raman were partially supported by National Science Foundation Grant DMS-0213305. Kasturi Varadarajan was supported by National Science Foundation CAREER Grant CCR-0237431.

A preliminary version of this article appeared as [Pemmaraju et al. 2004] in the Proceedings of the Fifteenth annual ACM-SIAM Symposium on Discrete Algorithms(SODA), 2004. pp. 562–571.

Author’s Address: Sriram Pemmaraju and Kasturi Varadarajan, The Department of Computer Science, The University of Iowa, Iowa City, IA 52240-1419. Rajiv Raman, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg, 85, Saarbrücken, Germany-66123.

e-mail: [sriram, kvaradar]@cs.uiowa.edu, rraman@mpi-inf.mpg.de

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 2008 ACM 0000-0000/2008/0000-0001 \$5.00

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Non-numerical Algorithms and Problems

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Approximation Algorithms, Buffer Minimization, Coloring, Online Algorithms, Scheduling

1. INTRODUCTION

1.0.0.1 *Max-Coloring.* 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 problem we consider here arises in the context of designing memory managers for wireless protocol stacks like GPRS or 3G. With the rapid growth of wireless communication devices, many telecommunication companies now license their wireless protocol stacks to vendors of mobile devices. These protocols stacks have stringent memory requirements as well as soft real-time constraints and a dedicated memory manager is a natural design choice. A dedicated memory manager for these stacks must have deterministic response times and use as little memory as possible. The most commonly used memory manager design for this purpose is the *segregated buffer pool*. This consists of a fixed set of buffers of various sizes with buffers of the same size linked together in a linked list. As each memory request arrives, it is satisfied by a buffer whose size is large enough.

The assignment of buffers to memory requests can be viewed as an assignment of colors to the requests – all requests that are assigned a buffer are colored identically. Thus the problem of determining whether a given segregated buffer pool suffices for a particular sequence of allocation requests can be formalized as follows. Let $G = (V, E)$ be a graph whose vertices are objects that need memory and whose edges connect pairs of objects that are alive at the same time. Let $w : V \rightarrow \mathbf{N}$ be a weight function that assigns a natural number weight to each vertex in V . For any object v , $w(v)$ denotes the size of memory it needs. Suppose the segregated buffer pool contains k buffers with weights w_1, w_2, \dots, w_k . The problem is to determine if there is a k -coloring of G into color classes C_1, C_2, \dots, C_k such that for each i , $1 \leq i \leq k$, $\max_{v \in C_i} w(v) \leq w_i$. We call the optimization version of this problem the *max-coloring problem*. Given a graph $G = (V, E)$ and a weight function $w : V \rightarrow \mathbf{N}$ the problem is to find a proper vertex coloring C_1, C_2, \dots, C_k of G that minimizes $\sum_{i=1}^k \max_{v \in C_i} \{w(v)\}$, the weight of the coloring C_1, \dots, C_k . Note that the special case of this problem in which $w(v) = 1$ for all $v \in V$ is simply the problem of coloring a graph with fewest colors. Solving the max-coloring problem and selecting k buffers of sizes $\max_{v \in C_i} \{w(v)\}$, for $i = 1, 2, \dots, k$, leads to a segregated buffer pool that uses minimum amount of total memory. Note that this is an off-line problem. In the on-line version, buffers have to be allocated to requests as they arrive and without knowledge of future requests. The off-line version is also useful because designers of protocol stacks would like to estimate the size of the total memory block needed by extracting large traces of memory requests and running

off-line algorithms on these traces¹.

In this paper, we only consider memory allocation requests from *straight-line programs*, i.e., programs without loops or branching statements. Each memory allocation request is made for a specific duration of time and thus these requests can be viewed as intervals on a real line. Requests which are live at the same time have to be satisfied by different buffers. Thus by restricting ourselves to straight-line programs we focus on solving the max-coloring problem for interval graphs [Golombic 1980]. Focusing on the special case of straight-line programs has been a standard approach to optimization problems that arise in compiler construction. For example, for the classical *register allocation* problem [Chaitin 1982; Briggs 1992], hardness results have been established for straight-line programs [Sethi 1973; Lee et al. 2007] and at the same time heuristics and practical implementations have also been developed [Goodman and Hsu 1988]. These heuristics for straight-line programs have been used for *local* register allocation and have motivated algorithms such as the *linear scan* algorithm [Poletto and Sarkar 1999] that performs global register allocation. It has been pointed out that modern compiler optimization techniques such as loop unrolling and trace scheduling can yield large “basic blocks” of straight-line code [Guo et al. 2004] and this is another justification for focusing on straight-line programs.

It is well-known that the chromatic number $\chi(G)$ of any interval graph G equals its maximum clique size $\omega(G)$. Interval graphs can be colored optimally by considering intervals in increasing order of their left end-points and using the smallest available color for each interval. It is easy to see that an optimal solution to the max-coloring problem may use more colors than the chromatic number. For example, consider a path with 4 vertices such that the vertices at the two ends of the path are assigned weight W and the two vertices in the middle are assigned weight w , where $w < W/2$. This path has a unique 2-coloring with weight $2W$, but the optimal solution to max-coloring uses 3 colors and has weight $W + 2w$.

The max-coloring problem has been studied by Gaun and Zhu [Guan and Zhu 1997], who motivate the problem by an application that involves the transmission of wireless messages in a metropolitan network. They study the maximum number of colors needed by an optimal max-coloring for various special classes of graphs, and also show that the max-coloring problem can be solved in polynomial time for graphs of bounded path-width. The max-coloring problem has also been studied by Govindarajan and Rengarajan [Govindarajan and Rengarajan 1996] who, like us, are motivated by the problem of allocating a small buffer, but in the context of digital signal processing applications. The authors experimentally evaluate a first-fit strategy for max-coloring on circular arc graphs; in their experiments the first-fit strategy produces a solution with weight within 2.1% of the optimal weight. While the focus of this paper is interval graphs, we point out later that using our algorithm for interval graphs, a 3-approximation for circular arc graphs is easily obtained.

The max-coloring problem is related to the well-studied *dynamic storage allocation problem*, also known as the *interval coloring problem*. Formally, an instance

¹Due to a pending patent application we are not able to mention names of specific companies that have used the above approach in the design of their protocol stacks.

of this problem consists of an interval graph $G = (V, E)$ and a weight function $w : V \rightarrow \mathbf{N}$. A feasible solution to this problem is an assignment of an interval $I(v)$ to each vertex v such that $|I(v)| = w(v)$ and $I(u) \cap I(v) = \emptyset$ if u and v are adjacent vertices. The goal is to minimize $|\cup_{v \in V} I(v)|$. Figure 1(a) shows an interval representation of an interval graph, each interval labeled with a “name” and a weight. Figure 1(b) shows a feasible assignment of intervals to each of the vertices A, B, \dots, H . Specifically, $I(A) = (0, 3)$, $I(B) = (3, 4)$, $I(C) = (0, 2)$, $I(D) = (2, 3)$, $I(E) = (1, 2)$, $I(F) = (0, 1)$, $I(G) = (3, 5)$, $I(H) = (1, 4)$. With this assignment, $\cup_{v \in V} I(v) = (0, 5)$. Stockmeyer proved this problem NP-complete in 1976 (see problem SR2 in Garey and Johnson [Garey and Johnson 1979]) and Kierstead presented the first constant-factor approximation algorithm in 1988 [Kierstead 1988]. This was an 80-approximation algorithm that used a first-fit strategy to perform on-line coloring of unweighted interval graphs. Kierstead [Kierstead 1991] subsequently improved this to a 6-approximation algorithm, which was then improved by Gergov [Gergov 1996; 1999] to a 5-approximation and then a 3-approximation algorithm. Recently, Buchsbaum et. al. [Buchsbaum et al. 2004] presented a $(2 + \varepsilon)$ -approximation for this problem.

The similarity between the interval coloring problem and the max-coloring problem can be best understood by casting these problems into a geometric setting as rectangle packing problems. Start with an interval representation $\{I_v \mid v \in V\}$ of the given interval graph $G = (V, E)$ ². Interpret each weight $w(v)$ as the height of interval I_v . In other words, the instance of the problem consists of axis-parallel rectangles $\{R_v \mid v \in V\}$, such that the projection of R_v on the x -axis is I_v and the height of R_v is $w(v)$. Each rectangle can be slid up or down but not sideways; all rectangles have to occupy the positive quadrant; and the regions of the plane they occupy have to be pairwise disjoint. Given these constraints, the interval coloring problem is equivalent to the problem of packing these rectangles so as to minimize the y -coordinate of the highest point contained in any rectangle. The max-coloring problem seeks a packing of the rectangles into disjoint horizontal strips $S_i = \{(x, y) \mid x \geq 0, \ell_i \leq y \leq u_i\}$, denoted by (ℓ_i, u_i) . The constraints are that every rectangle is completely contained in some strip and for any two rectangles R_u and R_v in a strip, their projections on the x -axis I_u and I_v are disjoint. Given these constraints, the max-coloring problem seeks a packing of the rectangles into strips so that the total height $\sum(u_i - \ell_i)$ of the strips is minimized. Figure 1 shows two rectangle packings of a set of rectangles; the packing on the left is optimal for the interval coloring problem and the packing on the right is optimal for the max-coloring problem.

Stated as rectangle packing problems, interval coloring and max-coloring seem similar. However, as we show below, the weights of optimal solutions for the two problems on the same input can be quite different. Let OPT_I denote the weight of an optimal interval coloring and let OPT_M denote the weight of an optimal max-coloring for a given instance. Since any feasible solution of the max-coloring problem

²Without loss of generality, we assume that the input to our algorithms is a set of weighted intervals. This is because there are many linear-time algorithms for recognizing interval graphs and most of these return an interval representation of the given graph, if it is an interval graph. See [Cornel et al. 1998] for a recent algorithm.

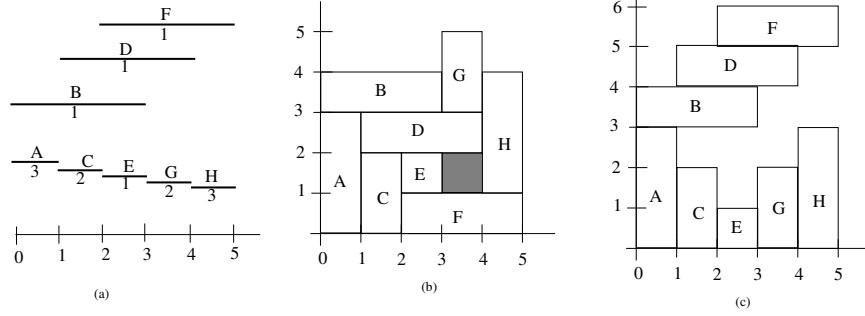


Fig. 1. (a) is an interval representation of an interval graph; the “name” and weight of each interval are shown. (b) is a rectangle packing of this interval graph corresponding to an optimal interval coloring, with weight 5. This figure is from [Buchsbbaum et al. 2004] (c) is a rectangle packing corresponding to an optimal max-coloring. In the packing on the right the rectangles are packed into 4 strips: $S_1 = (0, 3)$, $S_2 = (3, 4)$, $S_3 = (4, 5)$, and $S_4 = (5, 6)$, for a total weight of 6.

is also a feasible solution to the interval coloring problem, it follows that $OPT_I \leq OPT_M$. For any clique Q in the given graph, every vertex in the clique needs to have a distinct color and therefore $\sum_{v \in Q} w(v)$ is a lower bound on both OPT_I and OPT_M . Let $LOAD$ denote the maximum over all cliques Q in G of $\sum_{v \in Q} w(v)$. Equivalently, in the context of rectangle packings, $LOAD$ is the maximum sum of heights of rectangles that intersect any vertical line. Clearly, $LOAD \leq OPT_I \leq OPT_M$ and Gergov [Gergov 1999] shows that $OPT_I \leq 3 \cdot LOAD$. Buchsbbaum et.al. [Buchsbbaum et al. 2004] further investigate the relationship between $LOAD$ and OPT_I and show that $OPT_I = LOAD + O((w_{\max}/LOAD)^{1/7}) \cdot LOAD$, where w_{\max} is the maximum weight of any vertex in the graph.

It is easy to construct an instance for which $LOAD$ and OPT_I are poor lower bounds on OPT_M . Consider the weighted intervals shown in Figure 2. These form n disjoint cliques, Q_1, Q_2, \dots, Q_n , where clique Q_i contains i intervals each with weight $\lceil W/i \rceil$, where $W \geq n$ is an integer. Letting $w(Q_i)$ denote $\sum_{v \in Q_i} w(v)$ we see that $w(Q_i) = i \cdot \lceil W/i \rceil \leq W + (i-1)$. From this it follows that $LOAD \leq W + (n-1)$. Also note that for this instance $LOAD = OPT_I$. It can be verified that the optimal solution for max-coloring is an n -coloring C_1, C_2, \dots, C_n , where C_i contains exactly one interval each from Q_n, Q_{n-1}, \dots, Q_i . Letting $w(C_i)$ denote $\max_{v \in C_i} w(v)$ we see that $w(C_i) = \lceil W/i \rceil$. This implies that $OPT_M = \sum_{i=1}^n \lceil W/i \rceil \geq W \cdot H_n$, where H_n is the n th harmonic number. The upper bound on $LOAD$ along with the above lower bound on OPT_M together imply that for this family of instances $OPT_M = \Omega(LOAD \cdot \log n)$. Despite the fact that the obvious lower bound can be rather loose, we are able to develop several $O(1)$ -approximation algorithms for the max-coloring problem.

1.0.0.2 *Online Coloring of Intervals with Bandwidth.* In an instance of the problem of *online coloring of intervals with bandwidth*, intervals are presented one at a time, with each interval i being associated with a bandwidth $b(i) \in (0, 1]$. Each interval must be assigned a color immediately after it has been presented (and before the next interval is presented) and a color assigned to an interval cannot be

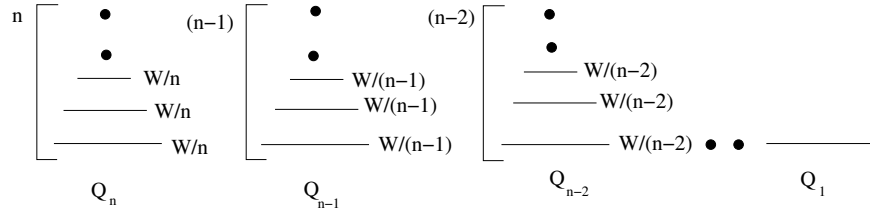


Fig. 2. An example of a weighted interval graph for which $OPT_M = \Omega(LOAD \cdot \log n)$.

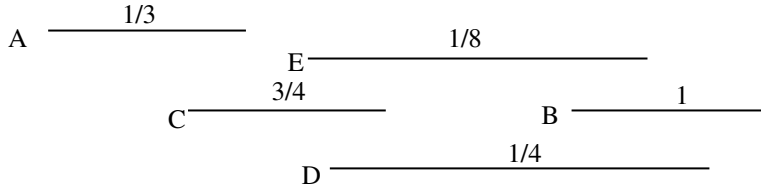


Fig. 3. An instance of the problem of online interval coloring with bandwidths. The numbers shown above the intervals correspond to their bandwidths.

changed later. An algorithm for this problem assigns colors to intervals in the manner described above, so that for any color c and any real r , the sum of bandwidths of intervals containing r and colored c is at most 1. The goal is to minimize the number of colors used. This problem was introduced by Adamy and Erlebach in [Adamy and Erlebach 2004]. Their study is motivated by problems in resource allocation such as channel allocation in an all optical WDM (wavelength-division-multiplexing) network, minimizing total duration in a call scheduling problem, and minimizing the number of machines used in a job scheduling problem in which each bandwidth represents the fraction of a machine that can be used by a job.

An instance of this problem is shown in Figure 3. For any real r , define $\text{density}(r) = \sum_{i:r \in i} b(i)$. Clearly, $\lceil \max_r \text{density}(r) \rceil$ is a lower bound on the number of colors needed. In the instance shown in Figure 3, $\max_r \text{density}(r) = 1 \frac{3}{8}$ and therefore at least 2 colors are needed for this instance. It is easy to check that $\{A, E, D\}, \{C, B\}$ is a feasible coloring for this instance and is therefore optimal.

Note that if all the bandwidths are 1, then we have the well-studied problem of the online coloring of intervals. In 1981, Kierstead and Trotter [Kierstead and Trotter 1981] presented a simple algorithm for online coloring of intervals that uses at most $3\chi - 2$ colors, where χ is the minimum number of colors needed (the chromatic number of the corresponding interval graph). The first-fit algorithm, a natural strategy for on-line coloring of intervals, has also been analyzed extensively [Irani 1994; Kierstead 1988; Kierstead and Qin 1995]. The best known bounds are by Kierstead and Qin [Kierstead and Qin 1995], who show that the algorithm uses no more than 26χ colors. The best lower bound on the number of colors used by first-fit is 4.4χ [Chrobak and Ślusarek 1988].

As mentioned by Adamy and Erlebach [Adamy and Erlebach 2004], the problem of online coloring of intervals with bandwidth is a simultaneous generalization of

online coloring of intervals and online bin packing. The First-Fit algorithm for online coloring of intervals without bandwidths naturally extends to the current problem, with bandwidth constraints. Given a coloring of a subset $S \subseteq \mathcal{I}$ of the intervals, we define

$$\text{density}_c(r) = \sum_{x \in S: r \in x, \text{color}(x)=c} b(x)$$

for each color c and real r . Then the First-Fit algorithm for the online interval coloring problem with bandwidths uses the following first-fit rule: for each interval x that arrives, color x with the smallest color c such that $b(x) + \text{density}_c(r) \leq 1$ for all $r \in x$. If intervals in the instance shown in Figure 3 arrive in the order A, B, C, D, E , then the First-Fit algorithm uses 3 colors because it colors A and B with color 1, C and D with color 2, and E with color 3.

The performance of First-Fit algorithm for the problem of online coloring of intervals with bandwidths differs in a fundamental way from the First-Fit algorithm for the usual problem of online coloring of intervals. We have noted the existence of an upper bound $26 \cdot \chi(\mathcal{I})$ on the number of colors used by First-Fit for the usual problem. However, as pointed out in [Adamy and Erlebach 2004], for the problem with bandwidths the First-Fit algorithm can be forced to do arbitrarily poorly. We present a bad example in Figure 4. Pick an integer $k > 1$ and a real $\varepsilon \in (0, 1)$. Corresponding to each pair (k, ε) of values we construct an instance that contains intervals with bandwidth ε and intervals with bandwidth 1. Figure 4 shows an instance with $k = 5$. The long (and thin) intervals all have bandwidth $\varepsilon > 0$ and the short (and thick) intervals all have bandwidth 1. A k -coloring of this instance produced by the First-Fit algorithm is shown, where the intervals in color 1 are provided first to the algorithm, followed by all the intervals in color 2, and so on. Notice that for each interval of bandwidth ε , there is an intersecting interval with bandwidth 1 in each of the previous color classes. Similarly, for each interval of bandwidth 1, there is an intersecting interval of bandwidth ε in each of the previous color classes. This justifies the first-fit coloring shown in the figure. Also notice that lower bound on the optimal number of colors needed is $\max_r \text{density}(r) = 1 + (k - 1) \cdot \varepsilon$. Choosing $\varepsilon = 1/(k - 1)$, we get a lower bound of 2. A 2-coloring of this instance can be obtained by using color 1 for all intervals with bandwidth 1 and for the interval with bandwidth ε with the rightmost right endpoint. It is easy to see that the remaining intervals, all of which have bandwidths ε , can be colored with color 2.

1.0.0.3 *Organization..* The first results of this paper are approximation algorithms for the max-coloring problem on interval graphs, obtained via a connection to online coloring. These are described in Section 2, where we present a 3-approximation and then a 2-approximation algorithm. We then show, in Section 3, that the max-coloring is NP-hard on interval graphs. In Section 4 we present an analysis of the first-fit algorithm for the on-line coloring problem on interval graphs and show an upper bound of $8 \cdot \chi(G)$ on the number of colors used for any interval graph G . This analysis shows that the first-fit strategy produces an 8-approximation for max-coloring, but more importantly it provides the best known upper bounds on the number of colors used by first-fit improving significantly on

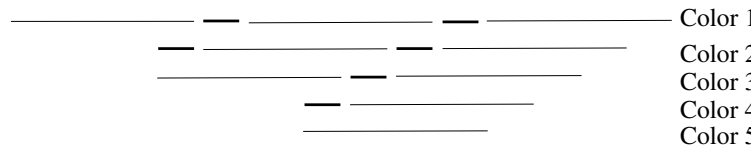


Fig. 4. An instance of intervals with bandwidth showing that First-Fit can perform arbitrarily badly.

```

MCA( $G, w$ )
1. Sort the vertices of  $G$  in non-increasing order of weights.
   (Let  $(v_1, v_2, \dots, v_n)$  be this ordering of the vertices of  $G$ .)
2. Present the vertices in the order  $v_1, v_2, \dots, v_n$  to  $A$ .
3. Return the coloring produced by  $A$ .

```

Fig. 5. The MCA algorithm

the $26 \cdot \chi(G)$ upper bound of Kierstead and Qin [Kierstead and Qin 1995].

We next consider the problem of online coloring of interval graphs with bandwidths. In Section 5.1 we show that the Adamy-Erlebach algorithm is 35-competitive. This requires an extension of the analysis of First-fit in Section 4 to the case of intervals with bandwidths. The extended analysis is presented in Section 5.2. We conclude with some open questions in Section 6.

2. APPROXIMATION ALGORITHMS FOR MAX-COLORING

In an instance of the *on-line graph coloring problem*, vertices of a graph are presented one at a time and when a vertex is presented, all edges connecting that vertex to previously presented vertices are also revealed. Each vertex must be assigned a color immediately after it has been presented (and before the next vertex is presented) and a color assigned to a vertex cannot be changed later. An algorithm for the on-line graph coloring problem assigns colors to vertices in the manner described above, so as to construct a proper vertex coloring of the graph. We say that an algorithm A for the on-line graph coloring problem k -colors a graph G , if no matter which order the vertices of G are presented in, A uses at most k colors to color G .

Let A be an algorithm for the on-line graph coloring problem. We use A as a “black-box” to devise a simple algorithm for the max-coloring problem. The algorithm, called MCA (short for max-coloring algorithm) is given in Figure 5.

We will now make a connection between the number of colors used by A and the weight of the coloring produced by MCA. This connection, along with known results on on-line coloring of interval graphs will lead to constant factor approximation algorithms for max-coloring for interval graphs.

THEOREM 2.1. *Let \mathcal{C} be a hereditary class³ of graphs and let A be an algorithm for on-line graph coloring such that for some integer constant $c > 0$ and for any graph $G \in \mathcal{C}$, A colors G with at most $c \cdot \chi(G)$ colors. Then, for any $G \in \mathcal{C}$ and for any weight function $w : V(G) \rightarrow \mathbf{N}$, MCA produces a coloring for G whose weight is at most $c \cdot OPT_M(G)$.*

PROOF. Let C_1, C_2, \dots, C_k be a coloring of G that is optimal for the max-coloring problem. Let $w_i = \max_{v \in C_i} w(v)$ and without loss of generality assume that $w_1 \geq w_2 \geq \dots \geq w_k$. Now note that $k \geq \chi(G)$ and $OPT_M(G) = \sum_{i=1}^k w_i$. Let A_1, A_2, \dots, A_t be the coloring of G produced by MCA. Let $a_i = \max_{v \in A_i} w(v)$ and without loss of generality assume that $a_1 \geq a_2 \geq \dots \geq a_t$. From our hypothesis it follows that $t \leq c \cdot \chi(G) \leq c \cdot k$. For notational convenience, define sets $A_{t+1} = A_{t+2} = \dots = A_{c \cdot \chi(G)} = \emptyset$ and let $a_i = 0$ for $i, t < i \leq c \cdot \chi(G)$. We will now claim that for each $i, 1 \leq i \leq k$, and each $j, c(i-1) < j \leq c \cdot i$, we have $w_i \geq a_j$. Showing this would imply the result we seek because the coloring produced by MCA has weight

$$\sum_{\ell=1}^{c \cdot \chi(G)} a_\ell = \sum_{i=1}^{\chi(G)} \sum_{j=c(i-1)+1}^{c \cdot i} a_j \leq \sum_{i=1}^{\chi(G)} c \cdot w_i \leq c \cdot OPT_M(G).$$

Since w_1 is the maximum weight of any vertex in G , the claim is trivially true for $i = 1$. For any $i \geq 2$, let $V_i \subseteq V$ be defined as $V_i = \{v \mid w(v) > w_i\}$. The coloring C_1, C_2, \dots, C_k of G , restricted to V_i is an $(i-1)$ -coloring of $G[V_i]$, the subgraph of G induced by V_i . Because of the order in which vertices are presented to A , all vertices in V_i are presented to A before any vertex with weight w_i . Therefore, by our hypothesis, algorithm A colors $G[V_i]$ with no more than $c \cdot (i-1)$ colors. Therefore, the weight of the heaviest vertex in color classes A_j for $j, c(i-1) < j \leq c \cdot i - 1$ is at most w_i . \square

Recall that Kierstead and Trotter [Kierstead and Trotter 1981] presented a simple algorithm for on-line coloring of interval graphs such that for any interval graph G with $\chi(G) = k$, the algorithm $(3k-2)$ -colors the graph. From this result, and Theorem 2.1, a 3-approximation algorithm for max-coloring interval graphs follows, after noting that interval graphs are a hereditary class of graphs.

THEOREM 2.2. *There is a 3-approximation algorithm for solving the max-coloring problem on interval graphs.*

Rather than use the Kierstead-Trotter algorithm as a black box, if we make a simple modification to one of the steps in the Kierstead-Trotter algorithm, we can reduce the approximation factor from 3 to 2. The Kierstead-Trotter algorithm maintains sets S_1, S_2, \dots such that when a vertex u is presented, it finds the smallest i such that $S_1 \cup S_2 \cup \dots \cup (S_i \cup \{u\})$ does not contain an $(i+1)$ -clique. The vertex u is then inserted into S_i . Kierstead and Trotter show that each induced subgraph $G[S_i]$ is the union of disjoint paths. Therefore $G[S_i]$ can be 3-colored using the “first-fit” on-line algorithm that assigns to each presented vertex the smallest available color. The fact that $G[S_i]$ is the union of disjoint paths follows from the fact that no

³A class \mathcal{C} of graphs is hereditary if $G \in \mathcal{C}$ implies that every induced subgraph of G is also in \mathcal{C} .



Fig. 6. This shows the two “configurations” that are forbidden from appearing in any of the Kierstead-Trotter sets S_i . The absence of such configurations implies that each set S_i is a disjoint union of paths.

```

BETTER-MCA( $G, w$ )
0. Sort the vertices of  $G$  in non-increasing order of weights.
   (Let  $(v_1, v_2, \dots, v_n)$  be this ordering of the vertices of  $G$ .)
1. Let  $k \leftarrow 1$ ;  $S_k \leftarrow \emptyset$ 
2. for  $j \leftarrow 1$  to  $n$  do
3.   Insert  $v_j$  into set  $S_i$ , where  $i \leq k$  is the smallest value such that
      $S_1 \cup S_2 \cup \dots \cup (S_i \cup \{v_j\})$  does not contain an  $(i+1)$ -clique.
     If no such  $i$  exists, set  $k \leftarrow k+1$  and insert  $v_j$  into  $S_k$ 
4. Use color 1 to color vertices in  $S_1$ 
5. for  $i \leftarrow 2$  to  $k$  do
6.   Use colors  $2i-2$  and  $2i-1$  to 2-color the vertices in  $S_i$ 
7. Return the coloring

```

Fig. 7. The BETTER-MCA algorithm

two intervals in S_i can have the configuration shown in Figure 6(a) and no three intervals in S_i can have the configuration shown in Figure 6(b). We now argue that no two intervals in S_i can have the configuration shown in Figure 6(a); the argument relating to the second configuration is similar. Let C denote the set of intervals in $S_1 \cup S_2 \cup \dots \cup S_{i-1}$ that have arrived before v . Observe that interval v (see Figure 6(a)) is inserted into S_i by the algorithm because $\{v\} \cup C$ contains an i -clique. Since all neighbors of v are also neighbors of u , it follows that $\{u, v\} \cup C$ contains an $(i+1)$ -clique. Hence, both u and v could not have been inserted into set S_i by the algorithm. The new algorithm for max-coloring is given in Figure 7.

Steps (2) and (3) come from the Kierstead-Trotter algorithm. The modification we make to the Kierstead-Trotter algorithm is that instead of coloring S_i on-line with 3 colors, we just color S_i off-line using two colors. This is possible because, as mentioned above, $G[S_i]$ is a union of disjoint paths [Kierstead and Trotter 1981].

THEOREM 2.3. BETTER-MCA is a 2-approximation algorithm for the max-coloring problem on interval graphs.

PROOF. Let C_1, C_2, \dots, C_k be a coloring of G that is optimal for the max-coloring problem. Let $w_i = \max_{v \in C_i} w(v)$ and without loss of generality assume that $w_1 \geq w_2 \geq \dots \geq w_k$. Now note that $k \geq \chi(G)$ and $OPT_M(G) = \sum_{i=1}^k w_i$.

Suppose that at the end of Step (3) in BETTER-MCA, we have sets S_1, S_2, \dots, S_t . An element is inserted into S_t only because $S_1 \cup S_2 \cup \dots \cup (S_{t-1} \cup \{u\})$ has a t -clique. Therefore, $\chi(G) \geq t$ and it follows that $t \leq k$. Let $s_i = \max_{v \in S_i} w(v)$. We now claim that for each i , $1 \leq i \leq t$, $s_i \leq w_i$.

It is clear that $s_1 = w_1$. To obtain a contradiction, suppose that for some i ,

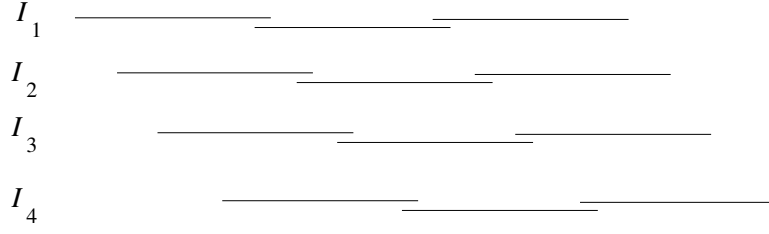


Fig. 8. This shows $\mathcal{I}(k)$ for $k = 4$. When all intervals in $\mathcal{I}(k)$ are assigned unit weights, an optimal max-coloring has weight $k + 1$, whereas BETTER-MCA produces a max-coloring of weight $2k$.

$s_i > w_i$ and let i be the smallest such value. This implies any vertex x with weight s_i or larger is in an earlier color class, C_j , for some $j < i$ in the optimal coloring. Therefore, vertices with weight s_i or larger are $(i - 1)$ -colorable, so they induce a clique of size at most $i - 1$ and therefore in Step (3) no vertex with weight s_i will get inserted into S_i - a contradiction.

In Steps (4) and (5) we convert the vertex partition S_1, S_2, \dots, S_t into a coloring whose weight is at most $s_1 + 2 \cdot \sum_{i=2}^t s_i$. The following inequalities

$$s_1 + 2 \cdot \sum_{i=2}^t s_i \leq w_1 + 2 \cdot \sum_{i=2}^t w_i \leq 2 \cdot OPT_M(G)$$

give the result we seek. \square

It is easy to see that the above analysis is tight. Let $\mathcal{I}(k)$ denote a set of intervals constructed by taking the union of k sets of intervals, $\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_k$, such that the interval graph of each \mathcal{I}_j is a 3-path. Thus, $|\mathcal{I}(k)|$ has $3k$ intervals. As illustrated in Figure 8, each \mathcal{I}_j , $1 < j \leq k$, is obtained by taking a copy of \mathcal{I}_{j-1} and shifting it to the right. This is done in a manner so that the size of a maximum clique in $\mathcal{I}(k)$ is $k + 1$. Suppose that the weights of all intervals in $\mathcal{I}(k)$ are unit. Then an optimal max-coloring of $\mathcal{I}(k)$ has weight $k + 1$. Since all weights are identical, any ordering of the intervals is a valid output of Step 0 in BETTER-MCA. Assume that the intervals are ordered so that intervals in \mathcal{I}_1 come first, followed by intervals in \mathcal{I}_2 , followed by intervals in \mathcal{I}_3 , and so on. Furthermore, suppose that the intervals in each \mathcal{I}_j are presented in left-to-right order of their left endpoints. It is easy to check that Steps 2-3 of BETTER-MCA partition the intervals in $\mathcal{I}(k)$ into sets S_1, S_2, \dots, S_{k+1} such that S_1 and S_2 are independent sets, but the remaining sets S_j , $2 < j \leq k + 1$ are not. To be more specific, suppose that the intervals in each set \mathcal{I}_j are labeled I_j^1 , I_j^2 , and I_j^3 respectively in left-to-right order of their left endpoints. Then $S_1 = \{I_1^1, I_1^3\}$ and $S_2 = \{I_1^2, I_2^3\}$, both independent sets. For $2 < j \leq k$, $S_j = \{I_{j-1}^1, I_{j-1}^2, I_j^3\}$ and $S_{k+1} = \{I_k^1, I_k^2\}$. Note that each of the sets S_j , $2 < j \leq k + 1$, contains a pair of adjacent vertices. Thus BETTER-MCA yields a max-coloring of weight $2k$.

3. NP-COMPLETENESS RESULT

In this section, we show that the problem of max-coloring interval graphs is NP-hard. Our proof is considerably simpler than the one presented in the preliminary

version of this paper [Pemmaraju et al. 2004] and is heavily influenced by an NP-completeness proof for minimum sum coloring on interval graphs by D. Marx [Marx 2005]. The reduction is from coloring *circular-arc graphs*. A graph $G = (V, E)$ is a circular-arc graph if its vertices can be placed in one-to-one correspondence with open arcs on the circumference of a circle such that two arcs intersect if and only if the corresponding vertices are adjacent. The decision version of the problem, whether there exists a k -coloring for a given circular-arc graph $G = (V, E)$ was proved NP-complete by Garey, et. al. in [Garey et al. 1980]. We now describe the problem and the reduction.

CIRCULAR-ARC GRAPH COLORING

INPUT: A circular-arc graph $G = (V, E)$ and an integer k .

QUESTION: Is G k -colorable ?

We may assume that a circular arc representation of G is given to us. This assumption is justified by the availability of a polynomial time algorithms (linear time, in fact) for recognizing a given graph G as a circular arc graph and returning a circular arc representation if G is indeed a circular arc graph [McConnell 2003]. Without loss of generality, we can assume that there exists a point on the circle that is contained in precisely k circular arcs. If not, consider a sector that does not contain any end-points of any of the arcs. Assume that this sector is contained in $l < k$ arcs. We can introduce $k - l$ arcs that are contained only in this sector. The modified graph is k -colorable if and only if the original graph is k -colorable. This is because any coloring of the original graph that uses at least k colors can be extended to the modified graph, as the newly added arcs have at least $k - l$ available colors. Note that if the sector was contained in at least $k + 1$ arcs, then G is not k -colorable, and the reduction can be trivially completed.

THEOREM 3.1. *Max-coloring on interval graphs is NP-complete.*

PROOF. Given a circular-arc graph, $G = (V, E)$, consider a ray r from the center of the circle that intersects precisely k circular arcs. For each such arc v_i , $i = 1, \dots, k$, partition v_i into two arcs l_i and r_i , where l_i has the same left end-point as v_i (in a clockwise direction), and has as right end-point, the point of intersection of r and v_i . r_i has left end-point the intersection point of r and v_i , and its right end-point is that of v_i . This gives an interval graph $G' = (V', E')$. The weights of the vertices V' are as follows. For each l_i, r_i , let $w(l_i) = w(r_i) = i$. For all other vertices v , let $w(v) = 1$. This reduction is illustrated in Figure 9.

If the circular-arc graph G has a k -coloring then, assigning $color_{G'}(v) := color_G(v)$, for all $v \notin \{l_i, r_i, \mid i = 1, \dots, k\}$, and $color_{G'}(l_i) := color_{G'}(r_i) := color_G(v_i)$, $i = 1, \dots, k$ gives a max-coloring of G' of cost $k(k + 1)/2$.

Note that the intervals l_i and the intervals r_i each form a clique of weight $k(k + 1)/2$. This is a lower bound on the max-coloring cost of G' . If G' has a max-coloring of cost $k(k + 1)/2$, then it must be the case that (1) $color_{G'}(l_i) = color_{G'}(r_i)$, for $i = 1, \dots, k$, and (2) the number of colors used in the max-coloring is k . Thus by setting $color_G(v_i) := color_{G'}(l_i)$, and $color_G(v) := color_{G'}(v)$ for all other vertices v , we get a k -coloring of G .

Thus if we could solve the decision version of the max-coloring problem in polynomial time, that would allow us to determine if G' has a max-coloring of cost at most $k(k + 1)/2$ and that in turn would allow us to decide CIRCULAR-ARC GRAPH

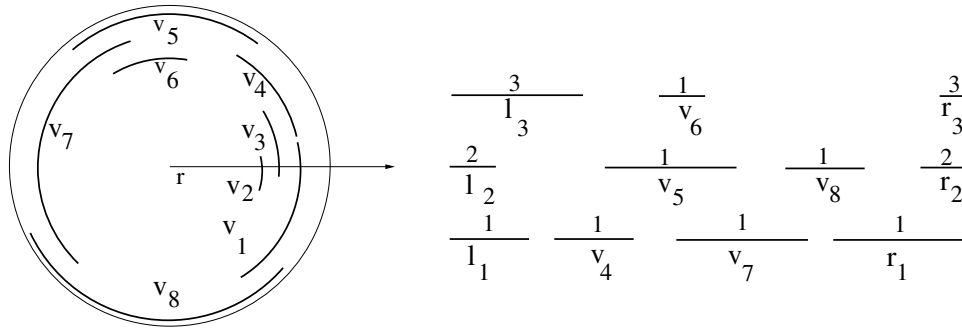


Fig. 9. This is an illustration of the NP-completeness reduction that shows that max-coloring on interval graphs is NP-complete. The figure on the left shows a circular arc representation of a circular arc graph. Here $k = 3$. The ray r “splits” each arc v_i , $1 \leq i \leq 3$, into two intervals ℓ_i and r_i , yielding the set of intervals shown in the figure on the right. Note that the intervals ℓ_i and r_i are assigned weight i , and the remaining intervals are assigned weight 1.

COLORING in polynomial time. \square

4. ANALYSIS OF FIRST-FIT

The first-fit algorithm for on-line coloring assigns the smallest available color to each presented vertex. In MCA, if we use first-fit instead of the Kierstead-Trotter algorithm for on-line coloring what performance guarantee can we provide? From Theorem 2.1 and from the result of Kierstead and Qin [Kierstead and Qin 1995] who show that the first-fit algorithm uses no more than $26 \cdot \chi(G)$ colors on an interval graph G , we get a 26-approximation algorithm for max-coloring on interval graphs. However, in practice first-fit shows excellent performance [Govindarajan and Rengarajan 1996] and along with best-fit is a popular choice as a memory allocation strategy. Furthermore, the lower bound on the number of colors used by first-fit to do on-line coloring of an interval graph is $4.4 \cdot \chi(G)$ [Chrobak and Ślusarek 1988]. This along with the belief expressed in [Kierstead and Qin 1995] that the upper bound of $26 \cdot \chi(G)$ could be improved, is the motivation for the new analysis of the first-fit strategy. Our analysis, inspired by some ideas in Gergov [Gergov 1999], is quite different from the analysis in [Kierstead 1988; Kierstead and Qin 1995] and in our view, much simpler. In a preliminary version of this paper [Pemmaraju et al. 2004], we had shown that first-fit uses at most $10 \cdot \chi(G)$ colors. In [Brightwell et al. 2003], Brightwell, et. al. modified the analysis slightly to achieve the bound of $8 \cdot \chi(G)$. Also, Narayanaswamy and Subhash Babu [Narayanaswamy and Babu 2004] presented a different modification which also leads to an $8 \cdot \chi(G)$ upper bound. We present our analysis from [Pemmaraju et al. 2004] as improved by [Narayanaswamy and Babu 2004], which leads to a 8-approximation algorithm for max-coloring on interval graphs.

THEOREM 4.1. *For any interval graph G , the first-fit strategy for on-line coloring of G uses at most $8 \cdot \chi(G)$ colors.*

It is convenient to work with a representation of G in terms of a set of intervals \mathcal{S} . We assume without loss of generality that each interval in the set \mathcal{S} of input

intervals is of the form $[i, j]$, where $0 \leq i < j \leq N$ are integers, and N is a sufficiently large positive integer. We denote by \mathcal{E} the set of *elementary* intervals $\{[i-1, i] \mid 1 \leq i \leq N\}$. We will refer to the ordering of the intervals in \mathcal{E} according to increasing order of the left endpoints as their *natural* ordering. Thus each input interval is a union of consecutive elementary intervals; two input intervals intersect if they both contain a common elementary interval. The *leftmost* (resp. *rightmost*) elementary interval of an interval $I = [i, j] \in \mathcal{S}$ is $[i, i+1]$ (resp. $[j-1, j]$).

We briefly review the first-fit algorithm for coloring intervals in \mathcal{S} . The intervals are presented to the algorithm in some arbitrary order. The first interval is assigned to color class 1. Each subsequent interval is then assigned to the lowest color class that will accept it. That is, assume that the algorithm has seen intervals $\mathcal{S}' \subseteq \mathcal{S}$ and assigned them to color classes $1, \dots, j$. When a new interval I arrives, the first-fit algorithm finds the smallest value of i , for $1 \leq i \leq j+1$, such that I does not intersect any interval in \mathcal{S}' that was assigned to the i 'th color class. Such an i must exist because no interval in \mathcal{S}' is assigned to the $(j+1)$ 'th color class. The algorithm then assigns I to the i 'th color class.

Clearly, the first-fit algorithm yields a proper coloring of the intervals in \mathcal{S} (more precisely, of the interval graph corresponding to \mathcal{S}). Suppose that it uses colors $1, \dots, m$. We will now argue that there is an elementary interval that is contained in at least $m/8$ input intervals. Note that this corresponds to a clique of size at least $m/8$ in the corresponding interval graph, which means $m/8$ is a lower bound on the number of colors used in any proper coloring of the interval graph. A useful way to visualize the coloring generated by first-fit is to imagine an interval $[l, r]$ that is assigned to color class k as the rectangle $\{(x, y) \mid l \leq x \leq r, k-1 \leq y \leq k\}$ of height one.

Column Construction

The key property of the first-fit coloring that will be needed in the proof is that if an interval $I \in \mathcal{S}$ is assigned to color class k , then for each $1 \leq i \leq k-1$ there is an interval I' assigned to color class i such that I intersects I' . The proof of Theorem 4.1 is based on a construction of a set of “columns” corresponding to the first-fit coloring. Referring to Figure 10 will clarify some of the following discussion. A column corresponds to a unique elementary interval e , and with some abuse of notation is referred to as column e . There may be elementary intervals that have no corresponding columns (e.g., columns $[2, 3]$, $[3, 4]$, and $[4, 5]$ in Figure 10). A column has a positive integral *height* associated with it. If a column has height t , we say that it is active at heights $1, \dots, t$ and inactive at heights $t+1, \dots, \infty$ (e.g., column $[1, 2]$ is active at heights 1, 2, and 3 and inactive at heights 4, 5, \dots). A column of height t is labeled, at each height i between 1 and t , with one symbol which is either “R”, “\$”, or “F”. A column e of height t is labeled “R” at some height $1 \leq i \leq t$ if and only if some interval $I \in \mathcal{S}$ that is assigned to the i 'th color class contains the elementary interval e . However, it could be the case that an elementary interval e is contained in an interval that is assigned to the i 'th color class and there is either no column corresponding to e or the column e is inactive at height i . For example, in Figure 10, elementary interval $[2, 3]$ is contained in an interval assigned to color 3, however there is no column corresponding to $[2, 3]$. It is useful to visualize a column e of height t as a rectangle $\{(x, y) \mid l(e) \leq x \leq r(e), 1 \leq y \leq t\}$ of width 1,

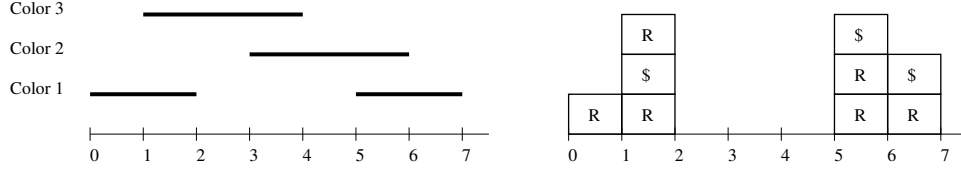


Fig. 10. The figure on the left is a first-fit coloring of a set of 4 intervals. The figure on the right is the result of applying the column construction procedure to this coloring. The columns correspond to elementary intervals $[0, 1]$, $[1, 2]$, $[5, 6]$, and $[6, 7]$. Column $[1, 2]$ has height 3 and has labels “R”, “\$”, and “R” associated with heights 1, 2, and 3 respectively.

where $l(e)$ and $r(e)$ are the left and right endpoints of elementary interval e ; the box $\{(x, y) | l(e) \leq x \leq r(e), i-1 \leq y \leq i\}$ contains the label of the column at height i , for $1 \leq i \leq t$. See Figure 10 for an example. It is worth pointing out that the goal of the column construction procedure is to find a column with at least $m/8$ “R” labels in it.

The column construction procedure works by starting with a set of columns that are active at height 1 and, for $i \geq 2$, choosing a subset of the active columns at height $i-1$ to be the active columns at height i . If the set of active columns at height i is empty, the procedure stops. For any set of columns, there is a natural ordering that is induced by the natural ordering of the corresponding elementary intervals. Let C_i denote the set of columns active at height i . If Figure 10, $C_1 = \{[0, 1], [1, 2], [5, 6], [6, 7]\}$, $C_2 = \{[1, 2], [5, 6], [6, 7]\}$, and $C_3 = \{[1, 2], [3, 4]\}$. For any $e \in C_i$, the left (resp. right) neighbor of e in C_i is the column in C_i immediately preceding (resp. succeeding) e in the natural ordering; if no such column exists, the left (resp. right) neighbor is undefined.

We are now ready to describe the column construction procedure. If an elementary interval e is contained in an interval assigned to the first color class, then e is added to C_1 and is assigned the label “R” at height 1. These are the only columns in C_1 . For $i \geq 2$, the following rules specify which columns from C_{i-1} are picked for C_i .

- (1) For each $e \in C_{i-1}$, if e is contained in some interval assigned to color class i , then e is added to C_i and is assigned a label “R”.
- (2) For each remaining $e \in C_{i-1}$, if e is the left neighbor in C_{i-1} of some column e' added to C_i by Rule 1, then e is added to C_i with a label “\$” at height i . For accounting purposes, we say that e' contributes a \$ to e at height i . Note that e is the left-neighbor of e' in C_i . For each remaining $e \in C_{i-1}$, if e is the right neighbor in C_{i-1} of some column e' added to C_i by Rule 1, then e is added to C_i with a label “\$” at height i . For accounting purposes, we say that e' contributes a \$ to e at height i . Note that e is the right neighbor of e' in C_i .
- (3) For each remaining $e \in C_{i-1}$, let e' be the left-neighbor of e in C_{i-1} . If undefined, we proceed to inspect the right neighbor of e in C_{i-1} . Suppose that e' is the left neighbor of e in C_j, \dots, C_{i-1} , and is not the left neighbor of e in C_1, \dots, C_{j-1} ; note that such a j does exist. If the number of “R” labels of e at heights $j, \dots, i-1$ is greater than $(i-j)/4$, then e is added to C_i with a label “F” at height i . If not, let e'' be the right neighbor of e in C_{i-1} . If undefined,



Fig. 11. (i) A snapshot in the construction of C_8 after Rule 1 has been applied. Columns b and e will get added to C_8 with $\$$ labels at height 8 due to Rule 2. (ii) Snapshot after Rule 2 has been applied. Column p is the left neighbor of r in C_6, C_7 and t is the right neighbor of r in C_5, C_6, C_7 . r is added to C_8 (with a label F at height 8) iff the number of its R labels at heights 6, 7 is greater than $2/4$ or the number of its R labels at heights 5, 6, 7 is greater than $3/4$.

e is not added to C_i . Otherwise, suppose that e'' is the right neighbor of e in C_k, \dots, C_{i-1} , and is not the right neighbor of e in C_1, \dots, C_{k-1} . If the number of “R” labels of e at heights $k, \dots, i-1$ is greater than $(i-k)/4$, then e is added to C_i with a label “F” at height i . If not, e is not added to C_i . See Figure 11 for an example.

We now provide some intuition for these column construction rules and explain how they lead to the analysis below. Rule 1 by itself does not lead to tall enough columns. For example, without Rule 2, the column construction in Figure 10 would end at height 2 and there would be no record of the interval colored 3. Together, Rules 1 and 2 yield tall columns. Specifically, if first-fit coloring uses m colors, then the column construction procedure is guaranteed to yield a column of height m . This is proved in Lemma 4.2. Thus, the fact that there are columns of height 3 in the example in Figure 10 is not a coincidence. It is not sufficient to have tall columns, we need tall columns with a lot of “R” labels in them. Using Rule 3, we show in Lemma 4.3 that the number of “F” labels in a column cannot be too large relative to the number of “R” labels; in particular, the number of “F” labels in a column is at most 3 times the number of “R” labels in that column. Now we just have to account for the $\$$ labels. By Rule 2, each $\$$ in a column e is “supported” by an “R” in a neighboring column (see Figure 12). By showing that the number of “R” labels in neighboring columns are bounded (Theorem 4.4), we obtain a bound on the number of $\$$ labels in column e .

The construction proceeds until round m' , after which *no* column is active, i.e., $C_{m'+1} = \emptyset$. Note that if $m' > m$, then at heights $m+1, \dots, m'$, only Rule 3 applies and the only symbols added are “F” symbols. The construction procedure maintains the following important invariant. Abusing notation, we say that interval I intersects a column e if it contains elementary interval e .

LEMMA 4.2. *Let $1 \leq i < j \leq m$, for any interval $I \in \mathcal{S}$ that is assigned to color class j , there is a column $e \in C_i$ such that I intersects e .*

PROOF. By induction on i . The lemma holds for $i = 1$ because of the key property of the first-fit coloring. For the inductive step, assume $i \geq 2$ and the

lemma holds for $i - 1$. Let I be an interval that is assigned to color class $j > i$. By the key property, there is an $I' \in \mathcal{S}$ that is assigned to the i 'th color class such that I' intersects I . By the induction hypothesis, I' (resp. I) intersects a non-empty set C' (resp. C) of consecutive columns (natural ordering) in C_{i-1} . If $C \cap C' \neq \emptyset$, we are done since all columns in C' are added to C_i by Rule 1. If some column $e \in C_{i-1}$ lies between the columns in C' and the columns in C , then I and I' cannot intersect because the elementary interval e lies between I and I' . So it must be that some $f \in C$ is either the left neighbor or right neighbor in C_{i-1} of some column in C' . The column f is added to C_i by Rule 2 if it is not already added by Rule 1, completing the proof. \square

The invariant implies that C_1, \dots, C_m are non-empty, and thus $m' \geq m$. For a column e with height at least j and $1 \leq i \leq j$, let $\rho_e(i, j)$, $\delta_e(i, j)$, and $\phi_e(i, j)$ denote, respectively, the number of R, \$, and F labels of e between heights i and j (inclusive). Define $\rho_e(j) = \rho_e(1, j)$, $\delta_e(j) = \delta_e(1, j)$, and $\phi_e(j) = \phi_e(1, j)$. Also, define $\rho_e(0) = \delta_e(0) = \phi_e(0) = 0$.

LEMMA 4.3. *For any column $e \in C_i$, and any integer $1 \leq i \leq m'$, $\rho_e(i) \geq \frac{1}{4}(\rho_e(i) + \phi_e(i))$.*

PROOF. The proof is by induction on i . The base cases $i = 0, 1$ are easily verified. Suppose $i \geq 2$ and the lemma is true for all $0 \leq i' < i$. If e is not labeled with F at height i , then the induction step goes through easily. So let us assume that e is labeled F at height i . So e was added to C_i by Rule 3 and so there exists a $1 \leq j \leq i - 1$ such that $\rho_e(j, i - 1) > (i - j)/4$. Since $\rho_e(j, i - 1)$ is an integer, this implies that $\rho_e(j, i) \geq (i - j + 1)/4$. From this and the induction hypothesis, it follows that

$$\begin{aligned} \rho_e(i) &= \rho_e(j, i) + \rho_e(j - 1) \\ &\geq \frac{1}{4}(i - j + 1) + \frac{1}{4}(\rho_e(j - 1) + \phi_e(j - 1)) \\ &\geq \frac{1}{4}(\rho_e(j, i) + \phi_e(j, i)) + \frac{1}{4}(\rho_e(j - 1) + \phi_e(j - 1)) \\ &= \frac{1}{4}(\rho_e(i) + \phi_e(i)) \end{aligned}$$

\square

We are now ready to obtain our main result.

THEOREM 4.4. *Let m denote the number of colors used by first-fit to color the set of intervals \mathcal{S} . There is a clique of size at least $m/8$ in the corresponding interval graph.*

PROOF. We will show that there is a column $e \in C_{m'}$ such that $\rho_e(m') \geq m/8$. Let e be any column in $C_{m'}$, and assume that e has both a left and right neighbor in $C_{m'}$. The case when either neighbor is absent is in fact easier. Let f_1, \dots, f_a be the left neighbors of e , where f_i is the left neighbor of e in $C_{t_{i-1}+1}, \dots, C_{t_i}$, $i = 1, \dots, a$, where $t_0 = 0 < t_1 < \dots < t_a = m'$. Similarly, let g_1, \dots, g_b be the right neighbors of e , with g_i being a right neighbor of e at $C_{n_{i-1}+1}, \dots, C_{n_i}$ for

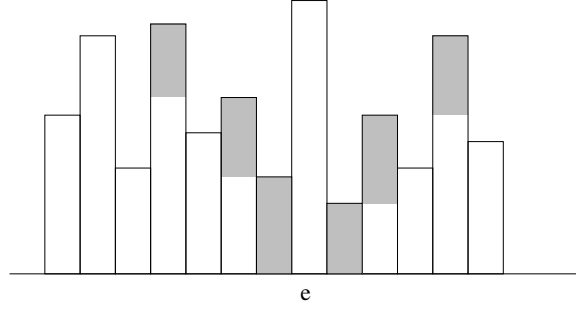


Fig. 12. The columns at the end of the construction procedure. The number of “\$” labels in column e is bounded by the number of “R” labels in all the shaded rectangles.

$i = 1, \dots, b$. $n_0 = 0 < n_1 < \dots < n_b = m'$. Now we claim that

$$\delta_e(m') \leq \sum_1^a \rho_{f_i}(t_{i-1} + 1, t_i) + \sum_1^b \rho_{g_i}(n_{i-1} + 1, n_i).$$

The claim follows from the observation that the label at height i in column e is a “\$” if and only if it is supported by either of its neighbors, i.e., a “\$” is added to e at level i by Rule 2. See figure 12. Since each f_i and g_i become inactive beyond a height of t_i and n_i respectively, by Rule 3 we have

$$\rho_{f_i}(t_{i-1} + 1, t_i) \leq \frac{1}{4}(t_i - t_{i-1}).$$

By an identical argument, we have that

$$\rho_{g_i}(n_{i-1} + 1, n_i) \leq \frac{1}{4}(n_i - n_{i-1}).$$

Hence,

$$\delta_e(m') \leq \sum_1^a \frac{1}{4}(t_i - t_{i-1}) + \sum_1^b \frac{1}{4}(n_i - n_{i-1}) \leq \frac{1}{4}(t_a + n_b) = \frac{m'}{2}$$

Since

$$\rho_e(m') + \phi_e(m') + \delta_e(m') = m',$$

and

$$\rho_e(m') + \phi_e(m') = m' - \delta_e(m') \geq m' - \frac{m'}{2} = \frac{m'}{2},$$

we can apply Lemma 4.3 to get

$$4\rho_e(m') \geq \frac{m'}{2}$$

or

$$\rho_e(m') \geq \frac{m'}{8}$$

Since no “R” labels are added after level m , $\rho_e(m) = \rho_e(m')$. Hence,

$$\rho_e(m) = \rho_e(m') \geq \frac{m'}{8} \geq \frac{m}{8}$$

By a similar argument, we can show that $\rho_e(m) \geq m/8$ in the cases where e has no left-neighbor or right-neighbor (or both). As mentioned earlier in the proof, these cases are in fact simpler than the case considered in the above proof. \square

5. ONLINE COLORING OF INTERVALS WITH BANDWIDTH

In this section, we consider the problem of online coloring interval graphs with bandwidth. Adamy and Erlebach [Adamy and Erlebach 2004] use a variant of first-fit to obtain a 195-competitive algorithm for the problem. By modifying the column construction procedure from Section 4, we show that the Adamy-Erlebach algorithm is 35-competitive. In fact, we show that the Adamy-Erlebach algorithm with a small modification is 30-competitive. These are not necessarily the best algorithms for the problem. Narayanaswamy [Narayanaswamy 2004; Azar et al. 2006] presented an algorithm that is 10-competitive for this problem, and this is the best bound known.

For any instance \mathcal{I} of the interval coloring problem with bandwidths, let $OPT(\mathcal{I})$ denote the fewest number of colors needed to color the intervals in \mathcal{I} such that the bandwidth constraints are satisfied. Whenever \mathcal{I} is obvious from the context, we simply use OPT . Adamy and Erlebach [Adamy and Erlebach 2004] present an online algorithm for solving the interval coloring problem with bandwidths that uses at most $195 \cdot OPT$ colors. The Adamy-Erlebach algorithm is a combination of the First-Fit algorithm and the Kierstead-Trotter algorithm. When an interval x arrives, if $b(x) \leq 1/2$, use the First-Fit rule to assign a color to x from a set C_1 of colors. Otherwise, if $b(x) > 1/2$, ignore the bandwidth of x and use the Kierstead-Trotter algorithm to assign a color to x from a set C_2 of colors. Here C_1 and C_2 are disjoint.

Adamy and Erlebach derive their approximation factor of 195 as follows. Let $\mathcal{I}_1 = \{x \in \mathcal{I} \mid b(x) \leq 1/2\}$ and $\mathcal{I}_2 = \{x \in \mathcal{I} \mid b(x) > 1/2\}$. No two intersecting intervals in \mathcal{I}_2 can be assigned the same color. Therefore, $OPT(\mathcal{I}_2) = \chi(\mathcal{I}_2)$. As mentioned above, the Kierstead-Trotter algorithm will use at most $3 \cdot \chi(\mathcal{I}_2)$ colors and therefore the number of colors used, $|C_2| \leq 3 \cdot OPT(\mathcal{I}_2) \leq 3 \cdot OPT(\mathcal{I})$. Most of the Adamy-Erlebach paper is devoted to showing that the First-Fit algorithm uses at most $192 \cdot OPT(\mathcal{I}_1)$ colors. The fact that $|C_1| \leq 192 \cdot OPT(\mathcal{I})$ implies that $|C_1| + |C_2| \leq 195 \cdot OPT(\mathcal{I})$.

5.1 Improved Analysis

In the following, we analyze the family of algorithms obtained by using a threshold $\alpha \in (0, 1)$ instead of $1/2$ in the Adamy-Erlebach algorithm. For each $\alpha \in (0, 1)$ we call the corresponding instance of the algorithm, the α -Adamy-Erlebach algorithm. The α -Adamy-Erlebach algorithm colors an interval with the First-Fit algorithm using a color palette C_1 if its bandwidth is at most α ; otherwise the interval is colored using the Kierstead-Trotter algorithm using a color palette C_2 . Our main result, proved in Section 5.2, is the following.

THEOREM 5.1. *For any set \mathcal{I} of intervals whose maximum bandwidth is $\alpha \in (0, 1)$, the First-Fit algorithm uses at most*

$$\frac{16}{(1-\alpha)} \cdot OPT(\mathcal{I}) + 1$$

number of colors.

Using this result we show the following.

THEOREM 5.2. *The α -Adamy-Erlebach algorithm uses at most*

$$\left(\frac{16}{(1-\alpha)} + 3(\lceil \frac{1}{\alpha} \rceil - 1) \right) \cdot OPT(\mathcal{I}) + 1$$

colors.

PROOF. Let $\mathcal{I}_1 = \{x \in \mathcal{I} \mid b(x) \leq \alpha\}$ and $\mathcal{I}_2 = \{x \in \mathcal{I} \mid b(x) > \alpha\}$. For intervals in \mathcal{I}_2 the α -Adamy-Erlebach algorithm ignores bandwidths and produces a coloring by using the Kierstead-Trotter algorithm. Hence, $|C_2| \leq 3 \cdot \chi(\mathcal{I}_2)$. Since $b(x) > \alpha$ for all $x \in \mathcal{I}_2$, in any coloring of \mathcal{I}_2 that satisfies the bandwidth constraint, a clique of size at most $\lceil 1/\alpha \rceil - 1$ can be colored using one color. Since \mathcal{I}_2 contains a clique of size $\chi(\mathcal{I}_2)$, any coloring of \mathcal{I}_2 that satisfies the bandwidth constraints, uses at least $\chi(\mathcal{I}_2)/(\lceil 1/\alpha \rceil - 1)$ colors. Therefore, we have

$$\chi(\mathcal{I}_2) \leq (\lceil 1/\alpha \rceil - 1) \cdot OPT(\mathcal{I}_2)$$

and therefore

$$|C_2| \leq 3 \cdot (\lceil 1/\alpha \rceil - 1) \cdot OPT(\mathcal{I}_2).$$

□

Substituting $\alpha = 1/2$ in the above expression we get an upper bound of $35 \cdot OPT + 1$ on the number of colors used by the Adamy-Erlebach algorithm.

COROLLARY 5.3. *The Adamy-Erlebach algorithm uses at most $35 \cdot OPT + 1$ colors.*

The function $\left(\frac{16}{(1-\alpha)} + 3(\lceil \frac{1}{\alpha} \rceil - 1) \right)$ has a minimum at roughly $\alpha = 1/3$ in the range $(0, 1)$. Substituting $\alpha = 1/3$ in this expression we get an upper bound of $30 \cdot OPT + 1$ on the number of colors used by the $1/3$ -Adamy-Erlebach algorithm.

COROLLARY 5.4. *The $1/3$ -Adamy-Erlebach algorithm uses at most $30 \cdot OPT + 1$ colors.*

5.2 Column Construction for Intervals with Bandwidth

In this section, we present a proof of Theorem 5.1 by showing that the column construction procedure of Section 4 and the analysis can be extended for use in the case of intervals with bandwidths as well. This parallels the work Adamy and Erlebach, whose analysis of the First-Fit algorithm for coloring intervals with bandwidth at most $1/2$ is a modification of the “centrality approach” used by Kierstead [Kierstead 1988; Kierstead and Qin 1995].

Without loss of generality, we assume that there is a positive integer N such that each interval in \mathcal{I} is $[x, y]$ for some integers x and y , $0 \leq x, y \leq N$. As

before, we denote by \mathcal{E} the set of *elementary* intervals $\{[x - 1, x] \mid 1 \leq i \leq N\}$. Suppose the First-fit algorithm uses colors $1, 2, \dots, m$ to color the intervals in \mathcal{I} . In the following we will show that there is an elementary interval e and a set of colors $C \subseteq \{1, 2, \dots, m\}$ such that $|C| \geq (m - 1)/8$ and for each color $c \in C$, $\text{density}_c(e) \geq (1 - \alpha)/2$, where $\text{density}_c(e)$ is the sum of the bandwidths of intervals in \mathcal{I} that contain e and are colored c . This implies that $\text{density}(e) = \sum_{c \in C} \text{density}_c(e) \geq (m - 1)(1 - \alpha)/16$. The key property of the first-fit coloring that we will repeatedly use in the analysis is:

First-Fit Property

If an interval x is colored $j \geq 1$, then for any color i , $1 \leq i < j$, there is an elementary interval e contained in x such that $\text{density}_i(e) \geq (1 - \alpha)$.

Now we describe the *column construction* procedure for intervals with bandwidth. The construction procedure is similar to the one described in Section 4, with a key difference. Rules 2 and 3 are identical to the corresponding rules in Section 4, while Rule 1 is modified to take into consideration the density of the intervals. The new Rule 1 is as follows. For notational convenience, let $C_0 = \mathcal{E}$.

Rule 1. For each $e \in C_{i-1}$, if $\text{density}_i(e) \geq (1 - \alpha)/2$, then e is added to C_i with a label “R” at height i .

We now show that even with this modified rule, there is at least one column that grows to a height of $(m - 1)$. Using Lemma 4.3 and Theorem 4.4, we obtain that among columns of height $m - 1$, there is one column, say e , that contains at least $(m - 1)/8$ “R” labels. Each “R” corresponds to a distinct color i such that $\text{density}_i(e) \geq (1 - \alpha)/2$, implying that $\text{density}(e) \geq (m - 1)(1 - \alpha)/16$.

LEMMA 5.5. *For $0 \leq i < j \leq m$, and any interval $I \in \mathcal{S}$ that is assigned color j , there is a column $e \in C_i$ that I intersects.*

PROOF. By induction on i . Since $C_0 = \mathcal{E}$, the base case $i = 0$ is straightforward. For the induction step, suppose $k \geq 1$, and the lemma holds for $i \leq k - 1$. Consider an interval I that is colored $j > k$. By the First-Fit property, I contains an elementary interval e such that $\text{density}_k(e) \geq 1 - \alpha$. The induction hypothesis implies that C_{k-1} is non-empty. If $e \in C_{k-1}$, then $e \in C_k$ by Rule 1 and the induction step is complete. Suppose therefore that $e \notin C_{k-1}$, and that a and b are the left and right neighbors of e in C_{k-1} . (The case where one of the neighbors is absent is handled similarly.) By the induction hypothesis, each interval that For the induction step, suppose $k \geq 1$, and the lemma holds for $i \leq k - 1$. Consider an interval I that is colored $j > k$. By the First-Fit property, I contains an elementary interval e such that $\text{density}_k(e) \geq 1 - \alpha$. The induction hypothesis implies that C_{k-1} is non-empty. If $e \in C_{k-1}$, then $e \in C_k$ by Rule 1 and the induction step is complete. Suppose therefore that $e \notin C_{k-1}$, and that a and b are the left and right neighbors of e in C_{k-1} . (The case where one of the neighbors is absent is handled similarly.) By the induction hypothesis, each interval that is colored k and contains e must contain either a or b . Thus, either $\text{density}_k(a) \geq (1 - \alpha)/2$, or $\text{density}_k(b) \geq (1 - \alpha)/2$. Suppose the former holds. Then $a \in C_k$ by Rule 1 and $b \in C_k$ either by Rule 1 or Rule 2. Now the induction hypothesis (applied to

$i = k - 1$ and j) tells us that I contains either a or b . Thus the induction step is complete. \square

6. CONCLUSION

In this paper, we have introduced the problem of max-coloring of interval graphs, and shown NP-hardness and a simple 2-approximation algorithm. We introduced a column construction procedure that yields an $8\chi(G)$ bound for online coloring interval graphs, improving the earlier bound of Kierstead and Qin [Kierstead and Qin 1995]. We have studied the problem of online coloring of intervals with bandwidth and showed that the column construction procedure can be applied in this case as well to improve bounds.

It is still an open problem to close the gap for online coloring of interval graphs. The best known lower bound is $4.4\chi(G)$ by Chrobak and Slusarek [Chrobak and Ślusarek 1988]. The column construction procedure doesn't seem to give bounds better than $8\chi(G)$ and some new ideas are required. Similarly, the best lower bounds we can obtain for online coloring of intervals with bandwidths is 6 by slightly modifying the construction of Chrobak and Slusarek. It is an open question whether we can close the gap in this case as well.

We have shown a simple 2-approximation algorithm for max-coloring interval graphs. It would be interesting to either improve the approximation or show that a hardness of approximation result.

Acknowledgements.

We thank Narayanaswamy and Subhash Babu for letting us present their clever improvement [Narayanaswamy and Babu 2004] of our analysis of First-Fit in the preliminary version [Pemmaraju et al. 2004]. We also thank Brightwell, Kierstead, and Trotter for sharing their improved analysis [Brightwell et al. 2003] with us. Finally, we thank the anonymous referees whose suggestions have improved the paper.

REFERENCES

- ADAMY, U. AND ERLEBACH, T. 2004. Online coloring of intervals with bandwidth. In *First International Workshop on Approximation and Online Algorithms, (WAOA 2003)*, K. Jansen and R. Solis-Oba, Eds. Lecture Notes in Computer Science, vol. 2909. Springer, 1–12.
- AZAR, Y., FIAT, A., LEVY, M., AND NARAYANASWAMY, N. S. 2006. An improved algorithm for online coloring of intervals with bandwidth. *Theor. Comput. Sci.* 363, 1, 18–27.
- BRIGGS, P. 1992. Register allocation via graph coloring. Ph.D. Thesis, Rice University.
- BRIGHTWELL, G., KIERSTEAD, H. A., AND TROTTER, W. T. 2003. A note on the first-fit coloring of interval graphs. personal communication.
- BUCHSBAUM, A. L., KARLOFF, H., KENYON, C., REINGOLD, N., AND THORUP, M. 2004. OPT versus LOAD in dynamic storage allocation. *SIAM J. Comput.* 33, 3, 632–646.
- CHAITIN, G. 1982. Register allocation and spilling via coloring. *ACM SIGPLAN Notices* 17, 6, 98–105.
- CHROBAK, M. AND ŚLUSAREK, M. 1988. On some packing problems related to dynamic storage allocation. *Informatique théorique et Applications/Theoretical Informatics and Applications* 22, 4, 487–499.
- CORNEIL, D., OLARIU, S., AND STEWART, L. 1998. The ultimate interval graph recognition algorithm? (extended abstract). In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*. 175–180.

- GAREY, M. AND JOHNSON, D. 1979. *Computers and Intractability: A Guide to the theory of NP-completeness*. W.H. Freeman and Company, San Francisco.
- GAREY, M., JOHNSON, D., MILLER, G., AND PAPADIMITRIOU, C. H. 1980. The complexity of coloring circular-arcs and chords. *SIAM Journal of Algebraic Discrete Methods* 1, 2, 216–227.
- GERGOV, J. 1996. Approximation algorithms for dynamic storage allocation. In *Proceedings of the 4th European Symposium on Algorithms: Lecture Notes in Computer Science 1136*. 52–61.
- GERGOV, J. 1999. Algorithms for compile-time memory optimization. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*. S907–S908.
- GOLUMBIC, M. 1980. *Algorithmic graph theory and perfect graphs*. Academic Press, NY.
- GOODMAN, J. R. AND HSU, W.-C. 1988. Code scheduling and register allocation in large basic blocks. In *ICS '88: Proceedings of the 2nd International conference on Supercomputing*. ACM, New York, NY, USA, 442–452.
- GOVINDARAJAN, R. AND RENGARAJAN, S. 1996. 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*.
- GUAN, D. AND ZHU, X. 1997. A coloring problem for weighted graphs. *Information Processing Letters* 61, 77–81.
- GUO, J., GARZARÁN, M., AND PADUA, D. 2004. The power of Belady's algorithm in register allocation for long basic blocks. In *LCPC 2003: Proceedings of the 16th International Workshop on Languages and Compilers for Parallel Computing: Springer Lecture Notes in Computer Science: 2958/2004*. 374–390.
- IRANI, S. 1994. Coloring inductive graphs on-line. *Algorithmica* 11, 1, 53–72.
- KIERSTEAD, H. 1988. The linearity of first-fit coloring of interval graphs. *SIAM J. Discrete Math* 1, 526–530.
- KIERSTEAD, H. 1991. A polynomial time approximation algorithm for dynamic storage allocation. *Discrete Mathematics* 88, 231–237.
- KIERSTEAD, H. AND QIN, J. 1995. Coloring interval graphs with first-fit. *Discrete Mathematics* 144, 47–57.
- KIERSTEAD, H. AND TROTTER, W. 1981. An extremal problem in recursive combinatorics. *Congressus Numerantium* 33, 143–153.
- LEE, J., PALSBERG, J., AND PEREIRA, F. 2007. Aliased register allocation for straight-line programs is NP-complete. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP): Springer Lecture Notes in Computer Science: 4596/2007*. 680–691.
- MARX, D. 2005. A short proof of the NP-completeness of minimum sum interval coloring. *Operations Research Letters* 33, 4, 382–384.
- MCCONNELL, R. M. 2003. Linear-time recognition of circular-arc graphs. *Algorithmica* 37, 2, 93–147.
- NARAYANASWAMY, N. 2004. Dynamic storage allocation and on-line coloring interval graphs. In *Tenth International Computing and Combinatorics Conference, (COCOON)*.
- NARAYANASWAMY, N. AND BABU, R. S. 2004. Analysis of first-fit coloring of interval graphs. personal communication.
- PEMMARAJU, S., RAMAN, R., AND VARADARAJAN, K. 2004. Buffer minimization using max-coloring. In *Proceedings of The ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 562–571.
- POLETTI, M. AND SARKAR, V. 1999. Linear scan register allocation. *ACM Trans. Program. Lang. Syst.* 21, 5, 895–913.
- SETHI, R. 1973. Complete register allocation problems. In *STOC '73: Proceedings of the fifth annual ACM symposium on Theory of computing*. ACM, New York, NY, USA, 182–195.

Received Month Year; revised Month Year; accepted Month Year