On the Max Coloring Problem*

Leah Epstein† Asaf Levin ‡

May 22, 2010

Abstract

We consider max coloring on hereditary graph classes. The problem is defined as follows. Given a graph $G = (V, E)$ and positive node weights $w : V \to (0, \infty)$, the goal is to find a proper node coloring of G whose color classes $C_1, C_2, \ldots, C_k$ minimize $\sum_{i=1}^{k} \max_{v \in C_i} w(v)$. We design a general framework which allows to convert approximation algorithms for standard node coloring into algorithms for max coloring. The approximation ratio increases by a multiplicative factor of at most e for deterministic offline algorithms and for randomized online algorithms, and by a multiplicative factor of at most 4 for deterministic online algorithms.

We consider two specific hereditary classes which are interval graphs and perfect graphs. For interval graphs, we study the problem in several online environments. In the List Model, intervals arrive one by one, in some order. In the Time Model, intervals arrive one by one, sorted by their left endpoint. For the List Model we design a deterministic 12-competitive algorithm, and a randomized $3e$-competitive algorithm. In addition, we prove a lower bound of 4 on the competitive ratio of any deterministic or randomized algorithm. For the Time Model, we use simplified versions of the algorithm and the lower bound of the List Model, to develop a deterministic 4-competitive algorithm, a randomized e-competitive algorithm, and to design a lower bounds of $\phi \approx 1.618$ on the deterministic competitive ratio and a lower bound of $\frac{4}{3}$ on the randomized competitive ratio. The former lower bounds hold even for unit intervals. For unit intervals in the List Model, we obtain a deterministic 8-competitive algorithm, a randomized $2e$-competitive algorithm and lower bounds of 2 on the deterministic competitive ratio and $\frac{11}{6} \approx 1.8333$ on the randomized competitive ratio.

Finally, we employ our framework to obtain an offline e-approximation algorithm for max coloring of perfect graphs, improving and simplifying a recent result of Pemmaraju and Raman.

1 Introduction

The (offline) max coloring problem is defined as follows: Given a graph $G = (V, E)$ and positive node weights $w : V \to (0, \infty)$, the goal is to find a proper node coloring of G (i.e., each pair of adjacent nodes are assigned distinct colors) whose color classes $C_1, C_2, \ldots, C_k$ minimize $\sum_{i=1}^{k} \max_{v \in C_i} w(v)$.

An interval graph has the property that its nodes can be represented as closed intervals on the real line so that two nodes are adjacent if and only if their respective intervals intersect. Motivated by a design of dedicated memory managers problem, Pemmaraju, Raman and Varadarajan introduced the max coloring problem [17]. In that paper it is mentioned that the problem is actually interesting in the online environment, but it is not studied in that context.

In the online max coloring problem the nodes arrive one by one, and each time a node v arrives the set of edges connecting v to the earlier nodes is revealed. In this paper we consider the online max coloring

*An extended abstract of this paper appeared in the *Proceedings of the 5th Workshop on Approximation and Online Algorithms (WAOA 2007)*.

†Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il.

‡Chaya fellow. Faculty of Industrial Engineering and Management, The Technion, 32000 Haifa, Israel. levinas@ie.technion.ac.il.

problem where G is an interval graph. In this case we assume the graph is given via its intervals representation. The intervals are presented to the algorithm one by one clairvoyantly, i.e., all information regarding the interval is revealed upon arrival. That is, we assume that each time an interval arrives its two endpoints are revealed. Each interval is to be colored before the next one is presented and this color assignment can not be changed afterwards. We are interested in two online versions of the problem. In the List Model, the intervals are given in an arbitrary order. In the Time Model, the intervals arrive sorted by their left endpoints. The study of the Time Model is motivated by the application of the design of memory managers in which each interval corresponds to a memory request that arrives along time (so the requests are ordered according to their left endpoints).

For an algorithm $\mathcal{A}$, we denote its cost by $\mathcal{A}$ as well. An optimal offline algorithm that knows the complete sequence of intervals, as well as its cost, are denoted by OPT. Since the problem is scalable, we consider the absolute competitive ratio and the absolute approximation ratio criteria. For an online algorithm we use the term competitive ratio whereas for an offline algorithm we use the term approximation ratio. The competitive ratio of $\mathcal{A}$ is the infimum $\mathcal{R}$ such that for any input, $\mathcal{A} \leq \mathcal{R} \cdot \mathrm{OPT}$. If $\mathcal{A}$ is randomized, the last inequality is replaced by $E(\mathcal{A}) \leq \mathcal{R} \cdot \mathrm{OPT}$. If the competitive ratio of an online algorithm is at most $\mathcal{R}$, then we say that it is $\mathcal{R}$-competitive. If an algorithm has an unbounded competitive ratio, we say that it is not competitive. The approximation ratio of a polynomial time offline algorithm is defined similarly to be the infimum $\mathcal{R}$ such that for any input, $\mathcal{A} \leq \mathcal{R} \cdot \mathrm{OPT}$. If the approximation ratio of a polynomial time offline algorithm is at most $\mathcal{R}$, then we say that it is an $\mathcal{R}$-approximation.

In [17], Pemmaraju, Raman and Varadarajan designed a 2-approximation algorithm for the max coloring problem on interval graphs. Further, they showed that the First-Fit algorithm, which colors a node using the first available color (in an order in which the colors are given), when the intervals are considered in a monotone non-increasing order of their weights, is a 10-approximation algorithm for the max coloring problem on interval graphs. In [15], Pemmaraju, Raman and Varadarajan designed an $O(\log n)$-approximation algorithm for the (offline) max coloring of chordal graphs. They also analyzed empirically several heuristics. In [16], Pemmaraju and Raman presented a 4-approximation algorithm for the (offline) max coloring of perfect graphs. Since every chordal graph is also a perfect graph, this result improves the earlier $O(\log n)$-approximation algorithm of [15] for chordal graphs. We recall that a perfect graph can be colored using ω colors, where ω is the size of the largest clique in the graph. Note that ω is a clear lower bound on the chromatic number of the graph. An algorithm that finds such a coloring is implied using the ellipsoid algorithm [6] (see also chapter 67 in [19]).

Coloring interval graphs has been intensively studied, Kierstead and Trotter [11] constructed an online algorithm which uses at most $3\omega - 2$ colors where ω is the maximum clique size of the interval graph. They also presented a matching lower bound of $3\omega - 2$ on the number of colors in a coloring of an arbitrary online algorithm. Note that the chromatic number of interval graphs equals to the size of a maximum clique, which is equivalent in the case of interval graphs to the largest number of intervals that intersect any point (see [8, 5]). This means that the optimal offline algorithm can color every interval graph with ω colors. This can be actually done by applying First-Fit to the intervals sorted by their left end points. Therefore, a 1-competitive algorithm exists for this problem in the Time Model. Many papers studied the performance of First-Fit for this problem in the List Model [9, 10, 17, 3]. The last paper shows that the performance of First-Fit is strictly worse than the one of the algorithm of [11].

Interval coloring received much attention recently. In [17], a simple reduction from offline max interval coloring to online interval coloring was shown. The upper bounds in this paper were shown by exploiting the algorithm of [11] (which becomes a 2-approximation instead of the 3-competitive algorithm, since a part of the computation can be done offline), and First-Fit (this paper first improved the known bound on First-Fit and then used it). The reduction simply applies the online algorithm to the set of intervals, sorted by non-increasing order of weight. Adamy and Erlebach [1] introduced the interval coloring with bandwidth problem. In this problem each interval has a bandwidth requirement in $(0, 1]$. The intervals are to be colored

so that at each point, the sum of bandwidths of intervals colored by a certain color which intersect this point, does not exceed 1. This problem was studied also in [2, 4].

Our results: We first present the positive results of this paper. That is, we present a randomized online algorithm that uses as a sub-routine an online node coloring algorithm. This sub-routine is applied to color graphs that are induced subgraphs of the original graph. We then show how to choose the parameters of our algorithm to obtain a deterministic online algorithm though with an inferior competitive ratio. Note that though we reduce the max coloring problem of interval graphs to an interval coloring problem, which is also done in [17], our reduction does not require pre-sorting of the intervals, and therefore our algorithms for interval graphs are online. Using known results for online minimum coloring of interval graphs we obtain the following results. For the List Model we design a deterministic 12-competitive algorithm, a randomized $3e$-competitive algorithm, and prove a lower bound of 4 on the deterministic or randomized competitive ratio. For the Time Model, we use simplified versions of the algorithm and lower bound of the List Model, to achieve a deterministic 4-competitive algorithm, a randomized e-competitive algorithm, a lower bound of $\phi \approx 1.618$ on the deterministic competitive ratio, and a lower bound of $\frac{4}{3}$ on the randomized competitive ratio. The lower bound holds even for unit intervals. For unit intervals and the List Model, we obtain a deterministic 8-competitive algorithm, a randomized $2e$-competitive algorithm and improved lower bounds of 2 and $\frac{11}{6} \approx 1.8333$ on the deterministic and randomized competitive ratios, respectively. Our upper bounds for online algorithms are based on using a general reduction which we introduce in this paper, that allows to convert an r-competitive algorithm for standard node coloring into a $4r$-competitive ($e \cdot r$-competitive) deterministic (randomized) algorithm for max coloring. Finally, we use our randomized algorithm with a derandomization procedure to obtain an offline (deterministic) e-approximation algorithm for max coloring of perfect graphs or any other hereditary graph classes for which the node coloring problem can be solved in polynomial time. We present the algorithms in Section 2, and the lower bounds in Section 3.

2 Algorithms

Before we define our algorithms, we would like to discuss the performance of First-Fit, which is clearly a natural algorithm for coloring. As shown in a sequence of papers [9, 10, 17], applying First-Fit to interval graphs for the standard node coloring problem results in a constant competitive algorithm, though First-Fit is worse than the algorithm of Kierstead and Trotter [11, 3]. However, we can show that First-Fit is not competitive for the max coloring problem.

Proposition 1 *First-Fit is not competitive for max coloring of interval graphs even in the Time Model and the case of unit intervals.*

Proof. Let M be a large constant fixed later. We introduce the input in blocks, where all intervals are of length 2. Block i ($i \geq 0$) consists of i copies of the interval $[4i, 4i+2]$, with weight 1 each, and one interval $[4i+1, 4i+3]$ of weight M. Clearly, the first i intervals of a block are colored using colors $1, \ldots, i$, since they arrive first, and they do not overlap with any previously presented intervals. The next interval which has larger weight is colored with color $i+1$. Denote by j the number of blocks in the input. Then, the cost of the algorithm is at least $M \cdot j$. An optimal offline algorithm would use one color for all intervals having the larger weight, and $j-1$ colors for all other intervals (note that the unit weight intervals can be colored using $j-1$ colors). This results in the cost $M+j-1$. Taking $M = j^2$ we get a competitive ratio of $\frac{j^3}{j^2+j-1}$. When j grows to infinity, this competitive ratio becomes arbitrarily large. ■

We design a framework for converting a deterministic C-competitive algorithm for online coloring of a given class of graphs into a randomized $e \cdot C$-competitive algorithm for max coloring on the same class of graphs. Our framework applies to hereditary classes of graphs (i.e., if a graph belongs to this class, then

every induced subgraph belongs to this class). We apply the scheme using only deterministic algorithms for coloring. This results in deterministic algorithms, using a deterministic reduction scheme, and in randomized algorithms, using a randomized reduction scheme. Clearly, the randomized scheme can be used for converting a randomized algorithm to a randomized one.

Our algorithm has a positive integer parameter k and another real parameter $\alpha > 1$. Our algorithm chooses an integer value $0 \leq \ell < k$ uniformly at random. Upon arrival of a new node we round down its weight as follows. We find the largest integer value t such that $\alpha^{kt+\ell}$ is no larger than the weight of the new node. The rounded weight of the node becomes $\alpha^{kt+\ell}$. In what follows, we always refer to the original weight by the term *weight*, and to the rounded weight by *rounded weight*.

Recall that OPT denotes the total weight (i.e., cost) of an optimal offline solution OPT for the original sequence (where the weights are not rounded). For a given color, the *weight of this color* is defined to be the largest weight of any node which is colored by OPT with this color. That is, the weight of the color is the exact cost which OPT is charged for this color. Consider the subset of colors, used by OPT, having a weight in the interval $\left[\alpha^{i-1}, \alpha^{i}\right)$. Let OPT_i denote the number of the such colors.

Denote by p the largest integer, such that OPT uses at least one color of weight in the interval $\left[\alpha^{k(p-1)}, \alpha^{kp}\right)$. Note that p is unknown to the algorithm and is used only for the analysis.

Lemma 2 OPT *satisfies* $\mathrm{OPT} \geq \sum_{i=-\infty}^{kp} \alpha^{i-1} \cdot \mathrm{OPT}_i$.

Proof. The maximum weight of each of the OPT_i colors of weight in the interval $\left[\alpha^{i-1}, \alpha^{i}\right)$ is at least α^{i-1}. ■

The input is partitioned online into subsequences (also called classes), such that each class is colored independently, using its own set of colors. The subsequence S_i for an integer value of i, contains all nodes whose weight is in the interval $\left[\alpha^{\ell+(i-1)k}, \alpha^{\ell+ik}\right)$, that is, nodes of a rounded weight of $\alpha^{\ell+(i-1)k}$.

Once we are coloring such a class S_i, the rounded weights of all intervals are identical, so weights are not taken into account and the problem is reduced to the classical online node coloring problem. We use a C-competitive algorithm to color such a class.

Lemma 3 *The number of colors that are used to color S_i is at most $C \cdot \sum_{j=\ell+(i-1)k+1}^{kp} \mathrm{OPT}_j$, for all i.*

Proof. OPT can use the colors of weight at least $\alpha^{\ell+(i-1)k}$ to color the nodes of S_i. Therefore, there are at most $\sum_{j=\ell+(i-1)k+1}^{kp} \mathrm{OPT}_j$ colors that are used by OPT to color S_i. Since we use a C-competitive algorithm to color S_i, the claim follows. ■

It remains to analyze the resulting (randomized) algorithm.

Lemma 4 *Assuming the existence of a C-competitive algorithm for online coloring, the randomized online algorithm has a competitive ratio of at most $\frac{C \cdot \alpha^{k+1}}{k(\alpha-1)}$.*

Proof. Since each color, that our algorithm uses to color S_i, has a weight of at most $\alpha^{\ell+ik}$, by Lemma 3, we conclude that for a given value of ℓ the cost of the solution returned by the algorithm is at most $C \cdot \sum_{i=-\infty}^{p} \alpha^{\ell+ik} \sum_{j=\ell+(i-1)k+1}^{kp} \mathrm{OPT}_j$. We now consider the expected cost of the returned solution (the expectation is over the randomized value of ℓ). Since ℓ is drawn uniformly at random from the set $\{0, 1, \ldots, k-1\}$, the expected cost is at most the following:

$$\frac{\sum_{\ell=0}^{k-1} \left(C \cdot \sum_{i=-\infty}^{p} \alpha^{\ell+ik} \sum_{j=\ell+(i-1)k+1}^{kp} \mathrm{OPT}_j \right)}{k} = \frac{C \sum_{j=-\infty}^{kp} \mathrm{OPT}_j \cdot \sum_{t=-\infty}^{j+k-1} \alpha^t}{k}$$

$$= \frac{C \sum_{j=-\infty}^{kp} \mathrm{OPT}_j \cdot \alpha^{j+k-1} \cdot \sum_{t=0}^{\infty} \left(\frac{1}{\alpha}\right)^t}{k} = \frac{C \sum_{j=-\infty}^{kp} \mathrm{OPT}_j \cdot \alpha^{j+k} \cdot \frac{1}{\alpha-1}}{k}$$

where the first equation holds by changing the order of summation, and the second equation holds by taking α^{j+k-1} outside the summation. Recall that by Lemma 2, $\mathrm{OPT} \geq \sum_{j=-\infty}^{kp} \alpha^{j-1} \cdot \mathrm{OPT}_j$. We next note that the coefficient of OPT_j in the lower bound of OPT times $\frac{C \cdot \alpha^{k+1}}{k(\alpha-1)}$ is the coefficient of OPT_j in the upper bound on the expected cost of the solution returned by the algorithm, and thus the claim follows. ■

Theorem 5 *Assuming the existence of a C-competitive deterministic algorithm for online coloring in one of the models, and a given hereditary graph class, there is a (randomized) online algorithm for max coloring (in the same model) with competitive ratio of at most $e \cdot C$ for the same graph class.*

Proof. By setting $\alpha = 1 + \frac{1}{k}$ and k being a large integer number, we can set $\ell = u \cdot k$ where u is uniformly random real number in the interval $[0, 1]$, and in this case $\alpha^{\ell} \approx e^{u}$ and $\alpha^{k} \approx e$. Then, by Lemma 4, the claim follows. ■

Theorem 6 *Assuming the existence of a C-competitive algorithm for online coloring in one of the models, and a given hereditary graph class, there is a deterministic online algorithm for max coloring (in the same model) with competitive ratio at most $4 \cdot C$ for the same graph class.*

Proof. By Lemma 4, and setting $\alpha = 2$ and $k = 1$. We note that for $k = 1$ our algorithm is deterministic as ℓ has a unique possible value of 0. ■

For max coloring of interval graphs we can use the following results: For the List Model, we use the 3-competitive algorithm of Kierstead and Trotter [11]. For the List Model with unit intervals, we use the 2-competitive algorithm of Epstein and Levy [4]. For the Time Model, we color each class optimally using First-Fit [8]. Therefore, we establish the following:

Corollary 7 *For online max coloring of interval graphs there is a randomized algorithm whose competitive ratio is $3e$ in the List Model, $2e$ in the List Model with unit intervals and e in the Time Model.*
For online max coloring of interval graphs there is a deterministic algorithm whose competitive ratio is 12 in the List Model, 8 in the List Model with unit intervals, and 4 in the Time Model.

We next note that for an offline algorithm, we can use a derandomization procedure to transform the (online) randomized algorithm into a deterministic approximation algorithm without increasing the approximation ratio. To obtain the derandomization note that for each node v, v belongs to at most two adjacent classes S_i and S_{i+1} for the different values of ℓ. Therefore, there are at most n *threshold values* $\mathcal{S}$ that can be found in advance. In such a threshold value τ there exists at least one node v such that v belongs to different classes for $\ell = \tau$ and $\ell = \tau + 1$. Recall that in the randomized algorithm we choose $\alpha = 1 + \frac{1}{k}$ and k is a huge integer number. Using these threshold values, we have to calculate only n solutions (the ones that correspond to the threshold values $\mathcal{S}$), and pick the best solution. Therefore, we establish the following theorem.

Theorem 8 *Given an hereditary class of graphs that has a ρ-approximation algorithm for the (offline) minimum node coloring problem, there is a deterministic $(e \cdot \rho)$-approximation algorithm for the offline max coloring problem of the same graph class.*

Note that for perfect graphs (which are known to be hereditary class of graphs) there exists such an optimal algorithm for the minimum node coloring (see [6]). Therefore, we obtain an e-approximation algorithm for perfect graphs improving the 4-approximation algorithm of [16].

Corollary 9 *There is a deterministic e-approximation algorithm for (offline) max coloring of perfect graphs.*

It was brought to our attention that in parallel to our work, Raman had developed the following results for hereditary graph classes in his Ph.D. thesis [18]. Let C be the known approximation ratio of a deterministic algorithm for graph coloring for a given graph class. In Theorem 3.5.3, a deterministic $4C$-approximation algorithm is given. An improved deterministic $3C$-approximation algorithm is given in Theorem 3.5.14. In addition, a randomized $e \cdot C$-approximation is given in Theorem 3.5.5.

3 Lower bounds

In this section we provide lower bounds on the competitive ratio of online algorithms for max coloring of interval graphs. We prove deterministic and randomized lower bounds for three models. The first lower bound is for the Time Model, and only unit intervals are used in the construction. For the List Model, we prove two separate lower bounds for unit intervals, and for intervals of arbitrary length.

All the lower bounds that we present have the same flavor. The idea is to present blocks of intervals, where in such a block, some of the intervals may overlap. The concepts used below are a *base block*, which is the first part of the block, and a notion of an *extended block*, which is a base block, together with an extension. The notion of *a block* can refer to either a base block or an extended block.

Blocks are very different in the three constructions, but in all cases, there is no overlap between intervals of different blocks. In the Time Model, all the intervals of one block clearly must arrive sorted by their left endpoint, while in the List Model, they arrive in an order specified in the construction. A block consists of two parts, which are sets of intervals presented to the algorithm, where the second part is optional and is called *the extension*. The first part is called a base block, and it has an index i which is related to the largest cardinality clique in it. A base block of index i is called a base i block. In the last proof, we use an additional parameter k, so that the maximum cardinality clique is a function of i and k. In this case, a base block with index i and parameter k is called a base (k, i) block. The entire construction (that is, the two parts) is called an extended block. If only an index i is used, then an extended block of index i is called an extended i block. If both a parameter k and an index i are used, then it is called an extended (k, i) block. Thus, an extended i block is simply a base i block followed by a suitable extension, and an extended (k, i) block is a base (k, i) block, followed by a suitable extension for the parameter k and the index i.

An input is constructed in the following way. After the first base block is built, the cost of the algorithm (or its expected cost, in the case of randomized algorithms) is checked. If it is sufficiently large, compared to the optimal cost for the instance, then the input is stopped. Otherwise, the extension is added to the base block, making it an extended block, and a new disjoint base block is presented. Weights are defined so that all the intervals of one base block have a common weight of α which is smaller than the common weight (of 1) of intervals in the extension (so there are only two weights in each extended block). These two weights of $\alpha, 1$ are the same in all blocks of one lower bound construction, and therefore each lower bound construction involves only two weights. The numerical value of α ($0 < \alpha < 1$) is different in the different proofs.

Unless the input is stopped after some base block, the construction continues in this way until we have presented sufficiently many (at least M, for a large integer M) extended blocks. That is, in this last case, the construction stops after the M-th extended block is completed. The details of the different constructions vary, and will be described in this section. We start with a lower bound for the Time Model, which has a simple structure of blocks. The blocks of the two other lower bounds have more complicated structures.

Theorem 10 *The competitive ratio of any deterministic online max coloring algorithm of interval graphs in the Time Model is at least $\phi \approx 1.618$, which holds even if the input is restricted to unit intervals. For randomized algorithms, the competitive ratio is at least $\frac{4}{3}$, which holds even if the input is restricted to unit intervals.*

Proof. As explained above, we need to define base blocks and extended blocks. In addition, we need

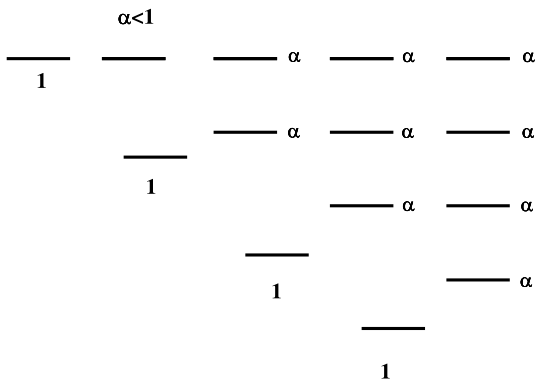

Figure 1: An example of the construction in Theorem 10; an instance with four extended blocks (an extended i block for $i = 1, 2, 3, 4$) and one additional base block (a base 5 block).

to define a stopping condition, under which a base block is not extended. Let N be a large integer. The sequence contains at most N blocks. Let $0 < \alpha < 1$ be a parameter (fixed to be $\alpha = \frac{\sqrt{5}-1}{2} = \phi - 1 \approx 0.618$ in the deterministic case, and $\alpha = \frac{1}{2}$ in the randomized case).

Definition 11 *A base i block is a clique (for some $i \geq 1$) consists of $i-1$ identical intervals. These intervals are copies of the interval $[4i, 4i+2]$, and have the weight α. The intervals of base blocks are called regular intervals.*

Definition 12 *An extended i block consists of a base i block and one additional interval of weight 1 (the expensive interval) which is adjacent to all the intervals of base i block, and it is located slightly shifted to the right (by half the length of an interval) from the intervals of the base block. That is, the extension is a single interval located at $[4i+1, 4i+3]$.*

Note that a base 1 block is an empty set of intervals. An extended i block consists of i intervals, while a base i block has only $i-1$ intervals. For each value of i, we first present the regular intervals, since a base i block consists of exactly those intervals. Afterwards we may present the expensive interval of the extended i block (in case the stopping condition is not met), since that interval is the extension. The first situation to be considered is the case where the sequence is either processed till the end, i.e., until the extended N block is presented. The second situation is the case where for some number $1 \leq i \leq N-1$ there are extended j blocks for $j = 1, 2, \ldots, i$ and in addition, there is a base $i+1$ block. See Figure 1 for an example of an instance. The input cannot be stopped after a base 1 block, since in this case the input is empty.

Note that the regular intervals of a block are located to the left of the expensive interval. Therefore, presenting the base block first and the extension later does not contradict the Time Model. By the construction, intervals of different blocks do not intersect. Clearly, all intervals in one block must receive distinct colors, but any pair of intervals from different blocks may receive the same color.

Note that we defined a set of instances, such that a deterministic online algorithm needs to be competitive for each instance of this set, and a randomized algorithm needs to be competitive against a probability measure over this set of instances. To complete the definition of the instance, we still need to define the stopping condition which will depend on the behavior of the online algorithm which we consider (in the randomized case, we make sure that the stopping condition does not depend on the realization of the random bits, but only on the prior probability of each event). We postpone this until we compute the cost of an optimal solution in each possible input sequence (defined by the above rules).

We now compute the optimal offline cost. If the input sequence consists of a number of extended blocks and no base block (which is not a part of an extended block), then by our construction, the sequence ends with an extended block only after N extended blocks were presented (otherwise, if it is stopped, then it is stopped immediately after a base block was presented). The optimal solution colors all expensive intervals from all extended blocks using one color, and at most one regular interval per base block with each one of $N-1$ additional colors. This gives a total cost of $1+\alpha(N-1)$. If there are $i \geq 1$ extended blocks, and one additional base block, then there are at most i intervals in each block. Therefore OPT needs only i colors, where one of these colors is used for all expensive intervals and in the base $i+1$ block it uses this color for one regular interval. The other $i-1$ colors that OPT uses are used for regular intervals (at most one such interval per base block). Therefore, in this case $\mathrm{OPT} = 1+\alpha(i-1)$.

Next, we consider the behavior of a fixed online algorithm. In the *deterministic* case, we make sure that the algorithm uses exactly i colors immediately after i extended blocks have arrived (if they indeed arrive). Note that the regular intervals in each block arrive first. If the algorithm uses at least one *new color* (which has not been used for coloring intervals of the first $i-1$ extended blocks) to color a regular interval of the base i block, then we stop the sequence after the intervals of base i block. That is, the stopping condition for stopping immediately after the base i block is that the (deterministic) online algorithm, which we consider, has used a new color, which has not been used in the previous extended blocks, for coloring a regular interval of the base i block. Otherwise, that is, if the stopping condition has not been met, the online algorithm is forced to use exactly one new color for coloring the expensive interval of extended i block, and therefore it uses exactly i colors.

We now compute the cost of the online algorithm in each case. Consider first the case in which N extended blocks arrive. This means that the algorithm used a new color for each expensive interval. Therefore, the total cost of the online algorithm in this case is exactly N. Now consider the other case, and let $i+1$ be the index of the base block which is not extended to an extended $i+1$ block, that is, the algorithm has used a new color to color a regular interval of the base $i+1$ block and so the stopping condition is met. The algorithm used i distinct colors for the expensive intervals of the extended j blocks for $j = 1, \ldots, i$. It also uses a new additional color for one regular interval of base $i+1$ block. Therefore, the cost of the solution returned by the algorithm is $i+\alpha$.

To complete the proof for deterministic algorithms, we compute the competitive ratio in each case. Consider the competitive ratio if there are N extended blocks. This ratio is $\frac{N}{1+\alpha N-\alpha}$. For a sufficiently large value of N, the ratio tends to $\frac{1}{\alpha} = \frac{\sqrt{5}+1}{2} = \phi \approx 1.618$. If the sequence stopped immediately after base $i+1$ block, the ratio is $\frac{i+\alpha}{1+i\alpha-\alpha} = \phi$ (for any value of i).

Next, we extend the proof for *randomized* algorithms. Let $0 < p < 1$ be a threshold probability (later fixed to be $p = \frac{2}{3}$). The construction is the same as above, however the stopping condition which we defined above is meaningful only for deterministic algorithms and not for randomized algorithms (since we do not know if the algorithm has actually used a new color to color the regular intervals of the base block). Therefore, we need to define a new stopping condition for the randomized case.

The sequence is stopped after the base $i+1$ block, if the probability that at least $i+1$ colors were used so far by the algorithm, is at least p, and otherwise the sequence continues. Note that for a fixed randomized online algorithm, we can compute these probabilities without the knowledge of the realization of the random bits operations of the algorithm. Hence, our stopping condition is allowed to depend on these probabilities. Clearly, the optimal cost does not change. We compute the cost of the sequence for a fixed randomized algorithm.

Let p_i be the probability that at least $i+1$ colors are used by the algorithm for the instance up to (and including) the base $i+1$ block. First consider the expected cost of the first two blocks. If the sequence is stopped at the earliest possible time, that is, if the second block is a base block, then the expected cost is at least $1+p_1 \cdot \alpha$. If the second block is an extended block, then the expected cost incurred by the algorithm so

far is at least $1 + p_1 \cdot \alpha + (1 - p_1)$. Since the extended block needs at least two colors, and if the interval of the base block has the same color as the block of the first extended block, then the cost of the second color of the second extended block is 1.

Similarly, color $i + 1$ is charged a cost of α with probability p_i and of 1 with probability $1 - p_i$. In the last case we know that this color is used the first time for an interval of weight 1. We get that if the sequence is stopped right after the base $i + 1$ block, the expected cost so far is at least $1 + \alpha \sum_{j=1}^{i} p_j + \sum_{j=1}^{i-1}(1 - p_j) = i + (\alpha - 1) \sum_{j=1}^{i-1} p_j + \alpha p_i \geq i + (\alpha - 1)(i - 1)p + \alpha p$, since $\alpha < 1$, $p_j < p$ for $j < i$ and $p_i \geq p$.

On the other hand, the cost after the extended N block is at least $1 + \alpha \sum_{j=1}^{N} p_j + \sum_{j=1}^{N}(1 - p_j) = N + 1 + (\alpha - 1) \sum_{j=1}^{N} p_j \geq N + 1 + (\alpha - 1)Np$.

We let $\alpha = \frac{1}{2}$ and $p = \frac{2}{3}$. We get the ratio of at least $\frac{N+1-(1-\alpha)Np}{1+\alpha(N-1)}$ in the case where N extended blocks are built. In this case the competitive ratio tends to $\frac{1-p+p\alpha}{\alpha} = \frac{4}{3}$ for large enough N. Otherwise, that is, if the sequence stops before we reach the extended N block, then the ratio is at least $\frac{i-(1-\alpha)(i-1)p+\alpha p}{1+\alpha(i-1)} = \frac{\frac{2i}{3}+\frac{2}{3}}{\frac{i}{2}+\frac{1}{2}} = \frac{4}{3}$. This completes the proof for the randomized case. ■

In the List Model, the previous lower bounds can be improved. This is done using blocks where intervals do not necessarily arrive from left to right. We first consider the case of unit intervals. This time the construction of blocks is similar to the $\frac{3}{2}$ lower bound of [4] for online coloring of unit interval graphs. Blocks are no longer simple cliques, and the construction of a base block depends on the behavior of the online algorithm on the previous intervals of this base block. However, blocks still have a relatively simple final structure.

Theorem 13 *The competitive ratio of any deterministic online max coloring algorithm of unit interval graphs in the List Model is at least* 2. *For randomized algorithms, the competitive ratio is at least* $\frac{11}{6} \approx 1.8333$.

Proof. We start with definitions of blocks, let $0 < \alpha < 1$ and consider the deterministic case first.

Definition 14 *A base i block, for $i \geq 1$, is a set of intervals of weight α, contained in the range $[4i, 4i + 3)$ (so there is no overlap between intervals of different base blocks). Each interval has a weight of α. The set of intervals has the following properties.*

- *The largest clique cardinality is $2i$.*
- *The online algorithm uses exactly $3i$ colors for the base block.*
- *There exists a range of length 1, $[x, x + 1] \subseteq [4i, 4i + 3)$, which we call the central range of the base i block, which has the following properties.*
  - *In the set of intervals having an overlap with $[x, x + 1]$, the largest clique size is $2i$.*
  - *The number of colors using by the algorithm for this last set of intervals is $3i$, that is, every color used by the online algorithm for this block is used for some interval overlapping with $[x, x + 1]$.*

Definition 15 *For $i \geq 1$, an extended i block is a base i block with the central range $[x, x+1]$ concatenated with two requests of weight 1 for $[x, x + 1]$. The online algorithm is therefore forced to use at least $3i + 2$ colors to color this block. Out of those $3i + 2$ colors, at least two must have a weight of 1.*

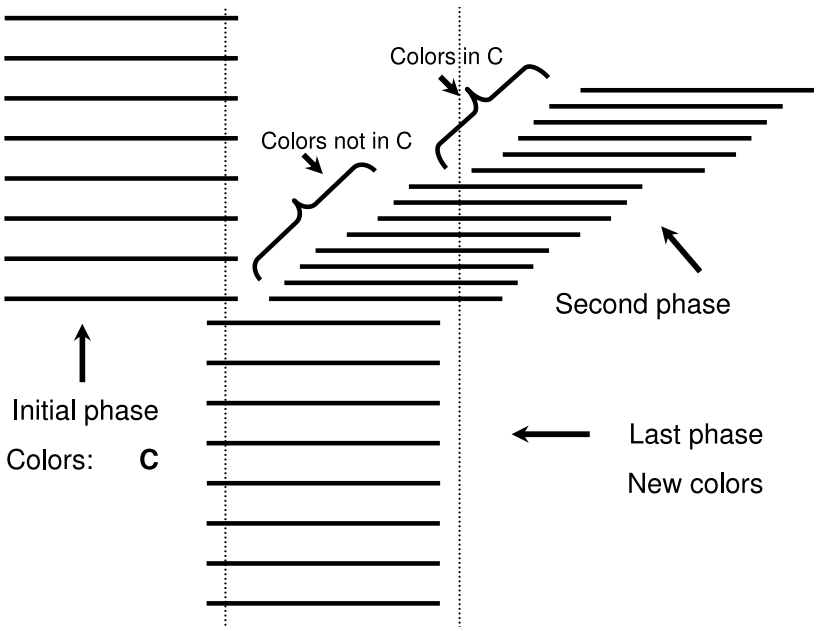

Figure 2: A single base i block of the construction in Theorem 13, where $i = 8$.

An extended 0 *block is similar to a base* 1 *block. It is contained in the range* $[0, 3)$*. The largest clique size is* 2*, while the online algorithm is forced to use three colors. However, the weight of every interval is* 1 *(rather than* α*, which is the weight of intervals in the base* 1 *block).*

The construction of such a base i block is similar to the lower bound of $\frac{3}{2}$, shown in [4] for online coloring of unit intervals and is described next. The construction is partitioned into three phases. In the initial phase we present i identical requests for an interval $[4i, 4i + 1]$. The online algorithm has to color these intervals with exactly i colors. We denote those colors by $c_1, \ldots, c_i$ and the set of those colors by $C = \{c_1, \ldots, c_i\}$.

In the second phase, we present at most $2i$ intersecting intervals, which do not have any overlap with the intervals of the first phase. Since the presented intervals are all intersecting, each one must be colored using a different color. The intervals of the second phase are given one by one. An invariant is kept, that all intervals colored using a color in C are slightly shifted to the right with respect to any interval that is colored by a color which is not in C. That is, the intervals of the second phase satisfy the condition that all left endpoints of intervals colored with a color not in C are located strictly to the left of the left endpoints of intervals which are colored with a color in C. The intervals of the second phase are presented until exactly i of them are colored by colors that are not in C. Since $|C| = i$, any set of $2i$ intersecting intervals must be colored using at least i colors which are not in C. If already after some number $i + j$ of intervals have been presented in the second phase, where $0 \leq j \leq i - 1$, there are i intervals colored with colors not in C, no additional intervals are presented in this phase. We next show the details of this second phase in the construction.

At any time during the presentation of intervals of the second phase, in order to present an additional interval, we use the following definitions. Let $I_1 = [a, a + 1]$ be the rightmost interval colored by some color $\bar{c} \notin C$ (if such an interval exists) and let $I_2 = [d, d + 1]$ be the leftmost interval colored by some color $c \in C$ (if such an interval exists). These two intervals are selected only among intervals introduced so far in the current phase (intervals presented in the initial phase and other blocks are not taken into account). If there is no interval colored $\bar{c}$, then we say that I_1 is empty. If there is no interval colored c, then we say that

I_2 is empty. Let $\varepsilon = \frac{1}{64i}$. A new interval, I, is presented as follows. If both I_1 and I_2 are non-empty, by the invariant, we assume $a < d$. A new interval is presented with a left endpoint between a and d, so this invariant will hold in the next step as well.

1. If both I_1 and I_2 are empty (this holds exactly once, when we introduce the first interval of the second phase) then $I = [4i + \frac{3}{2}, 4i + \frac{5}{2}]$. In this case, in the next step, exactly one of I_1 and I_2 would be empty. Specifically, if the algorithm colors I using a color in C then I_1 will be empty in the next step, and otherwise I_2 will be empty.

2. If only I_1 is empty, $I = [d - \varepsilon, d + 1 - \varepsilon]$. As a result, if I receives a color in C as well, then I_1 would still be empty in the next step. Otherwise I_1 of the next step would be I, while I_2 would not change. In this last case, in the next step, I_1 is to the left of I_2 as required.

3. If only I_2 is empty, $I = [a + \varepsilon, a + 1 + \varepsilon]$. As a result, if I receives a color not in C, then I_2 would still be empty in the next step. Otherwise I_2 of the next step would be I, while I_1 would not change. In this last case, in the next step, I_1 is to the left of I_2 as required.

4. If none of I_1 and I_2 is empty then, $I = [\frac{d+a}{2}, \frac{d+a}{2} + 1]$. That is, the unit length interval is located halfway between I_1 and I_2. If I receives a color from C, then in the next step I_2 would be I, and I_1 would remain unchanged. Otherwise, I_1 of the next step would be I and I_2 would remain unchanged . In both cases, the new I_1 would be to the left of the new I_2.

The stopping condition is the existence of i intervals of the second phase, not colored using a color from C. Recall that this means that the construction is stopped after introducing at most $2i$ intervals.

The distance between the leftmost left endpoint of any interval of the second phase, and the rightmost left endpoint is smaller than $\frac{1}{32}$, thus all the intervals of the second phase are intersecting. We deduce that these intervals are contained in the range $[4i + \frac{47}{32}, 4i + \frac{81}{32}]$. There is therefore no overlap between the intervals of the second phase and the intervals of the first phase. On the other hand, the left endpoints of all the intervals in the second phase are located within a distance of less than 1 from the right endpoints of the intervals of the initial phase.

Assume now that $[y_i + 1, y_i + 2]$ is the rightmost interval with color $\bar{c} \notin C$ after all intervals from phase 2 were presented. From the construction we have $4i + \frac{47}{32} < y_i + 1 < 4i + \frac{49}{32}$. We next define the last phase of the construction of the base block, where we present i requests for the interval $[y_i, y_i + 1]$. This interval intersects all the intervals with color not in C from the second phase, and no intervals with a color in C from the second phase. However, these last i intervals intersect all the intervals from the initial phase. This concludes the construction of a base i block (see Figure 2). The central range of this base i block is defined to be $[y_i, y_i + 1]$.

In the base i block, all the intervals presented in the last phase have an overlap with exactly $2i$ intervals of the first two phases, all having distinct colors, that is they intersect with intervals of exactly $2i$ different colors. These i colors are all colors of C, which are the colors of intervals of the initial phase, and additional i colors, which are not in C, resulting from the second phase. The intervals of the second phase, which have a color in C, intersect with all intervals of the second phase, but not with intervals of the last phase. In this way, an algorithm is forced to use $3i$ colors, while the largest clique of the base i block has a cardinality of $2i$. Moreover, it is possible to define $[y_i, y_i + 1]$ to be the central range of the base i block, since it has non-empty overlap with $3i$ intervals, each colored using a different color, as needed. This completes the proof that such a base i block exists, and can be constructed so that all its intervals lie within the range $[4i, 4i + 3)$.

We next define the construction of an *extended i block*. An extended i block contains a base i block (in which all weights of intervals are equal to α) followed by two additional requests for the interval $[y_i, y_i + 1]$ (which is the the central range of this base i block). The two last requests are of weight 1, thus the cardinality of largest clique of an extended i block is $2i + 2$. However, in every clique in the extended i block, all intervals have a weight of α, except for the last two intervals which have a larger weight of 1. This forces

any algorithm to use two additional colors, so the algorithm is clearly forced to use a total of $3i + 2$ colors to color the extended i block, out of which, two colors must have weight 1.

Once again, the sequence either consists of some number, $N + 1$ of extended blocks (an extended i block for $0 \leq i \leq N$), or of some number j ($1 \leq j \leq N$) of extended blocks (an extended i block for $0 \leq i \leq j - 1$), followed by one base j block. Recall that base 0 block is empty, and the extended 0 block consists of three or four intervals, for which the algorithm uses exactly three colors.

The further blocks are introduced one by one, by increasing index. After the base i block is constructed, the extension is possibly added afterwards, after examining the set of colors which was used for this base i block. Throughout the construction, the following invariant is kept. Just before a base i block is presented, the number of colors which the algorithm must be using is at least $3i - 1$ (the only exception is for $i = 1$ where the algorithm was already forced to use exactly three colors). This holds due to the properties of extended blocks. Due to the stopping rules defined below, it will always be the case that out of these colors, there are at least $i + 1$ colors of weight 1, since otherwise, the extension is not constructed and the block remains a base black, and the sequence stops.

For every i, for which an extended $i - 1$ block is constructed, we define a set of colors which we call *basic colors* for the index i. This is a subset of the colors which are used in the preceding extended blocks. We define this set recursively. For $i = 1$, these are the three colors used in the extended 0 block. For $i = 2$, these are the basic colors of $i = 1$, together with the first two colors used in the extended 1 block, which are not basic for $i = 1$. For $i > 2$, these are the basic colors for the index $i - 1$, together with the three first colors which are not basic, which are used in the extended i block. In total, if $i > 1$, there are $3i - 1$ basic colors for i, and there are three basic colors for $i = 1$.

In the extended i block, after the base i block was constructed, if at least three colors which are not basic colors for i were used, we stop the construction (if $i = 1$, we stop the construction if the base 1 block contains two colors which are not basic for $i = 1$). Otherwise the extension is added, and the base $i + 1$ block is built (unless $i = N$, in which case the input stops after the extension). The earliest time that the construction can be stopped is after the extended 0 block and the base 1 block were constructed.

We compute the optimal cost of the sequence up to a base i block for $i > 0$. The largest cardinality of any clique is $2i$, where the base i block and the extended $i - 1$ block both have such cliques. Every constructed block has two intervals of weight 1 and its other intervals are of weight α. The two exceptions are the extended 0 block, and the base i block. Clearly, the extended 0 block can be colored using two colors of weight 1. Therefore, to color the sequence, two colors of weight 1 and $2i - 2$ colors of weight α are sufficient. This gives a cost of $2 + 2(i - 1)\alpha$. If the sequence terminates at phase N with an extended N block, then the cost is the same as if a base $N + 1$ block were presented, i.e., it is $2 + 2N\alpha$.

After the extended 0 block is presented, the algorithm uses three colors of weight 1. If the sequence terminates after the base 1 block, then two colors which are not basic colors for $i = 1$ were used. That is, there are three basic colors of weight 1 and two colors of weight α. If the sequence terminates after the base i block for $i > 1$, three colors of weight α, which are not basic colors for $i - 1$ were used in this phase. We next calculate the cost of basic colors in previous blocks. For the extended j block with $j > 1$, there are three colors which are not basic for j, which become basic for $j + 1$. If $j = 1$ then there are two such colors. Consider an earlier extended j block, where $1 < j < i$. There are three colors that become basic for the next block. Since the extension was constructed, at most two of those colors used for intervals of weight α, since otherwise the base j block does not become an extended j block. Therefore, at least one color has a weight of 1. In the extended 0 block, three colors of weight 1 were used, and in the extended 1 block, there are two colors which become basic in the next block, out of which, at most one color has a weight of α.

Therefore, if the sequence terminates after the base i block for $i > 1$, the cost of the algorithm is at least $3 + \alpha + 1 + (2\alpha + 1)(i - 2) + 3\alpha = 2 + (2\alpha + 1)i$, whereas the optimal cost is $2 + 2(i - 1)\alpha$. For $i = 1$, if the sequence terminates after the base 1 block, the cost of the algorithm is at least $3 + 2\alpha$, whereas the optimal cost is 2, thus this case does not need to be considered separately.

The cost of the algorithm if the sequence is completed (i.e., all blocks are extended blocks) is $3+\alpha+1+(2\alpha+1)(N-2)+2\alpha+1 = 3-\alpha+(2\alpha+1)N$, whereas the optimal cost is $2+2N\alpha$. We get the ratio $\frac{3+2\alpha+(2\alpha+1)(i-1)}{2+2(i-1)\alpha}$ in the first case, and $\frac{3-\alpha+(2\alpha+1)N}{2+2N\alpha}$ in the second case. We choose a value of α such that $\frac{2\alpha+3}{2} = \frac{1+2\alpha}{2\alpha}$. The value $\alpha = \frac{1}{2} = 0.5$ satisfies this requirement. The ratio in the first case is $\frac{4+2(i-1)}{i+1} = 2$. The ratio in the second case tends to the same value for large enough values of N. The claim for deterministic algorithms follows.

We next provide the extension of this proof for randomized algorithms (which is similar to the proof for deterministic algorithms). As a first step, we explain how to extend the construction of blocks to randomized algorithms (a similar extension can be applied for online unit interval coloring [13]).

In the deterministic proof, we construct one extended i block for each value of i before moving on to the next block. In the randomized case, we introduce multiple non overlapping base i blocks for each value of i (and possibly extend them into extended i blocks). Consider the deterministic construction of a base i block. In this case, we decide on the locations of intervals of its second phase, to the left or to the right of the current interval, based on previous decisions of the algorithm. Specifically, we check whether a new interval is colored using a color which was used in the first phase of this block. However, if we apply this for a randomized algorithm, then we do not know which colors were assigned. Therefore, instead of checking the behavior of the algorithm, we try to guess it using the probability for each outcome. Building a base i block, we need to guess a binary property of $2i$ intervals. The probability to achieve a correct guess for all $2i$ intervals is $\frac{1}{2^{2i}}$. Clearly, for any $\varepsilon > 0$, there exists a number $S_{(i,\varepsilon)}$, such that if we repeat the construction independently $S_{(i,\varepsilon)}$ times, then with high probability of $1-(1-\frac{1}{2^{2i}})^S$ which is at least $1-\varepsilon$, at least one base block uses at least $3i$ colors.

We do not know which base block actually received $3i$ colors. However, if an extension (two additional intervals per block) is to be constructed, then we can add the extension to each one of the base i blocks. If there exists a base i block for which the guess was correct, then is has exactly $3i$ different colors. This base block has at least one new color compared to the set of $3i-1$ basic colors of an extended $i-1$ block for which the guess was correct. If $i=1$, then it can be the case that there are no new colors. We define a threshold $h > 0$ and the decision of whether to extend the base i blocks is performed as follows. Denote the conditional probability that among the colors used for this base i block, there are at least three new colors (which are not basic in the extended $i-1$ blocks) by q_i. The conditional probability is conditioned on the event that we guessed correctly. Recall that in such a case, only the first three such colors would become basic for blocks of the next index, if they are constructed. Denote the conditional probability that among these colors there are exactly two new basic colors by p_i. Therefore the conditional probability for a single new color is $1-p_i-q_i$. For $i=1$, then only the two first new colors would later become basic colors. We let q_1 denote the probability that there are at least two new colors, and p_1 is the probability that there is one new color.

For $i \geq 1$, if $p_i + 2q_i \geq h$, then we stop the sequence after the intervals of the base i blocks, and otherwise, if $i < N$, we extend these base i blocks, and afterwards we construct the base $i+1$ blocks.

Consider first the cost of the sequence up to the a base 1 blocks, in the case that the construction stops after these blocks. Since with high probability, there are three colors in some extended 0 block, we get an additional expected cost of at least $\alpha(p_1+2q_1)$ (in addition to the cost 3 of the basic colors of base block 1). Therefore, with high probability, the cost for the base 1 blocks, if the sequence is stopped is at least $3+\alpha(p_1+2q_1) \geq 3+\alpha h = 3-\alpha+\alpha(1+h)$. Consider the case that the sequence is not stopped after the base 1 blocks. The expected cost for the two new basic colors used in the extended i blocks (with high probability) is $(1-p_1-q_1)2+p_1(1+\alpha)+2\alpha q_1 = 2-p_1(1-\alpha)-2q_1(1-\alpha) \geq 2-h(1-\alpha)$.

If the sequence is not stopped after the base i blocks, for some $i > 1$, then the expected cost for the three new basic colors used in the extended i blocks (with high probability) is $(1-p_i-q_i)(2+\alpha)+p_i(1+2\alpha)+3\alpha q_i = 2+\alpha-(p_i+2q_i)(1-\alpha) \geq 2+\alpha-h(1+\alpha)$.

If base $i+1$ blocks are built, but not extended $i+1$ blocks, this adds (with high probability) a cost of at least $\alpha((1-p_1-q_1)+2p_1+3q_1) = \alpha(1+p_1+2q_1) \geq \alpha(1+h)$ to the cost of the extended i blocks.

Therefore, with high probability, if the sequence is stopped after base i blocks (for some $i \geq 1$), then the expected cost of the algorithm is at least $3-\alpha+\sum_{j=1}^{i-1}(2+\alpha-h(1-\alpha))+\alpha(1+h) = 3-\alpha+(i-1)(2-h)+i\alpha(h+1)$. Otherwise the cost is at least $3-\alpha+\sum_{j=1}^{N}(2+\alpha-h(1-\alpha)) = 3-\alpha+2N+\alpha N-hN+\alpha hN$. The optimal costs do not change. We get a competitive ratio of $\frac{3-\alpha+(i-1)(2-h)+i\alpha(h+1)}{2+2\alpha(i-1)}$ in the first case and of $\frac{3-\alpha+2N+\alpha N-hN+\alpha hN}{2+2N\alpha}$ in the second case. Using the values $\alpha = \frac{1}{2}$ and $h = \frac{4}{3}$ we get the lower bound $\frac{\frac{11}{6}i+\frac{11}{6}}{i+1} = \frac{11}{6}$ in the first case and in the second case, the ratio tends to $\frac{11}{6}$, for large N, which gives the same lower bound in that case as well. ■

The general lower bound for the List Model is based on blocks as well, however the construction of blocks is more complicated than in the previous proofs. The exact set of intervals of which a block consists depends on the behaviour of the online algorithm, and the structure of a block is involved. In fact, the construction of each block is similar to the construction of the lower bound of 3 for online interval coloring, in [11]. We prove the following theorem.

Theorem 16 *The competitive ratio of any deterministic or randomized online max coloring algorithm of interval graphs in the List Model is at least* 4.

Proof. We start with a proof of the deterministic lower bound and later show how to extend it for randomized algorithms. To prove the theorem, we use again base blocks and extended blocks but their structure is now different. Recall that intervals of different blocks cannot intersect.

We first describe the properties of blocks, and afterwards we show how to construct such blocks. Let $0 < \alpha < 1$ be a constant (later chosen to be $\frac{1}{2}$). Once again, all the intervals of every base block have a common weight of α, while the additional intervals of the extensions have a common weight of 1.

Set k to be a large constant and let $i \geq 1$ be an integer.

Definition 17 *A base (k,i) block is a construction of intervals, all contained in the range $(i-1,i)$ (which implies the property that intervals of different blocks cannot have any overlap). The intervals are defined dynamically, based on the behavior of the online algorithm. The construction is such that the online algorithm is forced to use at least $(3k-2)(i-1)$ colors for this base block. A base (k,i) block must satisfy the condition that the largest cardinality clique in it has a size of at most $k(i-1)$.*

Definition 18 *An extended (k,i) block is a construction of intervals, all contained in the range $(i-1,i)$, which consists of a base (k,i) block and an extension. The intervals are defined dynamically, based on the behavior of the online algorithm. The construction is such that the online algorithm is forced to use exactly $(3k-2)i$ colors for this extended block.*

An extended (k,i) block must satisfy the following conditions for an offline coloring of all intervals of the extended block, which uses a minimum number of colors.

- *The number of colors needed in the coloring is at most ki.*
- *There are at most $k(i-1)$ colors of weight α in the coloring.*
- *There are at most k colors of weight 1 in the coloring.*

We call a set of colors, used by the online algorithm to color a set of intervals: "the colors of the set". This set of intervals can be a subset of a block or a block (in which case we use "the colors of the block"). The set of colors does not necessarily include all colors actually used in the block, but it is assigned to sets of intervals, to be able to maintain a lower bound on the number of colors used, and on the cost of the algorithm.

When a (k,i) block is presented to an online algorithm, the numbers k and i are assumed to be known in advance to the algorithm (that is, already at the time of presentation of the first interval of the block). The specific construction is explained later. As usual, the construction of an extended (k,i) block consists of two parts. In the first part we present a base (k,i) block, and in the second optional part, additional unit weight intervals are presented.

The construction of a base (k,i) block for the deterministic case is such that if a set of $(3k-2)i$ colors is used at some point during the construction, then we stop the construction of the lower bound immediately. In the randomized version of the lower bound, the construction is not stopped in such a case, but the charge for the additional colors is performed later, in the sense that it is assumed in the cost calculation that these colors are new in later blocks. It is therefore assumed the algorithm uses at most $(3k-2)i$ colors in an extended (k,i) block.

It is left to describe how to obtain a base (k,i) block and an extended (k,i) block, for any $i \geq 1$ and a sufficiently large value of k. We use a construction which is similar to the lower bound of 3 in [11]. A difference with [11], already used in [4] is the knowledge of k and i that the algorithm is assumed to have.

Given a fixed value of k which is sufficiently large, and $3k-2$ is divisible by 4 (that is, k is even but not divisible by 4). We prove the following claim.

Claim 19 *For a given online algorithm, it is possible to construct an extended (k,i) block and a base (k,i) block, for any $i \geq 1$, according to the properties defined above.*

Proof. The construction of an extended (k,i) block consists of ki phases, where in the first $k(i-1)$ phases, all intervals are of weight α, whereas the last k phases consist of intervals of weight 1. That is, the construction of a base (k,i) block consists of the first $k(i-1)$ phases. To satisfy the conditions on an optimal coloring, we keep the invariant that each phase increases the cardinality of a maximum clique by 1. In addition, we will make sure that the extension is k-colorable. This is sufficient to maintain the conditions regarding optimal colorings, both of the base (k,i) block and the extended (k,i) block. Below, it is stated that the construction may be stopped if the number of colors used by the online algorithm is sufficiently large. Clearly, the optimal coloring can only benefit from stopping the construction prematurely.

After a phase is defined, we *shrink* some parts of the line into single points. By shrinking an interval $[\beta,\gamma]$ into a point, we mean that we perform a mapping m of the real line into itself such that $m(x)=x$ if $x \leq \beta$, $m(x)=x-(\gamma-\beta)$ for $x \geq \gamma$, and $m(x)=\beta$ for $x \in [\beta,\gamma]$. Throughout the construction of an extended (k,i) block we perform multiple shrinking operations, and each time we make sure that all previously presented intervals do not cross the range on which we perform the shrinking operation, that is, they are either contained in it or have no overlap with it at all. Given a point p, that is a result of shrinking an interval $[\beta,\gamma]$, every interval presented in the past which is contained in $[\beta,\gamma]$ is also shrunk into p, and therefore the point p inherits a list of colors that such intervals received. These colors cannot be assigned to any interval that contains the point p. The shrinking is done only for simplification purposes. In practice it means that for a given point p which is the result of shrinking, every future interval either contains this point or not, i.e., it either contains all intervals that were shrunk into this point, or has no overlap with any of them.

Let $U=(3k-2)i>3$ be the number of colors we would like to force the algorithm to use in this extended (k,i) block. If an algorithm ever uses more than U colors in the constructed block, then we stop the construction immediately (no matter whether this happens during the construction of the base (k,i) block

or the construction of the extension. Therefore, we assume that the algorithm is initially given a palette of U colors. As soon as all these colors are used, the proof is complete. Note that this is just one stopping condition. As in previous proofs, there is a condition defined later to decide where the extension should be constructed, or whether the block remains a base block.

Let $S = U^{3ki}$ be an initial number of disjoint intervals which would be presented in the first phase. As required, the presented intervals will be contained in the range $(i-1, i)$. Specifically, in the first phase, we introduce S intervals which are actually the points, $\frac{t}{S+1}$, for $1 \leq t \leq S$. The distance between two consecutive intervals is $\frac{1}{S+1}$. After the first phase was introduced, since we assume that the algorithm is using at most U colors, this means that there exists a set of at least $\frac{S}{U} = U^{3ki-1}$ intervals that share the exact same color c. This color is assigned to be the set of colors used in the intervals of the first phase.

We next define the next phases. In each phase, we will show that the number of intervals which can be used for the next phase decreases by a factor of at most U^3.The phases are constructed in a way that at the beginning of phase $j \geq 2$ there is a set of at least $U^{3ki-3j+5)}$ points that contain a given common subset of the U colors (which clearly holds after phase 1). These points are called *points of interest*. In addition, there may exist other points containing other subsets of colors. All these points are called *void points*. Recall that points can be the result of shrinking of intervals. We keep track of void points and make sure that future intervals do not have these points as endpoints.

We now define phase j (for $j \geq 2$). At this time, we choose exactly $U^{3ki-3j+5}$ points of interest and partition these points of interest into consecutive sets of four points of interest. Additional points of interest which were not chosen to participate become void points.

We next define the intervals of phase j, increasing the size of the largest cardinality clique (with respect to the number of intervals, i.e., ignoring weights) by exactly one. Given a set of four consecutive points of interest, listed from left to right a_1, a_2, a_3, a_4, let x be a point between a_1 and a_2 which is not a void point. Similarly, let y be a point between a_3 and a_4 which is not a void point. Let z be a point between a_2 and a_3 which is not a void point. We introduce the intervals $I_1 = [a_1, x]$ and $I_2 = [y, a_4]$.

If they both receive the same color by the online algorithm, we introduce the intersecting intervals $I_3 = [x, z]$ and $I_4 = [z, y]$. The interval I_3 intersects with a_2, and with I_1. The second interval I_4 intersects I_3, a_3 and I_2. Therefore, two new colors must be used for I_3 and I_4. In total, three new colors were used.

If I_1, I_2 receive distinct colors by the online algorithm, we introduce the interval $I_5 = [x, y]$. Interval I_5 intersects with I_1, I_2, a_2, a_3, and thus gets a new color. In total, three new colors were used. We shrink every such interval $[a_1, a_4]$ into a single point. Such a shrunk point a_1 receives three new colors that are added to its list of colors, which is initialized by the union of sets of colors of all intervals contained in it.

Note that we do not use more than U colors, and each new shrunk point receives three new colors. There are at most $\frac{U^3}{6}$ options to choose a set of three new colors from the available palette of $|U|$ colors. We can therefore find a set of $\frac{U^{(3ki-3j+5)/4}}{U^3/6} \geq U^{3ki-3j+2}$ points having the same set of used colors. $U^{3ki-3j+2}$ points containing these exact sets of colors become the points of interest of the next phase, and the others become void points of the next phase. Points that are void points of previous phases and are not contained in shrunk intervals remain void points. Note that the points where the new intervals intersect are points with no previous intervals, and therefore the maximum clique size increases by exactly 1. The set of colors of the intervals up to phase j is defined to be the set of phase $j-1$ together with the three new colors of the new points of interest.

After the first $k(i-1)$ phases, the construction of the base (k, i) block is completed, and we start presenting intervals of weight 1 instead of the weight α. The first phase of intervals of larger weight is different from all other phases, as we would like the set of all intervals of weight 1 to be k colorable. Thus, the first such phase we introduce has a clique size of exactly 1. Therefore, we introduce single intervals $[a_1, a_4]$ instead of the construction above, in this phase only. In this phase the algorithm has to use a single new color. If $i = 1$, then the base $(k, 1)$ block is empty. In this case, the points of interest for the first phase

of the construction of the extended $(k,1)$ block are defined to be the points $\frac{t}{S+1}$, for $1 \leq t \leq S$.

In a base (k,i) block, the number of colors of the entire input is at least $3k(i-1)-2 > (3k-2)(i-1)$, since three new colors are assigned to the input in each phase except for the first one. Thus the online algorithm uses at least $3k(i-1)-2 > (3k-2)(i-1)$ colors. These colors have each a weight of at least α. In the extension, if $i=1$, the number of colors (all of weight 1) is at least $3k-2$. Otherwise, if the base (k,i) block is not empty and $i \geq 2$,the number of colors is at least $3k(i-1)-2+3k-2 \geq (3k-2)i$.This completes the proof of the claim. ■

The structure of the lower bound construction is similar to previous proofs, in particular, at most N blocks are used. Recall that in an extended (k,i) block, the online algorithm is forced to use $(3k-2)i$ colors. After the construction of a base (k,i) block, it is possible to count the total number of colors m_i that were used in it, where $m_i \leq (3k-2)i$. Since the total number of colors in extended (k,i) block must be $(3k-2)i$ (since the construction is stopped where this number of colors is reached), we define $n_i = (3k-2)i - m_i$ to be the number of new colors needed to complete extended (k,i) block.

We next describe the condition under which a base (k,i) block is extended to be an extended (k,i) block. When block i is presented, if the algorithm used at least $(3k-2)(i-1)+2k$ colors for coloring the base (k,i) block, then the input sequence terminates. That is, the extension is not constructed. Otherwise, we continue to create the extended (k,i) block and the next additional base block. In particular, if the stopping condition of using at least $(3k-2)i$ colors is met already in the construction of the base (k,i) block, the part of the base (k,i) block which was constructed is considered to be the base (k,i) block which meets the condition of stopping the sequence. If it was decided to add the extension, and the condition of $(3k-2)i$ colors is met already in the construction of the extension, then the block is seen as a complete extended (k,i) block.

We compute the cost of the online algorithm, if it stops at block $i+1$ (i.e. the algorithm stops immediately after base $(k,i+1)$ block). All new colors in this base $(k,i+1)$ block have a cost of α. For evaluating the total cost, we first charge all other colors an amount of α and then assign an additional cost of $1-\alpha$ to colors that have weight 1. In an extended (k,j) block, there is an additional cost of $(1-\alpha)n_j$ for the n_j new colors used for intervals of weight 1. Therefore, the total cost is at least $((3k-2)i+2k)\alpha + (1-\alpha)\sum_{j=1}^{i} n_j$. Since previous blocks are extended blocks, we have $n_j \geq k-1$ for $2 \leq j \leq i$ and $n_1 = 3k-2$. We get a cost of at least $(3k-2)i\alpha + 2k\alpha + (1-\alpha)(i-1)(k-1) + (1-\alpha)(3k-2) = k(3i\alpha + 2\alpha + (1-\alpha)(i-1) + 3(1-\alpha)) - 2i\alpha - (1-\alpha)(i+1)$. The optimal offline cost for this input is $k + (i-1)\alpha k$. It remains to consider the case where there are no base blocks which were not extended to an extended (k,i) block, that is if there are N extended blocks. In this last case, the cost of the algorithm is at least $(3k-2)N\alpha + (1-\alpha)\sum_{j=1}^{N} n_j$. Since all blocks are extended blocks, we have $n_j \geq k-1$ for $2 \leq j \leq N$ and $n_1 = 3k-2$. This gives a cost of at least $(3k-2)N\alpha + (1-\alpha)(N-1)(k-1) + (1-\alpha)(3k-2) = k(3N\alpha + (1-\alpha)(N-1) + 3(1-\alpha)) - 2N\alpha - (1-\alpha)(N+1)$. In this case, OPT pays $k + (N-1)\alpha k$.

Choosing $\alpha = \frac{1}{2}$, in the first case we get a lower bound on the competitive ratio which tends to $\frac{4i+4}{i+1} = 4$ for sufficiently large k. In the second case, we get a lower bound on the competitive ratio which tends to $\frac{4N+2}{N+1}$ for sufficiently large k. Taking N to be large enough, this ratio tends to 4 as well.

In order to adapt the lower bound for randomized algorithms note that the values n_i can be used as thresholds on the expected number of colors rather than their exact number. In this case, there is no way to ensure that the total number of colors in a (base or extended) (k,i) block does not exceed $(3k-2)i$.

It was shown in [12] that the lower bound of 3 on the competitive ratio of online algorithms for interval coloring holds for randomized algorithms as well. They showed it using Yao's lemma [20] which states that a lower bound for the competitive ratio of deterministic algorithms on a fixed distribution on the input is also a lower bound for randomized algorithms. They used a distribution over the possible inputs constructed

above (similarly to the construction in [11]).

Since we would like to use a direct proof, we need to show this property directly (i.e., we now present an equivalent method to obtain the result of [12]). In the process above, each block is constructed exactly once since it is known what the online algorithm does at every step. Instead, we apply the process of constructing a base (k,i) block a large number of times. Distinct copies of a base block are placed in disjoints areas of the real line. Since we do not know which colors were assigned by the algorithm while we build a block, we try to guess the behavior of the algorithm. Clearly, there is a very large number of information bits to guess. If this number is f, then the probability to guess correctly all f decisions is $\frac{1}{2^f}$. However, for any $\varepsilon > 0$, there exists a number $S_{(f,\varepsilon)}$, such that if we repeat the construction $S_{(f,\varepsilon)}$ times, then with high probability (e.g., $1-(1-\frac{1}{2^f})^S_{(f,\varepsilon)}$ which is at least $1-\varepsilon$), at least one base (k,i) block has at least $(3k-2)(i-1)$ colors.

We next concentrate on this base (k,i) block that has $(3k-2)(i-1)$ colors. If the expected maximum of colors in a base (k,i) block colors is smaller than the threshold value m_i, we need to extend the base (k,i) block into extended (k,i) blocks. Since we do not know which base (k,i) block received the largest number of colors, we need to do this for all base (k,i) blocks. In fact, the number $S_{(f,\varepsilon)}$ should be large enough so that there are sufficiently many base (k,i) blocks with $(3k-2)(i-1)$ colors, so that with high probability, at least one of them would be extended into an extended (k,i) block with at least $(3k-2)i$ colors.

The rest of the proof and calculations are the same as in the deterministic case, however the costs of the algorithm mentioned above hold with high probability, and not with probability 1. ■

4 Concluding remarks

We presented a framework for converting a deterministic C-competitive algorithm for online coloring of a given hereditary class of graphs into a deterministic $4C$-competitive algorithm, and a randomized $e \cdot C$-competitive algorithm for max coloring on the same class of graphs. For example, consider bipartite graphs. Lovász, Saks and Trotter [14] showed a deterministic online algorithm which colors such a graph on n nodes (which is 2 colorable) using $O(\log n)$ colors. Note that Gyárfás and Lehel [7] proved a deterministic lower bound of $\Omega(\log n)$ on the online coloring of bipartite graphs (this holds already for trees). This immediately implies a deterministic $O(\log n)$-competitive algorithm for online max coloring of bipartite graphs. Note that the deterministic lower bound of $\Omega(\log n)$ holds for max coloring since node coloring is a special case of max coloring (using a common weight 1 for all nodes). The best offline approximation for max coloring of bipartite graphs has an approximation ratio of $\frac{8}{7}$ [16]. In the last paper it is shown that unless $P = NP$, this is best possible.

References

[1] U. Adamy and T. Erlebach. Online coloring of intervals with bandwidth. In *Proc. of te First International Workshop on Approximation and Online Algorithms (WAOA'03)*, pages 1–12, 2003.

[2] Y. Azar, A. Fiat, M. Levy and N. S. Narayanaswamy. An improved algorithm for online coloring of intervals with bandwidth. *Theoretical Computer Science*, 363(1):18-27, 2006.

[3] M. Chrobak and M. Ślusarek. On some packing problems relating to dynamical storage allocation. *RAIRO Journal on Information Theory and Applications*, 22:487–499, 1988.

[4] L. Epstein and M. Levy. Online interval coloring and variants. In *Proceedings of The 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, pages 602–613, 2005.

[5] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[6] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1993.

[7] A. Gyárfás and J. Lehel. On-line and first-fit colorings of graphs. *Journal of Graph Theory*, 12:217–227, 1988.

[8] T. R. Jensen and B. Toft. *Graph coloring problems*. Wiley, 1995.

[9] H. A. Kierstead. The linearity of first-fit coloring of interval graphs. *SIAM Journal on Discrete Mathematics*, 1(4):526–530, 1988.

[10] H. A. Kierstead and J. Qin. Coloring interval graphs with First-Fit. *SIAM Journal on Discrete Mathematics*, 8:47–57, 1995.

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

[12] S. Leonardi and A. Vitaletti. Randomized lower bounds for online path coloring. In *Proc. of the second International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM'98)*, pages 232–247, 1998.

[13] M. Levy. Private communication, 2005.

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

[15] S. V. Pemmaraju, S. Penumatcha, and R. Raman. Approximating interval coloring and max-coloring in chordal graphs. *ACM Journal of Experimental Algorithms*, 10, article 2.8, 2005.

[16] S. V. Pemmaraju and R. Raman. Approximation algorithms for the max-coloring problem. In *Proceedings of The 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, pages 1064–1075, 2005.

[17] S. V. Pemmaraju, R. Raman, and K. R. Varadarajan. Buffer minimization using max-coloring. In *Proc. of 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, pages 562–571, 2004.

[18] R. Raman. Chromatic scheduling. Ph.D. thesis. Department of Computer Science, The University of Iowa, 2007.

[19] A. Schrijver. *Combinatorial Optimization Polyhedra and Efficiency*. Springer-Verlag, 2003.

[20] A. C. C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. of the 18th Annual Symposium on Foundations of Computer Science (FOCS'77)*, pages 222–227, 1977.