1 Introduction

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 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 \( G = (V, E) \). 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 \((\ell_i, u_i)\). 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.

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 \( \text{MCA} \) (short for max-coloring algorithm) is given below.

\[
\text{MCA}(G, w)
\]

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 \( \text{MCA} \). 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 1** Let \( C \) be a hereditary class\(^2\) 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 \( G \in C \), \( A \) colors \( G \) with at most \( c \cdot \chi(G) \) colors. Then, for any \( G \in C \) and for any weight function \( w : V(G) \to \mathbb{N} \), \( \text{MCA} \) produces a coloring for \( G \) whose weight is at most \( c \cdot \text{OPT}_M(G) \).

**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

\(^2\)A class \( C \) of graphs is hereditary if \( G \in C \) implies that every induced subgraph of \( G \) is also in \( C \).
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 \( A_{t+1} = A_{t+2} = \cdots = A_{c\cdot\chi(G)} = \emptyset \) 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

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

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 \((i-1)\)-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 \( w_i \).

From this and the induction hypothesis, it follows that

\[
\rho_e(i) = \rho_e(j, i) + \rho_e(j-1) \\
\geq \frac{1}{4}(i - j + 1) + \frac{1}{4}(\rho_e(j-1) + \phi_e(j-1)) \\
\geq \frac{1}{4}(\rho_e(j, i) + \phi_e(j, i)) + \frac{1}{4}(\rho_e(j-1) + \phi_e(j-1)) \\
= \frac{1}{4}(\rho_e(i) + \phi_e(i))
\]

Acknowledgements.

We thank Narayanaswamy and Subhash Babu for letting us present their clever improvement [4] of our analysis of First-Fit in the preliminary version [5]. We also thank Brightwell, Kierstead, and Trotter for sharing their improved analysis [1] with us. Finally, we thank the anonymous referees whose suggestions have improved the paper.

References


