Buffer Minimization using Max-Coloring*

Sriram V. Pemmaraju Rajiv Raman Kasturi Varadarajan

October 7, 2003

Abstract

Given a graph $G = (V, E)$ and positive integral vertex weights $w : V \to \mathbf{N}$, the *max-coloring problem* seeks to find a proper vertex coloring of G whose color classes $C_1, C_2, \ldots, C_k$, minimize $\sum_{i=1}^{k} max_{v \in C_i} w(v)$. 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. Specifically, companies have tried to solve this problem in the design of memory managers for wireless protocol stacks such as GPRS or 3G.

Though this problem seems similar to the well-known dynamic storage allocation problem, we point out 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 a 10-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 $10 \cdot \chi(G)$ 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.

1 Introduction

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 $w : V \to N$ 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 $w_1, w_2, \ldots, w_k$. The problem is to determine if there is a k-coloring of G into color classes $C_1, C_2, \ldots, C_k$ 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 $w : V \to \mathbf{N}$ the problem is to find a proper vertex coloring $C_1, C_2, \ldots, C_k$ of G that minimizes $\sum_{i=1}^{k} \max_{v \in C_i} w(v)$. 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 $i = 1, 2, \ldots, k$, 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

*All three authors are at the Department of Computer Science, The University of Iowa, Iowa City, IA 52240-1419. E-mail: [sriram, rraman, kvaradar]@cs.uiowa.edu. The first two authors have been partially supported by NSF Grant DMS-0213305. The last author has been partially supported by an NSF CAREER award CCR-0237431.

size of the total memory block needed by extracting large traces of memory requests and running off-line algorithms on these traces[1].

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.

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 endpoints 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 $W + 2w$.

As far as we know, prior to this paper, the max-coloring problem has been studied only in [7]. Like us, the authors of that paper are motivated by the problem of allocating a small buffer, but their problem arises in the context of digital signal processing applications. [7] experimentally evaluates 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 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 in 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 $|\cup_{v \in V} I(v)|$. Stockmeyer proved this problem NP-complete in 1976 (see problem SR2 in Garey and Johnson [4]) and Kierstead presented the first constant-factor approximation algorithm in 1988 [11]. This was an 80-approximation algorithm that used a first-fit strategy to perform on-line coloring of unweighted interval graphs. Kierstead [12] subsequently improved this to a 6-approximation algorithm, which was then improved by Gergov [5, 6] to a 5-approximation and then a 3-approximation algorithm. Recently, Buchsbaum et.al., [1] presented a 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 [1] 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 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

[1]Due 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.

[2]Without 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 [3] for a recent algorithm.

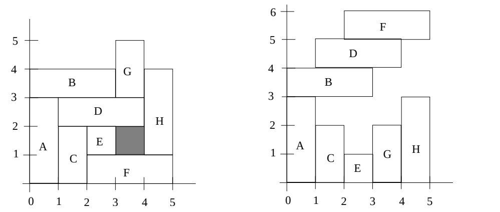

Figure 1: On the left is a rectangle packing corresponding to an optimal interval coloring, with weight 5. On the right is a rectangle packing corresponding to an optimal max-coloring. In the packing on the right the rectangles are packed into 4 strips: $S_1 = (0, 3)$, $S_2 = (3, 4)$, $S_3 = (4, 5)$, and $S_4 = (5, 6)$, for a total weight of 6.

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 [6] shows that $OPT_I \leq 3 \cdot LOAD$. Buchsbaum et.al. [1] further investigate the relationship between $LOAD$ and OPT_I and show that $OPT = LOAD + O((w_{\max}/LOAD)^{1/7}) \cdot LOAD$, 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, $Q_1, Q_2, \ldots, Q_n$, 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 $C_1, C_2, \ldots, C_n$, where C_i contains exactly one interval each from $Q_n, Q_{n-1}, \ldots, Q_i$. 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 nth 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.

The rest of the paper is organized into three sections. In Section 2 we make a connection between max-coloring and on-line coloring and using this we present a 3-approximation and then a 2-approximation algorithm for max-coloring on interval graphs. In Section 3 we present an analysis of the first-fit algorithm for the on-line coloring problem on interval graphs and show

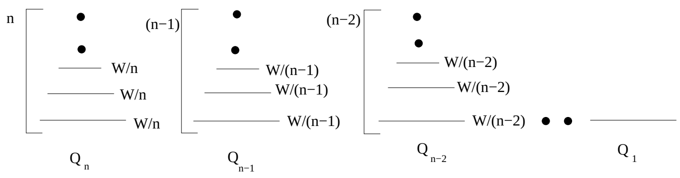

Figure 2: An example of a weighted interval graph for which $OPT_M = \Omega(LOAD \cdot \log n)$.

an upper bound of $10 \cdot \chi(G)$ on the number of colors used for any interval graph G. This analysis shows that the first-fit strategy produces a 10-approximation for max-coloring, but more importantly it provides the best known upper bounds on the number of colors used by first-fit. The analysis of first-fit has a long history: following a sequence of papers showing super-linear bounds, Kierstead [11] showed that first-fit uses at most $40 \cdot \chi(G)$ colors. Kierstead and Qin [13] refined the analysis of [11] to get an improved bound of $26 \cdot \chi(G)$, and expressed the belief that further improvements should be possible. The best-known lower bound on the number of colors used by first-fit to do on-line coloring of an interval graph is $4.4\chi(G)$ [2]. Our new analysis of the first-fit strategy, inspired by some ideas in Gergov [6], is quite different from the analysis in [11, 13] and in our view, much simpler. In Section 4 we show the NP-hardness of max-coloring on interval graphs.

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 below.

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

3. Return the coloring produced by A.

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 class*[3] *of graphs and let A be an algorithm for on-line graph coloring problem such that for some integer constant $c > 0$ and for any graph $G \in \mathcal{C}$, A k-colors G for some $k \leq c \cdot \chi(G)$. 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 $C_1, C_2, \ldots, C_k$ 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 $w_1 \geq w_2 \geq \cdots \geq w_k$. Now note that $k \geq \chi(G)$ and $OPT_M(G) = \sum_{i=1}^{k} w_i$. Let $A_1, A_2, \ldots, A_t$ 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 $a_1 \geq a_2 \geq \cdots \geq a_t$. From our hypothesis it follows that $t \leq c \cdot \chi(G) \leq c \cdot k$. For notational convenience, define sets $A_{t+1} = A_{t+2} = \ldots = A_{c \cdot \chi(G)} = \emptyset$ 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 $C_1, C_2, \ldots, C_k$ of G, restricted to V_i is an ℓ-coloring, for some $\ell \leq i - 1$, of the induced subgraph $G[V_i]$. 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 $c \cdot \ell \leq c \cdot (i-1)$ 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. □

In 1981, Kierstead and Trotter [14] 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 the above connection, a 3-approximation algorithm for max-coloring interval graphs follows.

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

The connection proved in Theorem 2.1 leads to approximation algorithms for max-coloring for other classes of graphs as well. For example, using the results of Irani [10] on the on-line coloring of d-inductive graphs we get $O(\log n)$-approximation algorithms for max-coloring on chordal graphs. Similarly, using the algorithm of Lovász, Saks, and Trotter [15] for the on-line coloring problem on arbitrary graphs, we get an $O(n/\log^* n)$-approximation algorithm for max-coloring on arbitrary 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 $S_1, S_2, \ldots$ such that when a vertex u is presented, it finds the smallest i such that $S_1 \cup S_2 \cup \ldots \cup (S_i \cup \{u\})$ 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 for $i \geq 2$ and S_1 is an independent set. 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 new algorithm for max-coloring is given below.

BETTER-MCA(G, w)
1. Sort the vertices of G in non-increasing order of weights. (Let $(v_1, v_2, \ldots, v_n)$ be this ordering of the vertices of G.)
2. **for** $j \leftarrow 1$ **to** n **do**
3. Insert v_j into set S_i, where i is the smallest value such that $S_1 \cup S_2 \cup \ldots \cup (S_i \cup \{u\})$ does not contain an $(i+1)$-clique
endfor
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
endfor
7. Return the coloring

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 the fewest possible colors.

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

Proof: Let $C_1, C_2, \ldots, C_k$ be a coloring of G that is optimal for the max-coloring problem. Let $w_i =$

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

$\max_{v \in C_i} w(v)$ and without loss of generality assume that $w_1 \geq w_2 \geq \cdots \geq w_k$. 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 $S_1, S_2, \ldots, S_t$. An element u is inserted into S_t only because $S_1 \cup S_2 \cup \ldots \cup (S_{t-1} \cup \{u\})$ 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, $s_i > w_i$ and let i be the smallest such value. This implies any interval x with weight s_i or larger is in an earlier color class, C_j, for some $j < i$ in the optimal coloring. Therefore, intervals with weight s_i or larger induce a clique of size at most $i - 1$ and therefore in Step (3) no interval with weight s_i will get inserted into S_i - a contradiction.

In Steps (4) and (5) we convert the vertex partition $S_1, S_2, \ldots, S_t$ 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. □

From Theorem 2.3, we can easily obtain a 3-approximation for the max-coloring problem on circular arc graphs. Consider a circular arc representation of the graph and pick some arbitarary point p on the circle. All the circular arcs that contain p are assigned a distinct color. The remaining circular arcs induce an interval graph, which we color using the algorithm of Theorem 2.3 using a palette of colors distinct from the colors used for arcs containing p. It is easy to see that this is a 3-approximation.

3 Analysis of First-Fit

The first-fit algorithm for on-line coloring assigns the smallest available color to each presented vertex. This is a natural strategy for on-line coloring and has been analyzed extensively [10, 11, 13] for different graph classes. 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 [13] who show that the first-fit algorithm uses no more than $26\chi(G)$ 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 [7] 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 $4.4\chi(G)$ [2]. This along with the belief expressed in [13] that the upper bound of $26 \cdot \chi(G)$ could be improved, is the motivation for new analysis of the first-fit strategy. Our analysis, inspired by some ideas in Gergov [6], is quite different from the analysis in [11, 13] and in our view, much simpler. We show the following result, which leads to a 10-approximation algorithm for max-coloring on interval graphs.

THEOREM 3.1. *For any interval graph G, the first-fit strategy for on-line coloring of G uses at most $10 \cdot \chi(G)$ colors.*

Let $\mathcal{S}$ be a set of intervals on the real line that correspond to the interval graph G. We assume without loss of generality that each interval in $\mathcal{S}$ 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 $\{[i-1, i] | 1 \leq i \leq N\}$. 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.

Suppose that the first-fit algorithm uses colors $1, \ldots, m$ to color the intervals in $\mathcal{S}$ when they are presented in some arbitarary order. We will now argue that there is an elementary interval that is contained in at least $m/10$ input intervals. Note that this corresponds to a clique of size at least $m/10$ in the corresponding interval graph, which means $m/10$ is a lower bound on the size of 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 a rectangle $\{(x, y) | l \leq x \leq r, k-1 \leq y \leq k\}$ 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 is based on a construction of a set of "columns" corresponding to the first-fit coloring. 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. A column has a positive integral *height* associated with it. If a column has height t, we say that it is *active* at heights $1, \ldots, t$ and *inactive* at heights $t+1, \ldots, m$. (The height of any column will be at most m.) 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. It is useful to visualize a column e of height t as a rectangle $\{(x,y)|l(e) \leq x \leq r(e), 0 \leq y \leq t\}$ of width 1, 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 $1 \leq i \leq t$. It is worth pointing out that the goal of the column construction procedure is to find a column with at least $m/10$ "R" labels in it.

The column construction procedure works by starting with a set of columns that are active at height 1 and, for $2 \leq i \leq m$, choosing a subset of the active columns at height $i-1$ to be the active columns at height i. 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. 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 $2 \leq i \leq m$, the following rules specify which columns from C_{i-1} are picked in 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 with a label "R" at height i.

2. For each remaining $e \in C_{i-1}$, if e is the left neighbor or right neighbour 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. See Figure 3. Note that for such a column e, either its left or right neighbour in C_i must have an "R" label (even though more columns may be added to C_i by the next rule).

3. For each remaining $e \in C_{i-1}$, let e' be the left-neighbor of e in C_{i-1}. If e' is undefined, we proceed to inspect the right neighbor of e in C_{i-1}. Suppose that e' is the left neighbor of e in $C_j, \ldots, C_{i-1}$, and is not the left neighbor of e in $C_1, \ldots, C_{j-1}$; note that such a j does exist (and could be 1). If the number of "R" labels of e at heights $j, \ldots, i-1$ 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 e'' is undefined, e is not added to C_i. Otherwise, suppose that e'' is the right neighbor of e in $C_k, \ldots, C_{i-1}$, and is not the left neighbor of e in $C_1, \ldots, C_{k-1}$. If the number of "R" labels of e at heights $k, \ldots, i-1$ 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 3.

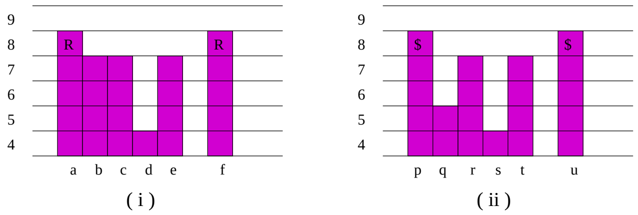

Figure 3: (i) A snapshot in the construction of C_8 after Rule 1 has been applied. Columns b and e will get added to C_8 with \$ labels at height 8 due to Rule 2. (ii) Snapshot after Rule 2 has been applied. Column p is the left neighbour of r in C_6, C_7 and t is the right neighbour of r in C_5, C_6,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

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 3.1. *Let $1 \leq i \leq j \leq m$, 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 $j \geq i$. If $j = i$, the induction step follows easily from the induction hypothesis and Rule 1. Suppose $j > i$. 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. □

The invariant implies that $C_1, \ldots, C_m$ are non-empty. For a column e with height at least j and $1 \le i \le j$, 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). Let $\rho_e(j) = \rho_e(1,j)$, $\delta_e(j) = \delta_e(1,j)$, and $\phi_e(j) = \phi_e(1,j)$. Define $\rho_e(0) = \delta_e(0) = \phi_e(0) = 0$.

LEMMA 3.2. *For any column $e \in C_i$, $\rho_e(i) \ge \frac{1}{4}(\rho_e(i) + \phi_e(i))$.*

Proof: The proof is by induction on i. The base cases $i = 0, 1$ are easily verified. Suppose $i \ge 2$ and the lemma is true for all $0 \le i' < i$. 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 $1 \le j \le i-1$ such that $\rho_e(j, i-1) > (i-j)/4$. This implies that $\rho_e(j,i) \ge (i-j+1)/4$. From this and the induction hypothesis, it follows that

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

□

We are now ready to obtain our main result.

THEOREM 3.2. *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 $m/10$ in the corresponding interval graph.*

Proof: We will show that there is a column $e \in C_m$ such that $\rho_e(m) \ge m/10$. Let e be the column in C_m that is last in the natural ordering. Thus e has no right neighbor in C_m. Suppose first that e has a left neighbor in C_m. Suppose $f_1, \ldots, f_a$ are columns such that for $1 \le i \le a$, f_i is the left neighbor of e in $C_{t_{i-1}+1}, \ldots, C_{t_i}$, where $t_0 = 0 < t_1 < \cdots < t_a = m$. Similarly, suppose $e_1, \ldots, e_b$ are columns such that for $1 \le i \le b$, e_i is the right neighbor of e in $C_{n_{i-1}+1}, \ldots, C_{n_i}$, where $0 = n_0 < n_1 < \cdots < n_b < m$.

If $\rho_{f_a}(t_{a-1}+1, t_a) \ge m/10$, we are done since $f_a \in C_m$ and $\rho_{f_a}(m) \ge \rho_{f_a}(t_{a-1}+1, t_a) \ge m/10$. So let us assume that $x = \rho_{f_a}(t_{a-1}+1, t_a) < m/10$. We claim that

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

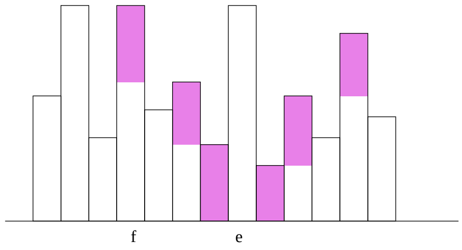

Figure 4: The columns at the end of the construction procedure. The number of \$ labels in column e is bounded by the number of R labels in all the shaded rectangles. Rule 3 implies that in each shaded rectangle except the one contained in column f, the number of R labels is at most one-fourth of the height of the rectangle.

The claim follows from the observation that for e to be labeled with a \$ at height i, either its left or right neighbor in C_i must be labeled R at height i (Rule 2). See Figure 4 for an illustration. Now for any $1 \le i \le a-1$, the column f_i is active at height t_i but not at height $t_i + 1$, because f_i is the left neighbor of e in C_{t_i} but not in C_{t_i+1}. Since e is the right neighbor of f_i in $C_{t_{i-1}+1}, \ldots, C_{t_i}$ and not in $C_1, \ldots, C_{t_{i-1}}$, and f_i became inactive in C_{t_i+1}, we have by Rule 3 that

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

By an identical argument, we have that for any $1 \le i \le b$,

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

We thus obtain that

$$\begin{aligned}
\delta_e(m) &\le x + \sum_1^{a-1} \frac{1}{4}(t_i - t_{i-1}) + \sum_1^{b} \frac{1}{4}(n_i - n_{i-1}) \\
&\le x + \frac{1}{4}(t_{a-1} + t_b) \\
&\le x + \frac{1}{4}(m - x + m) \\
&= 3x/4 + m/2 \\
&\le 23m/40.
\end{aligned}$$

We use above the fact that $t_{a-1} \le m - x$. Thus $\rho_e(m) + \phi_e(m) \ge 17m/40$. Using Lemma 3.2, we obtain that $\rho_e(m) \ge 17m/160 \ge m/10$.

The case where e has no left-neighbor in C_m is in fact easier. By a similar argument, we get $\rho_e(m) \ge m/8$ in this case. □

Remark: By optimizing x, we get an improved bound of $2m/19$ in Theorem 3.2. Some further small improvements seem possible, but it appears that new ideas will be needed to make our basic approach yield anything better than $m/8$.

4 Hardness Results

In this section we discuss the hardness of max-coloring on interval graphs. Our main result is an NP-hardness proof for max-coloring that uses a reduction from the problem, E4-SET SPLITTING [8, 9], defined below. The very high level idea of the reduction follows that of [16].

E4-SET SPLITTING
INPUT: A universe U and a family $\mathcal{F}$ of subsets of U, each of size 4.
QUESTION: Is there a partition of U into U_1 and U_2 such that this partition splits each set $S \in \mathcal{F}$ (that is, $U_0 \cap S \neq \emptyset$ and $U_1 \cap S \neq \emptyset$)?

THEOREM 4.1. *The max-coloring problem for interval graphs is NP-complete.*

Proof: Let $U = \{x_1, x_2, \ldots, x_n\}$ and let $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$. As part of our construction, we construct a number of gadgets, each of which is a set of weighted intervals. Corresponding to each element x_i, we have a gadget G_i, called an *element gadget*. In addition we have two gadgets, P_0 and P_1, which we call *partition gadgets*. For each subset S_j that x_i belongs to, G_i contains a small number (5, to be exact) of weighted intervals. We denote this subset of G_i by G_{ij}. Suppose that element x_i belongs to subsets $S_{j_1}, S_{j_2}, \ldots, S_{j_k}$. In addition to the disjoint subsets, $G_{ij_1}, G_{ij_2}, \ldots, G_{ij_k}$, the gadget G_i contains sets of *connector intervals* $C_{i0}, C_{i1}, \ldots, C_{ik}$, such that the connector intervals in C_{ip} are "in-between" those in G_{ij_p} and $G_{ij_{p+1}}$. In summary, we have

$$G_i = (\cup_{p=1}^{k} G_{ij_p}) \cup (\cup_{p=0}^{k} C_{ip}).$$

Variable gadget G_i. We now describe the variable gadget G_i in detail. Suppose that element x_i belongs to subsets $S_{j_1}, S_{j_2}, \ldots, S_{j_k}$, where $1 \le j_1 < j_2 < \cdots < j_k \le m$. Let $L_p = 10 \cdot j_p$, let $\varepsilon > 0$ be a constant whose value will be determined later, and let $O_i = (i-1)\varepsilon$. For each p, $1 \le p \le k$, the subset of intervals G_{ij_p} consists of the following 5 weighted intervals:

$$\begin{aligned}
A &= \Big((L_p + O_i, L_p + 1 + O_i), (2i-1)n\Big),\\
B &= \Big((L_p + O_i, L_p + 2 + O_i), (2i-1)n + 1\Big),\\
C &= \Big((L_p + 1 + O_i, L_p + 2 + O_i), (2i-1)n\Big),\\
D &= \Big((L_p + 1 + O_i, L_p + 3 - O_i), 2in\Big),\\
E &= \Big((L_p + 2 + O_i, L_p + 4 - O_i), 2in + 1\Big).
\end{aligned}$$

Each weighted interval is described as a pair $((\ell, r), w)$, where ℓ and r denote the left and right endpoints respectively of the interval and w denotes its weight. O_i is an amount by which each of the intervals is perturbed. We pick ε small enough that perturbing the endpoints of the intervals by O_i, does not change the pairwise intersections of the intervals A, B, C, D, and E. For example, suppose element x_1 occurs in subset S_2. Two colorings of the intervals in G_{12} are shown in Figure 5. Each interval is represented as a rectangle whose height is the weight of the corresponding interval. Since O_i is small enough, it is assumed to be zero in Figure 5. These two colorings will play an important role in the proof that our reduction works and will be called coloring `ZERO` and coloring `ONE` respectively. More precisely, any coloring of G_{ij_p} in which the color classes are $\{D\}, \{B, E\}, \{A, C\}$ is called `ZERO` and any coloring in which the color classes are $\{C, E\}, \{A, D\}, \{B\}$ is called `ONE`.

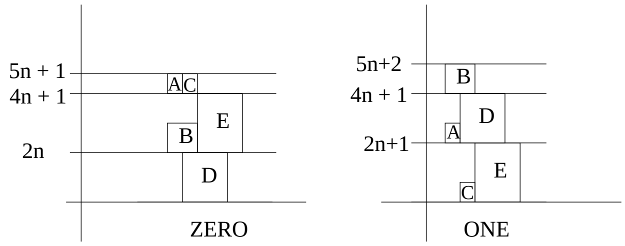

Figure 5: On the left is a 3-coloring of G_{12} with weight $5n+1$ and on the right is a 3-coloring with weight $5n+2$. These colorings are called `ZERO` and `ONE` respectively.

For each p, $1 \le p \le k-1$, the set of connector intervals C_{ip} contains the following 3 weighted intervals:

$$\begin{aligned}
F &= \Big((L_p + 2 + O_i, L_{p+1} + O_i)), (2i-1)n\Big),\\
G &= \Big((L_p + 4 - O_i, L_{p+1} + O_i)), 2in\Big),\\
H &= \Big((L_p + 4 - O_i, L_{p+1} + 1 + O_i), N\Big).
\end{aligned}$$

The "left-most" set of connectors C_{i0} is the following:

$$\begin{aligned}
F &= \Big((0, L_1 + O_i)), (2i-1)n\Big),\\
G &= \Big((0, L_1 + O_i), 2in\Big),\\
H &= \Big((0, L_1 + 1 + O_i), N\Big).
\end{aligned}$$

The "right-most" set of connectors C_{ik} is the following:

$$\begin{aligned}
F &= \Big((L_k + 2 + O_i, 10(m+1))), (2i-1)n\Big),\\
G &= \Big((L_k + 4 - O_i, 10(m+1))), 2in\Big),\\
H &= \Big((L_k + 4 - O_i, 10(m+1)), N\Big).
\end{aligned}$$

The value N will be determined later; for now we only note that N is larger than the weights of all other intervals.

For example, suppose that element x_1 appears in subsets S_2 and S_4. Figure 6 shows two colorings of the intervals in the gadget G_1, both of weight $N + 3n + 1$. In the coloring shown at the top, both G_{12} and G_{14} are colored `ZERO`, while in the coloring at the bottom both G_{12} and G_{15} are colored `ONE`. It turns out that *any* optimal coloring of an element gadget G_i has weight $N + (4i - 1)n + 1$ and furthermore this is achieved only when all of the subsets G_{ij_p} are colored `ONE` or all of the subsets G_{ij_p} are colored `ZERO`. In this manner, in an optimal coloring the element x_i declares its membership in one of the two subsets U_0 or U_1 of the universe U. This phenomena is formalized in the definitions and the claim below.

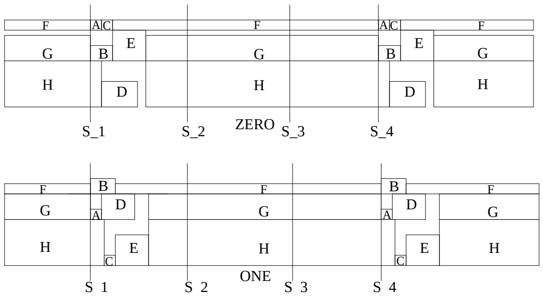

Figure 6: Here are two 3-colorings of G_1 with weight $N + 3n + 1$. In the coloring on the top, both G_{12} and G_{14} are colored `ZERO`, whereas in the coloring in the bottom, both are colored `ONE`.

Let x_i be an element that belongs to subsets $S_{j_1}, S_{j_2}, \ldots, S_{j_k}$. A coloring of the intervals in G_i is said to be *consistent*, if either all subsets $G_{ij_1}, G_{ij_2}, \ldots, G_{ij_k}$ are colored `ZERO` or all subsets $G_{ij_1}, G_{ij_2}, \ldots, G_{ij_k}$ are colored `ONE`. A coloring is said to color a gadget G_i `ZERO` (respectively, `ONE`) if in this coloring all of $G_{ij_1}, G_{ij_2}, \ldots, G_{ij_k}$ are colored `ZERO` (respectively, `ONE`).

LEMMA 4.1. *For any G_i, $1 \le i \le n$, the weight of an optimal coloring of G_i is $N + (4i - 1)n + 1$. Furthermore, any coloring that achieves this weight is consistent.*

More importantly, we show that in any optimal coloring of $\cup_{i=1}^{n} G_i$ the colors used for the subset G_i is distinct from the colors used for any G_j, $j \neq i$. This of course implies that to color $\cup_{i=1}^{n} G_i$ optimally, we need to color each of the G_i's optimally. This is formalized in the claim below.

LEMMA 4.2. *In any optimal coloring of $\cup_{i=1}^{n} G_i$, for any i, j, u, and v, if $u \in G_i$ and $v \in G_j$, $i \neq j$, then u and v are in different color classes. The weight of an optimal coloring of $\cup_{i=1}^{n} G_i$ is $nN + 2n^3 + n^2 + n$.*

The partition gadgets. We will now define the two partition gadgets P_0 and P_1. Both P_0 and P_1 are sets of pairwise disjoint intervals. The set P_0 is:

$$\left\{ \Big((10i + 1 + O_n, 10i + 3 - O_n), (2n+1)n \Big) \mid 1 \le i \le m \right\}$$
$$\cup \left\{ \Big((10i + 3, 10(i+1)), 2n^2 \Big) \mid 1 \le i \le n \right\}$$
$$\cup \left\{ \Big((0, 10i + 1), 2n^2 \Big) \right\}$$

The set P_1 is

$$\left\{ \Big((10i + 2 + O_n, 10i + 4 - O_n), (2n+1)n \Big) \mid 1 \le i \le m \right\}$$
$$\cup \left\{ \Big((10i + 4, 10(i+1)), 2n^2 \Big) \mid 1 \le i \le n \right\}$$
$$\cup \left\{ \Big((0, 10i + 2), 2n^2 \Big) \right\}.$$

It is easy to check that an optimal coloring of $P_0 \cup P_1$ assigns one color to all intervals in P_0 and a second color to all intervals in P_1 for a total weight of $2n(2n + 1)$. The motivation for defining P_0 and P_1 as above is this. Consider an optimal coloring of $\cup_{i=1}^{n} G_i$ and an optimal coloring of $P_0 \cup P_1$ using two colors C_0 and C_1, that are not used in the coloring of $\cup_{i=1}^{n} G_i$. Suppose that C_0 and C_1 are used to color P_0 and P_1, respectively. By Lemma 4.1, in any optimal coloring of $\cup_{i=1}^{n} G_i$, each G_i is colored `ZERO` or `ONE`. Now suppose that for each j, $1 \le j \le m$, there is a z_j such that gadget $G_{z_j j}$ is colored `ZERO`. This allows us to exchange the colors assigned to the interval $D \in G_{z_j j}$ and the interval $(10j + 1 + O_n, 10j + 3 - O_n) \in P_0$, for each j. This exchange leaves the weight of the coloring of $\cup_{i=1}^{n} G_i$ unchanged, but it reduces the weight of color class C_0 from $(2n + 1)n$ to $2n^2$. Similarly, if for each j, $1 \le j \le m$, there is a o_j such that gadget $G_{o_j j}$ is colored `ONE`, then it is possible to exchange the colors of the intervals $E \in G_{o_j j}$ and the interval $(10i + 2 + O_n, 10i + 4 - O_n) \in P_1$. Again, this leaves the weight of the coloring of $\cup_{i=1}^{n} G_i$ unchanged, and decreases the weight of P_1 from $(2n + 1)n$ to $2n^2 + 1$. Therefore, if there is an optimal coloring of $\cup_{i=1}^{n} G_i$ in which for each j, $1 \le j \le m$, there are values z_j and o_j such that $G_{z_j j}$ is colored `ZERO` and $G_{o_j j}$ is colored `ONE`, then there is a coloring of $\cup_{i=1}^{n} G_i \cup P_0 \cup P_1$ of weight $nN + 2n^3 + n^2 + n + 2n^2 + (2n^2 + 1) = nN + 2n^3 + 5n^2 + 1$. This leads to the following lemma.

LEMMA 4.3. *If there is a partition of U into subsets U_0 and U_1 that split S_j for each j, $1 \le j \le m$, then there is a coloring of $\cup_{i=1}^{n} G_i \cup P_0 \cup P_1$ of weight $nN + 2n^3 + 5n^2 + 1$.*

It turns out that the converse of this claim is also true.

LEMMA 4.4. *Suppose there is a coloring of* $\cup_{i=1}^{n} G_i \cup P_0 \cup P_1$ *of weight* $nN + 2n^3 + 5n^2 + 1$. *Then this coloring is optimal and there exists an optimal coloring of* $\cup_{i=1}^{n} G_i$ *in which for each* j, $1 \leq j \leq m$, *there are values* z_j *and* o_j *such that* $G_{z_j j}$ *is colored* `ZERO` *and* $G_{o_j j}$ *is colored* `ONE`.

These claims together yield the fact that the max-coloring problem for interval graphs is NP-hard. □

Our motivation for reducing from E4-SET SPLITTING, beyond the fact that it was convinient to use a "SAT-like" problem, but without negative literals, is the following. Haståd [9] considers MAX E4-SET SPLITTING, the version of E4-SET SPLITTING that seeks to maximize the number of sets split, and shows that for any $\varepsilon > 0$, there is no $(8/7 - \varepsilon)$-approximation for the problem, unless $P = NP$. Furthermore, if we restrict MAX E4-SET SPLITTING to instances in which each element $x \in U$ appears in at most B sets in $\mathcal{F}$, for some constant B, even then there is no $(8/7 - \varepsilon)$-approximation for the problem, unless $P = NP$ (personal communication, Håstad). We believe that we can modify our reduction to obtain a reduction from this restricted version of MAX E4-SET SPLITTING to max-coloring on interval graphs, that would show that existence of constant $\delta > 1$ such that if there is a δ-approximation algorithm for max-coloring on interval graphs, then $P = NP$. The details of this reduction have not been worked out yet.

References

[1] A.L. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold, and M. Thorup. OPT versus LOAD in dynamic storage allocation. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 556–564, 2003.

[2] M. Chrobak and M. Ślusarek. On some packing problems related to dynamic storage allocation. *Informatique théorique et Applications/Theoretical Informatics and Applications*, 22(4):487–499, 1988.

[3] D.G. Corneil, S. Olariu, and L. Stewart. The ultimate interval graph recognition algorithm? (extended abstract). In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 175–180, 1998.

[4] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the theory of NP-completeness.* W.H. Freeman and Company, San Francisco, 1979.

[5] J. Gergov. Approximation algorithms for dynamic storage allocation. In *Proceedings of the 4th European Symposium on Algorithms: Lecture Notes in Computer Science 1136*, pages 52–61, 1996.

[6] J. Gergov. Algorithms for compile-time memory optimization. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pages S907–S908, 1999.

[7] R. Govindarajan and S. Rengarajan. Buffer allocation in regular dataflow networks: An approach based on coloring circular-arc graphs. In *Proceedings of the 2nd International Conference on High Performance Computing*, 1996.

[8] V. Guruswami. Inapproximability results for set splitting and satisfiability problems with no mixed clauses. In *APPROX*, pages 155–166, 2000.

[9] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48:798–859, 2001.

[10] S. Irani. Coloring inductive graphs on-line. *Algorithmica*, 11(1):53–72, 1994.

[11] H.A. Kierstead. The linearity of first-fit coloring of interval graphs. *SIAM J. Discrete Math*, 1:526–530, 1988.

[12] H.A. Kierstead. A polynomial time approximation algorithm for dynamic storage allocation. *Discrete Mathematics*, 88:231–237, 1991.

[13] H.A. Kierstead and J. Qin. Coloring interval graphs with first-fit. *Discrete Mathematics*, 144:47–57, 1995.

[14] H.A. Kierstead and W.T. Trotter. An extremal problem in recursive combinatorics. *Congressus Numerantium*, 33:143–153, 1981.

[15] L. Lovász, M. Saks, and W.T. Trotter. An on-line graph coloring algorithm with sublinear performance ratio. *Discrete Math.*, 75:319–325, 1989.

[16] T. Szkaliczki. Routing with minimum wire length in the dogleg-free manhattan model is np-complete. *SIAM Journal of Computing*, 29(1), 1999.