

# Approximating Interval Coloring and Max-Coloring in Chordal Graphs<sup>1</sup>

Sriram V. Pemmaraju<sup>2</sup>

Sriram Penumatcha

Rajiv Raman<sup>2</sup>

January 23, 2004

## Abstract

We consider two coloring problems: interval coloring and max-coloring for chordal graphs. Given a graph  $G = (V, E)$  and positive integral vertex weights  $w : V \rightarrow \mathbf{N}$ , the *interval coloring* problem seeks to find an assignment of a real interval  $I(u)$  to each vertex  $u \in V$  such that two constraints are satisfied: (i) for every vertex  $u \in V$ ,  $|I(u)| = w(u)$  and (ii) for every pair of adjacent vertices  $u$  and  $v$ ,  $I(u) \cap I(v) = \emptyset$ . The goal is to minimize the span  $|\cup_{v \in V} I(v)|$ . The *max-coloring problem* seeks to find a proper vertex coloring of  $G$  whose color classes  $C_1, C_2, \dots, C_k$ , minimize the sum of the weights of the heaviest vertices in the color classes, that is,  $\sum_{i=1}^k \max_{v \in C_i} w(v)$ . Both problems arise in efficient memory allocation for programs. The interval coloring problem models the compile-time memory allocation problem and has a rich history dating back at least to the 1970's. The max-coloring problem models the problem of minimizing the total buffer size needed by a dedicated memory manager. This problem 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. Both problems are NP-complete even for interval graphs, though there are constant-factor approximation algorithms for both problems on interval graphs.

In this paper we consider these problems for *chordal graphs*. These graphs naturally generalize interval graphs and can be defined as the class of graphs that have no induced cycle of length  $> 3$ . There are no known constant-factor approximation algorithms for either interval coloring or for max-coloring on chordal graphs. However, we point out in this paper that there are several simple  $O(\log(n))$ -factor approximation algorithms for both problems. We experimentally evaluate and compare three simple heuristics: first-fit, best-fit, and a heuristic based on partitioning the graph into vertex sets of similar weight. Our experiments show that in general first-fit performs better than the other two heuristics and is typically very close to OPT, deviating from OPT by about 6% in the worst case for both problems. Best-fit provides some competition to first-fit, but the graph partitioning heuristic performs significantly worse than either. Our basic data comes from about 10000 runs of the each of the three heuristics for each of the two problems on randomly generated chordal graphs of various sizes and sparsity.

Our experiments also reveal that for interval coloring, chordal graphs that are somewhat “regular” favour first-fit, whereas those that are somewhat “irregular” favor best-fit. On the other hand, for max-coloring, first-fit outperforms the other algorithms most of the time.

## 1 Introduction

**Interval coloring.** Given a graph  $G = (V, E)$  and positive integral vertex weights  $w : V \rightarrow \mathbf{N}$ , the *interval coloring* problem seeks to find an assignment of an interval  $I(u)$  to each vertex  $u \in V$  such that two constraints are satisfied: (i) for every vertex  $u \in V$ ,  $|I(u)| = w(u)$  and (ii) for every pair of adjacent vertices  $u$  and  $v$ ,  $I(u) \cap I(v) = \emptyset$ . The goal is to minimize the span  $|\cup_v I(v)|$ . The interval coloring

---

<sup>1</sup>All three authors are at the Department of Computer Science, The University of Iowa, Iowa City, IA 52240-1419. E-mail: [sriram, spenumat, rraman]@cs.uiowa.edu

<sup>2</sup>Partially supported by National Science Foundation Grant DMS-0213305

problem has a fairly long history dating back, at least to the 70's. For example, Stockmeyer showed in 1976 that the interval coloring problem is NP-complete even when restricted to interval graphs and vertex weights in  $\{1, 2\}$  (see problem SR2 in Garey and Johnson [5]). The main application of the interval coloring problem is in the *compile-time memory allocation problem*. Fabri [4] made this connection in 1979. In order to reduce the total memory consumption of source-code objects (simple variables, arrays, structures), the compiler can make use of the fact that the memory regions of two objects are allowed to overlap provided that the objects do not “interfere” at run-time. This problem can be abstracted as the interval coloring problem, as follows. The source-code objects correspond to vertices of our graph, run-time interference between pairs of source code objects is represented by edges of the graph, the amount of memory needed for each source-code object is represented by the weight of the corresponding vertex, and the assignment of memory regions to source code objects is represented by the assignment of intervals to vertices of the graph. Minimizing the size of the union of intervals corresponds to minimizing the amount of memory allocation.

If we restrict our attention to straight-line programs, that is, programs without loops or conditional statements, then the compile-time memory allocation problem can be modeled as the interval coloring problem for interval graphs. This is because the interference graph for source-code objects in a straight-line program is an interval graph. As mentioned before, the interval coloring problem is NP-complete for interval graphs and so research has focused on designing approximation algorithms for the problem. Kierstead [10] gave the first constant-factor approximation algorithm – an 80-approximation, that was improved by Kierstead himself to a 6-approximation [11]. This was followed by papers by Gergov [6, 7] who improved the approximation factor to 3. Recently, Buchsbaum et.al. [1] gave a  $(2 + \varepsilon)$ -algorithm for the problem.

**Max-coloring.** Like interval coloring, the *max-coloring problem* takes as input a vertex-weighted graph  $G = (V, E)$  with weight function  $w : V \rightarrow \mathbf{N}$ . The problem requires that we find a proper vertex coloring of  $G$  whose color classes  $C_1, C_2, \dots, C_k$ , minimize the sum of the weights of the heaviest vertices in the color classes, that is,  $\sum_{i=1}^k \max_{v \in C_i} w(v)$ . The max-coloring problem models that problem of minimizing the total buffer size need for memory management in different applications. For example, [8] uses max-coloring to minimize buffer size in digital signal processing applications. In [13], max-coloring models the problem of minimizing buffer size needed by memory managers for wireless protocol stacks like GPRS or 3G. In general, 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 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 at least as large as the size of the memory request. 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. Requests that do not interfere with each other can be assigned the same color/buffer. Thus the problem of minimizing the total size of the buffer pool corresponds to the max-coloring problem.

[13] shows that the max-coloring problem is NP-complete for interval graphs and presents the first constant-factor approximation algorithm for the max-coloring problem for interval graphs. The paper makes a connection between max-coloring and on-line graph coloring and using a known result of Kierstead and Trotter [12] on on-line coloring interval graphs, they obtain a 2-approximation algorithm for interval graphs and a 3-approximation algorithm for circular arc graphs.

**Connections between interval coloring and max-coloring.** Given a coloring of a vertex weighted graph  $G = (V, E)$  with color classes  $C_1, C_2, \dots, C_k$ , we can construct an assignment of intervals to vertices as follows. For each  $i$ ,  $1 \leq i \leq k$ , let  $v_i \in C_i$  be the vertex with maximum weight in  $C_i$ . Let  $H(1) = 0$ , and for each  $i$ ,  $2 \leq i \leq k$ , let  $H(i) = \sum_{j=1}^{i-1} w(v_j)$ . For each vertex  $v \in C_i$ , we set  $I(v) = (H(i), H(i) + w(v))$ . Clearly, no two vertices is distinct color classes have overlapping intervals and therefore this is a valid interval coloring of  $G$ . We say that this is the interval coloring *induced* by

the coloring  $C_1, C_2, \dots, C_k$ . The span of this interval coloring is  $\sum_{i=1}^k w(v_i)$ , which is the same as the weight of the coloring  $C_1, C_2, \dots, C_k$  viewed as a max-coloring. In other words, if there is a max-coloring of weight  $W$  for a vertex weighted graph  $G$ , then there is an interval coloring of  $G$  of the same weight.

However, in [13] we show an instance of a vertex weighted interval graph on  $n$  vertices for which the weight of an optimal max-coloring is  $\Omega(\log n)$  times the weight of the heaviest clique. This translates into an  $\Omega(\log n)$  gap between the weight of an optimal max-coloring and the span of an optimal interval coloring because an optimal interval coloring of an interval graph has span that is within  $O(1)$  of the weight of a heaviest clique.

In general, algorithms for max-coloring can be used for interval coloring with minor modifications to make the interval assignments more “compact.” These connections motivate us to study interval coloring and max-coloring in the same framework.

**Chordal graphs.** For both the interval coloring and max-coloring problems, the assumption that the underlying graph is an interval graph is somewhat restrictive since most programs contain conditional statements and loops. In this paper we consider a natural generalization of interval graphs called chordal graphs. A graph is a *chordal graph* if it has no induced cycles of length 4 or more. Alternately, every cycle of length 4 or more in a chordal graph has a chord.

The approximability of interval coloring and max-coloring on chordal graphs is not very well understood yet. As we point out in this paper, there are several  $O(\log(n))$ -factor approximation algorithms for both problems on chordal graphs, however the existence of constant-factor approximation algorithms for these problems is open.

There are many alternate characterizations of chordal graphs. One that will be useful in this paper is the existence of a perfect elimination ordering of the vertices of any chordal graph. An ordering  $v_n, v_{n-1}, \dots, v_1$  of the vertex set of a graph is said to be a *perfect elimination ordering* if when vertices are deleted in this order, for each  $i$ , the neighbors of vertex  $v_i$  in the remaining graph,  $G[\{v_1, v_2, \dots, v_i\}]$  form a clique. A graph is a chordal graph iff it has a perfect elimination ordering. Tarjan and Yannakakis [15] describe a simple linear-time algorithm called *maximum cardinality search* that can be used to determine if a given graph has a perfect elimination ordering and to construct such an ordering if it exists. Given a perfect elimination ordering of a graph  $G$ , the graph can be colored by considering vertices in reverse perfect elimination order and assigning to each vertex the minimum available color. It is easy to see that this greedy coloring algorithm uses exactly as many colors as the size of the largest clique in the graph and therefore produces an optimal vertex coloring.

Every interval graph is also a chordal graph (but not vice versa). To see this, take an interval representation of an interval graph and order the intervals in left-to-right order of their left endpoints. It is easy to verify that this gives a perfect elimination ordering of the interval graph. Thus chordal graphs generalize interval graphs and one of our motivations in considering chordal graphs is to determine if the constant-factor algorithms for interval coloring and max-coloring interval graphs can be extended to chordal graphs. Another motivation for considering chordal graphs is that the way certain kinds of compilers such as algebraic compilers process source code, the interference graph of source objects ends up being a chordal graph [14]. A final motivation is that others have considered the problem of finding approximation algorithms for interval coloring chordal graphs, but with limited success. For example, [3] shows a 2-approximation algorithm for the interval coloring problem on claw-free chordal graphs, leaving the problem open for chordal graphs in general.

**The rest of the paper.** In this paper, we consider three simple heuristics for the interval coloring and max-coloring problems and experimentally evaluate their performance. These heuristics are:

- **First fit.** Vertices are considered in decreasing order of weight and each vertex is assigned the first available color or interval.
- **Best fit.** Vertices are considered in reverse perfect elimination order and each vertex is assigned the color class or interval it “fits” in best.

- **Graph partitioning.** Vertices are partitioned into groups with similar weight and we use the greedy coloring algorithm to coloring each subgraph with optimal number of colors. The interval assignment induced by this coloring is returned as the solution to the interval coloring problem.

In each case, we take as the solution to the interval coloring problem the interval assignment induced by the constructed coloring. First fit and best-fit are fairly standard heuristics for many resource allocation problems and have been analyzed extensively for problems such as the bin packing problem. Using old results and a few new observations, we point out that the first fit heuristic and the graph partitioning heuristic provide an  $O(\log n)$  approximation guarantee. The best-fit heuristic provides no such guarantee and we provide an example of a vertex weighted interval graph for which the best-fit heuristic returns a solution to the max-coloring problem whose weight is  $\Omega(\sqrt{n})$  times the weight of the optimal solution.

Our experiments show that in general first-fit performs better than the other two heuristics and is typically very close to OPT, deviating from OPT by about 6% in the worst case for both problems. Best-fit provides some competition to first-fit, but the graph partitioning heuristic performs significantly worse than either. Our basic data comes from about 10000 runs of the each of the three heuristics for each of the two problems on randomly generated chordal graphs of various sizes and sparsity.

Our experiments also reveal that best-fit performs better on chordal graphs that are “irregular”, while the performance of first-fit deteriorates slightly. In all other cases, first-fit is the best algorithm. Here, “regularity” refers to the variance in the sizes of maximal cliques – greater this variance, more irregular the graph.

## 2 The Algorithms

In this section we describe three simple algorithms for the interval coloring and max-coloring problems.

### 2.1 Algorithm 1: First-fit in weight order

For the interval coloring problem, we preprocess the vertices and “round up” their weights to the nearest power of 2. Then, for both problems we order the vertices of the graph in non-increasing order of weights. Let  $v_1, v_2, \dots, v_n$  be this ordering. We process vertices in this order and use a “first-fit heuristic” to assign intervals and colors to vertices to solve the interval coloring and max-coloring problem respectively.

The algorithm for interval coloring is as follows. To each vertex we assign a real interval with non-negative endpoints. To vertex  $v_1$ , we assign  $(0, w(v_1))$ . When we get to vertex  $v_i$ ,  $i > 1$ , each vertex  $v_j$ ,  $1 \leq j \leq i - 1$  has been assigned an interval  $I(v_j)$ . Let  $U_i$  be the union of the intervals already assigned to neighbors of  $v_i$ . Then  $(0, \infty) - U_i$  is a non-empty collection of disjoint intervals. Because the weights are powers of 2 and vertices are considered in non-increasing order of weights, every interval in  $(0, \infty) - U_i$  has length at least  $w(v_i)$ . Of these, pick an interval  $I = (a, b)$  with smallest right endpoint and assign the interval  $(a, a + w(v_i))$  to  $v_i$ . This is  $I(v_i)$ .

For a solution to the max-coloring problem, we assume that the colors to be assigned to vertices natural numbers, and assign to each vertex  $v_i$  the smallest color not already assigned to a neighbor of  $v_i$ . We denote the two algorithms described above by FFI (short for first-fit by weight order for interval coloring) and FFM (short for first-fit by weight order for max-coloring) respectively.

We now observe that both algorithms provide an  $O(\log(n))$ -approximation guarantee. The following result is a generalization of the result from [2].

**Theorem 1** *Let  $C$  be a class of graphs and suppose there is a function  $\alpha(n)$  such that the first-fit on-line graph coloring algorithm colors any  $n$ -vertex graph  $G$  in  $C$  with at most  $\alpha(n) \cdot \chi(G)$  colors. Then, for any  $n$ -vertex graph  $G$  in  $C$  the FFI algorithm produces a solution with span at most  $2\alpha(n) \cdot OPT_I(G)$ , where  $OPT_I(G)$  is the optimal span of any feasible assignment of intervals to vertices.*

The following is a generalization of the result from [13].

**Theorem 2** Let  $C$  be a class of graphs and suppose  $\alpha(n)$  is a function such that the first-fit on-line graph coloring algorithm colors any  $n$ -vertex graph  $G$  in  $C$  with at most  $\alpha(n) \cdot \chi(G)$  colors. Then, for any  $n$ -vertex graph  $G$  in  $C$  the FFM algorithm produces a solution with weight at most  $\alpha(n) \cdot OPT_M(G)$ , where  $OPT_M(G)$  is the optimal weight of any proper vertex coloring of  $G$ .

Irani [9] has shown that the first-fit graph coloring algorithm uses at most  $O(\log(n)) \cdot \chi(G)$  colors for any  $n$ -vertex chordal graph  $G$ . This fact together with the above theorems implies that FFI and FFM provide  $O(\log(n))$ -approximation guarantees.

An example that is tight for both algorithms is easy to construct. Let  $T_0, T_1, T_2, \dots$  be a sequence of trees where  $T_0$  is a single vertex and  $T_i, i > 0$ , is constructed from  $T_{i-1}$  as follows. Let  $V(T_{i-1}) = \{u_1, u_2, \dots, u_k\}$ . To construct  $T_i$ , start with  $T_{i-1}$  and add vertices  $\{v_1, v_2, \dots, v_k\}$  and edges  $\{u_i, v_i\}$  for all  $i = 1, 2, \dots, k$ . Thus the leaves of  $T_i$  are  $\{v_1, v_2, \dots, v_k\}$  and every other vertex in  $T_i$  has a neighbor  $v_j$  for some  $j$ . Now consider a tree  $T_n$  in this sequence. Clearly,  $|V(T_n)| = 2^n$ . Assign to each vertex in  $T_n$  a unit weight. To construct an ordering on the vertices of  $T_n$  first delete the leaves of  $T_n$ . This leaves the tree  $T_{n-1}$ . Recursively construct the ordering on vertices of  $T_{n-1}$ , and prepend to this the leaves of  $T_n$  in some order. It is easy to see that first-fit coloring algorithm that considers the vertices of  $T_n$  in this order uses  $n$  colors. As a result, both FFI and FFM have cost  $n$ , whereas  $OPT$  in both cases is 2. See Figure 1 for  $T_0, T_1, T_2$ , and  $T_3$ .

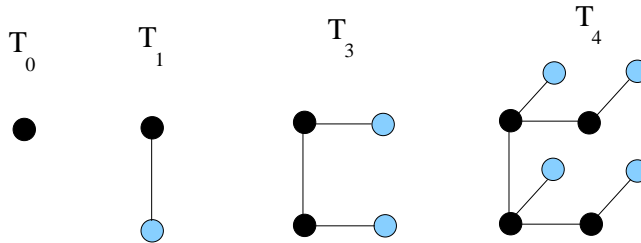


Figure 1: The family of tight examples for FFI and FFM.

## 2.2 Algorithm 2: Best-fit in reverse perfect elimination order

A second pair of algorithms that we experiment with are obtained by considering vertices in reverse perfect elimination order and using a “best-fit” heuristic to assign intervals or colors. Let  $v_1, v_2, \dots, v_n$  be the reverse of a perfect elimination ordering of the vertices of  $G$ . Recall that if vertices are considered in reverse perfect elimination order and colored, using the smallest color at each step, we get an optimal coloring of the given chordal graph. This essentially implies that the example of a tree with unit weights that forced FFI and FFM into worst case behavior will not be an obstacle for this pair of algorithms.

The algorithm for interval coloring is as follows. As before, to each vertex we assign a real interval with non-negative endpoints and to vertex  $v_1$ , we assign  $(0, w(v_1))$ . When we get to vertex  $v_i, i > 1$ , each vertex  $v_j, 1 \leq j \leq i - 1$  has been assigned an interval  $I(v_j)$ . Let  $M = |\cup_{j=1}^{i-1} I(v_j)|$  and let  $U_i$  be the union of the intervals  $I(v_j)$ , where  $1 \leq j \leq i - 1$  and  $v_j$  is a neighbor of  $v_i$ . If  $U_i = (0, M)$ , then  $v_i$  is assigned the interval  $(M, M + w(v_i))$ . Otherwise, if  $U_i \neq (0, M)$ , then  $(0, M) - U_i$  is a non-empty collection of disjoint intervals. However, since the vertices were not processed in weight order, we are no longer guaranteed that there is any interval in  $(0, M) - U_i$  with length at least  $w(v_i)$ . There are two cases.

**Case 1.** If there is an interval in  $(0, M) - U_i$  of length at least  $w(v_i)$ , then pick an interval  $I \in (0, M) - U_i$  of smallest length such that  $|I| \geq w(v_i)$ . Suppose  $I = (a, b)$ . Then assign the interval  $(a, a + w(v_i))$  to  $v_i$ .

**Case 2.** Otherwise, if all intervals in  $(0, M) - U_i$  have length less than  $w(v_i)$ , pick the largest interval  $I = (a, b)$  in  $(0, M) - U_i$  (breaking ties arbitrarily) and assign  $(a, a + w(v_i))$  to  $v_i$ . Note that this

assignment of an interval to  $v_i$  causes the interval assignment to become infeasible. This is because there is some neighbor of  $v_i$  that has been assigned an interval with left endpoint  $b$  and  $(a, a+w(v_i))$  intersects this interval. To restore feasibility, we increase the endpoints of all intervals “above”  $b$  by  $i\Delta = (a+w(v_i)) - b$ . In other words, for every vertex  $v_j$ ,  $1 \leq j \leq i$ ,  $I(v_j) = (c, d)$ , if  $c \geq b$ , then  $I(v_j) = (c + \Delta, d + \Delta)$ . It is easy to see that this restores feasibility to the interval assignment.

Consider the chordal graph shown in Figure 2. The numbers next to vertices are vertex weights and the letters are vertex labels. The ordering of vertices  $A, B, C, D, E$  is a reverse perfect elimination ordering. By the time we get to processing vertex  $E$ , the assignment of intervals to vertices is as shown in the middle in Figure 2. When  $E$  is processed, we look for “space” to fit it in and find the interval  $(10, 15)$ , which is not large enough for  $E$ . So we move the interval  $I(D)$  up by 5 units to make space for  $I(E)$  and obtain the assignment shown on the right.

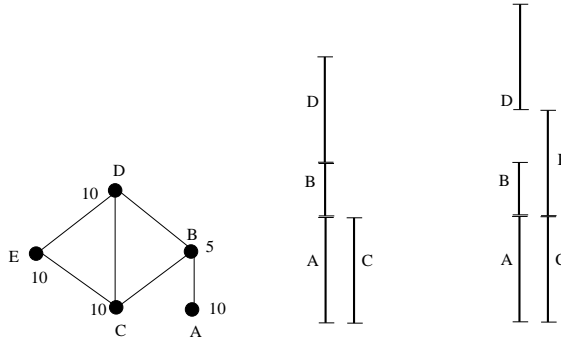


Figure 2: The best-fit heuristic in action for interval coloring.

A similar “best-fit” solution to the max-coloring problem is obtained as follows. Let  $k$  be the size of a maximum clique in  $G$ . Start with a palette of colors  $C = \{1, 2, \dots, k\}$  and an assignment of color 1 to vertex  $v_1$ . Let  $AC(v_i) \subseteq C$  be the colors available for  $v_i$ . For each color  $j$ , let  $W_j$  denote the maximum weight among all vertices colored  $j$ ; for an empty color class  $j$ ,  $W_j = 0$ . Color vertex  $v_i$  with a color  $j \in AC$  that maximizes  $W_j$ , with ties broken arbitrarily. This ensures that the color we assign to  $v_i$  minimizes the increase in the weight of the coloring.

We will call these “best-fit” algorithms for interval coloring and max-coloring, **BFI** and **BFM** respectively.

An example that forces the best-fit algorithms, **BFI** and **BFM** to perform badly is the following. Consider a graph  $G$  with  $n$  disjoint cliques, each clique containing  $n$  vertices. Let the cliques be labeled  $C_1, C_2, \dots, C_n$ . For each  $i$ ,  $1 \leq i \leq n$ , the distribution of weights of vertices in  $C_i$  is as follows: there are  $(i - 1)$  vertices with weight 2, one vertex with weight  $W$ , and  $(n - i)$  vertices with weights 1. Any ordering of vertices is a perfect elimination ordering of  $G$ . So suppose that **BFM** processes vertices in the following order: vertices of  $C_1$ , followed by vertices of  $C_2$ , followed by vertices of  $C_3$ , etc. The vertices of each  $C_i$  are ordered as follows: vertices with weight 2 come first, followed by the vertex of weight  $W$ , followed by the vertices of weight 1. It is easy to check that if vertices are processed in this order then **BFM** will produce a coloring with  $n$  color classes, such that each color class contains a vertex of weight  $W$ . This solution has weight  $n \cdot W$  as compared to **OPT** which has weight  $W + 2(n - 1)$  and thus this is an example that forces **BFM** to produce a solution at least  $\Omega(n)$  times **OPT**. In Figure 3, this example is shown as a set of intervals with  $n = 4$ . The intervals correspond to vertices and pairwise intersection of intervals corresponds to edges. Each row of intervals corresponds to a color class chosen by **BFM**. The optimal coloring in this instance would put the intervals with weight  $W$  in one row.

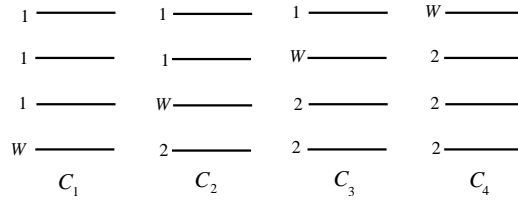


Figure 3: A bad example for the best-fit heuristic.

### 2.3 Algorithm 3: via graph partitioning

Another pair of algorithms for interval coloring and max-coloring can be obtained by partitioning the vertices of the given graph into groups with similar weight. Let  $W$  be the maximum vertex weight. Fix an integer  $k \geq 1$  and partition the range  $[1, W]$  into  $(k + 1)$  subranges:

$$\left[ W, \frac{W}{2} \right), \left[ \frac{W}{2}, \frac{W}{2^2} \right), \dots, \left[ \frac{W}{2^{k-1}}, \frac{W}{2^k} \right), \left[ \frac{W}{2^k}, 1 \right].$$

For  $i$ ,  $1 \leq i \leq k$ , let  $R_i = [W/2^{i-1}, W/2^i)$  and let  $R_{k+1} = [W/2^k, 1]$ . Partition the vertex set  $V$  into subsets  $V_i$ ,  $1 \leq i \leq (k + 1)$  defined as  $V_i = \{v \in V \mid w(v) \in R_i\}$ . For each  $i$ ,  $1 \leq i \leq (k + 1)$  let  $G_i$  be the induced subgraph  $G[V_i]$ . We ignore the weights and color each subgraph  $G_i$  with the fewest number of colors, using a fresh palette of colors for each subgraph  $G_i$ . For the max-coloring problem, we simply use this coloring as the solution.

For the interval coloring problem, we turn the coloring into an assignment of intervals to vertices as follows. Let  $C_1, C_2, \dots, C_t$  be color classes produced by the above algorithm. For each  $i$ ,  $1 \leq i \leq t$ , let  $W_i$  be the maximum vertex weight of a vertex in  $C_i$ . Let  $W_0 = 0$ . For each  $i$ ,  $1 \leq i \leq t$ , let

$$J_i = \left( \sum_{s=0}^{i-1} W_s, \sum_{s=0}^i W_s \right)$$

and assign to the interval  $J_i$  to all vertices in  $C_i$ . This is an interval coloring of  $G$  because vertices in distinct color classes are assigned disjoint intervals. The span of this interval assignment is  $\sum_{s=1}^t W_s$ . This is identical to the total weight of the solution to max-coloring as well.

We will call these graph partitioning based algorithms for interval coloring and max-coloring, GPI and GPM respectively.

**Theorem 3** *If we set  $k = 2 \log(n)$ , then GPI and GPM produce  $(4 \cdot \log(n) + o(1))$ -approximations to both the interval coloring as well as the max-coloring problems.*

**Proof:** For  $i$ ,  $1 \leq i \leq k$ , let  $\alpha_i$  be the weight of the heaviest clique in  $G[V_i]$ . Let  $\chi_i = \chi(G[V_i])$ . Clearly,  $\alpha_i \geq \chi_i \cdot W/2^i$ . Let  $\text{OPT}$  refer to the weight of an optimal max-coloring and let  $\text{OPT}_i$  refer to the weight of an optimal max-coloring restricted to vertices in  $V_i$ . Note that  $\text{opt}_i \geq \alpha_i$ . Since GPM colors each  $V_i$  with exactly  $\chi_i$  colors and since the weight of each vertex in  $V_i$  is at most  $W/2^{i-1}$ , the weight of the coloring that GPM assigns to  $V_i$  is at most  $\chi_i \cdot W/2^{i-1} \leq 2 \cdot \alpha_i \leq 2 \cdot \text{OPT}_i$ . Since GPM uses a fresh palette of colors for each  $V_i$ , the weight of the coloring of  $\cup_{i=1}^k V_i$  is at most

$$2 \cdot \sum_{i=1}^k \text{OPT}_i \leq 2 \cdot \sum_{i=1}^k \text{OPT} = 4 \log(n) \cdot \text{OPT}.$$

Since  $k = 2 \log(n)$ ,  $W/2^k = W/n^2$ . Therefore, any coloring of  $V_{k+1}$  adds a weight of at most  $W/n$  to the coloring of the rest of the graph. Since  $W \leq \text{OPT}$ , GPM colors the entire graph with weight at most  $(4 \cdot \log(n) + 1/n)\text{OPT}$ .

The lower bound on  $\alpha_i$  that was used in the above proof for max-coloring also applies to interval coloring and we get the same approximation factor for interval coloring.  $\square$

	MODE 1									
No. of maximal cliques	149	126	110	126	147	116	119	119	128	149
Size of largest clique	13	14	12	12	14	11	12	12	12	14
Size of smallest clique	4	3	5	3	5	4	4	4	3	5
Mean clique size	8.58	7.35	8.35	7.83	9.41	7.51	7.10	7.31	7.98	9.53
Variance	4.06	3.83	3.23	2.35	3.32	1.99	2.36	2.82	3.61	3.23
	MODE 2									
No. of maximal cliques	220	216	216	218	218	219	213	216	219	219
Size of largest clique	29	33	33	31	14	31	30	36	30	30
Size of smallest clique	5	3	5	7	4	5	7	5	4	4
Mean clique size	20.13	22.34	22.37	24.00	21.15	21.83	25.17	23.70	22.80	20.89
Variance	28.43	30.02	30.69	29.55	33.98	31.73	48.72	32.66	25.81	29.68

Figure 4: Properties of 20 instances of graphs with  $n = 250$  and  $\alpha = 0.9$ . Ten of these were generated in Mode 1 and the other ten in mode 2.

### 3 Overview of the Experiments

#### 3.1 How chordal graphs are generated

We have implemented an algorithm that takes in parameters  $n$  (a positive integer) and  $\alpha$  (a real number in  $[0, 1]$ ) and generates a random chordal graph with  $n$  vertices, whose sparsity is characterized by  $\alpha$ . The smaller the value of  $\alpha$  the more sparse the graph. In addition, the algorithm can run in two modes; in mode 1 it generates somewhat “regular” chordal graphs and in mode 2 it generates somewhat “irregular” chordal graphs.

The algorithm generates chordal graphs with  $n, (n - 1), \dots, 2, 1$  as a perfect elimination ordering. In the  $i$ th iteration of the algorithm vertex  $i$  is connected to some subset of the vertices in  $\{1, 2, \dots, i - 1\}$ . Let  $G_{i-1}$  be the graph containing vertices  $1, 2, \dots, (i - 1)$ , generated after iteration  $(i - 1)$ . Let  $\{C_1, C_2, \dots, C_t\}$  be the set of maximal cliques in  $G_{i-1}$ . It is well known that any chordal graph on  $n$  vertices has at most  $n$  maximal cliques. So we explicitly maintain the list of maximal cliques in  $G_{i-1}$ . We pick a maximal clique  $C_j$  and a random subset  $S \subseteq C_j$  and connect  $i$  to the vertices in  $S$ . This ensures that the neighbors of  $i$  in  $\{1, 2, \dots, i - 1\}$  form a clique, thereby ensuring that  $n, (n - 1), \dots, 2, 1$  is a perfect elimination ordering.

We use the parameter  $\alpha$  in order to pick the random subset  $S$ . For each  $v \in C_j$ , we independently add  $v$  to set  $S$  with probability  $\alpha$ . This makes the expected size of  $S$  equal  $\alpha \cdot |C_j|$ . The algorithm also has a choice to make on how to pick  $C_j$ . One approach is to choose  $C_j$  uniformly at random from the set  $\{C_1, C_2, \dots, C_t\}$ . This is mode 1 and it leads to “regular” random chordal graphs, that is, random chordal graphs in which the sizes of maximal cliques show small variance. Another approach is to choose a maximal clique with largest size from among  $\{C_1, C_2, \dots, C_t\}$ . This is mode 2 and it leads to more “irregular” random chordal graphs, that is, random chordal graphs in which there are a small number of very large maximal cliques and many very small maximal cliques. Graphs generated in the two modes seem to be structurally quite different. This is illustrated in the table in Figure 4, where we show information associated with 10 instances of graphs with  $n = 250$  and  $\alpha = 0.9$  generated in mode 1 and in mode 2. Each column corresponds to one of the 10 instances and comparing corresponding mode 1 and mode 2 rows easily reveals the the fairly dramatic difference in these graphs. For example, the mean clique size in mode 1 is about 8.5, while it is about 22 in mode 2. Even more dramatic is the large difference in the variance of the clique sizes and this justifies our earlier observation that mode 2 chordal graphs tend to have a few large cliques and many very small cliques, relative to mode 1 chordal graphs.

	MODE 1			MODE 2		
	BFM	FFM	GPM	BFM	FFM	GPM
Equals OPT	1312	2975	6	4792	1465	3
Equals $\chi$	4950	3384	9	4950	1626	6
% Deviation	10.17	2.08	49.18	0.21	4.57	34.07

Figure 5: This table shows aggregate performance over all 4950 runs of the three heuristics for max-coloring, separately for mode 1 and mode 2 graphs, when we know the value of OPT. The row “Equals OPT” lists the number of times each heuristic produces a coloring with weight equal to OPT, the row “Equals  $\chi$ ” lists the number of times each heuristic produces a coloring using minimum number of colors, and the row “Deviation” lists the percentage deviation of the weight of the solution produced from OPT, averaged over the 4950 runs.

## 3.2 How Weights are Assigned

Once we have generated a chordal graph  $G$  we assign weights to the vertices as follows. This process is parameterized by  $W$ , the maximum possible weight of a vertex. Let  $k$  be the chromatic number of  $G$  and let  $\{C_1, C_2, \dots, C_k\}$  be a  $k$ -coloring of  $G$ . Since  $G$  is a chordal graph, it contains a clique of size  $k$ . Let  $Q = \{v_1, v_2, \dots, v_k\}$  be a clique in  $G$  with  $v_i \in C_i$ . For each  $v_i$ , pick  $w(v_i)$  uniformly at random from the set of integers  $\{1, 2, \dots, W\}$ . Thus the weight of  $Q$  is  $\sum_{i=1}^k w(v_i)$ . For each vertex  $v \in C_i - \{v_i\}$ , pick  $w(v)$  uniformly at random from  $\{1, 2, \dots, w(v_i)\}$ . This ensures that  $\{C_1, C_2, \dots, C_k\}$  is a solution to max-coloring with weight  $\sum_{i=1}^k w(v_i)$  and the interval assignment induced by this coloring is an interval coloring of span  $\sum_{i=1}^k w(v_i)$ . Since  $\sum_{i=1}^k w(v_i)$  is also the weight of the clique  $Q$ , which is a lower bound on OPT in both cases, we have that  $\text{OPT} = \sum_{i=1}^k w(v_i)$  in both cases. The advantage of this method of assigning weights is that it is simple and gives us the value of OPT for both problems. The disadvantage is that, in general OPT for both problems can be strictly larger than the weight of the heaviest clique and thus by generating only those instances for which OPT equals the weight of the heaviest clique, we might be missing a rich class of problem instances.

We also tested our algorithms on instances of chordal graphs for which the weights were assigned at random. For these algorithms, we use the maximum weighted clique as a lower bound for OPT.

## 3.3 Main Observations

For our main experiment we generated instances of random chordal graphs with number of vertices  $n = 10, 20, 30, \dots, 550$ . For each value of  $n$ , we used values of  $\alpha = 0.1, 0.2, \dots, 0.9$ . For each of the  $55 \times 9$   $(n, \alpha)$  pairs, we generated 10 random vertex weighted chordal graphs. We ran each of the three heuristics for each of the two problems and averaged the weight and span of the solutions over the 10 instances for each  $(n, \alpha)$  pairs. Thus each heuristic was evaluated on 4950 instances, for each problem. The vertex weights are assigned as described above, with the maximum weight  $W$  fixed at 1010. We first conducted this experiment for the max-coloring problem on mode 1 and mode 2 chordal graphs and then repeated them for the interval coloring problem.

We then generated the same number of instances, but this time assigning to each vertex, a weight chosen uniformly from  $[0, 1010]$ . We repeated all six algorithms on these random instances, and used the maximum weighted clique as a lower bound for OPT. Note that in these cases, OPT might be quite large compared to the size of the maximum weighted clique.

### 3.3.1 Max-coloring

The data for the max-coloring problem is presented in the following tables. Figure 5 summarizes the performance of the three heuristics for the max-coloring problem for both mode 1 and mode 2 chordal

	MODE 1			MODE 2		
	BFI	FFI	GPI	BFI	FFI	GPI
Equals OPT	2512	3389	1982	2226	1367	272
% Deviation	13.98	3.24	10.89	4.35	8.91	16.47
Equals LB (R)	1251	1603	974	330	206	128
% Deviation (R)	36.78	12.86	19.86	18.87	21.05	27.37

Figure 6: This table shows aggregate performance over all 4950 runs of the three heuristics for interval coloring, separately for mode 1 and mode 2 graphs. The row “Equals OPT” lists the number of times each heuristic produces a coloring with weight equal to OPT and the row “% Deviation” lists the percentage deviation of the weight of the solution produced from OPT, averaged over the 4950 runs. The first two rows correspond to the runs when OPT is known, and the last two rows, marked with an (R) correspond to the runs where the vertex weights are assigned randomly. Here LB represents the runs where the algorithm achieved the lower bound (in which case OPT is equal to the maximum weighted clique).

graphs. The tables showing the values for the various runs are presented in the appendix for lack of space. We make the following observations regarding the max-coloring problem, based on the results.

1. First fit in decreasing weight order is clearly a heuristic that returns solutions very close to OPT for max coloring. Overall percentage deviations from OPT, each being an average over 4950 runs, are 2.08 and 4.57 for max-coloring on mode 1 and mode 2 graphs respectively, for the cases where we know OPT. In the other case first-fit deviates from the lower bound by atmost 20%.
2. The best-fit heuristic seems to be at a disadvantage because it is constrained to always use as many colors as the chromatic number. The first fit heuristic uses more colors than optimal a fair number of times. Examining Figure 5 , we note that for max-coloring, first fit uses more colors than optimal about 31.6% and 67.15% of the time for mode 1 and mode 2 graphs respectively.
3. The graph partitioning heuristic is not competitive at all, relative to best-fit and first fit, in any of the cases, despite the  $O(\log(n))$ -factor approximation guarantee it provides.

### 3.4 Interval Coloring

We now present the data for the interval coloring problem on mode 1 chordal graphs. Again, we only present the summary of the results here, while the data values for the various runs are presented in the appendix.

We summarize our results in Table 3.3.1, which shows the average deviation of the three algorithms over all the runs, for both modes, and we make the following observations.

1. The algorithm of choice for interval coloring is clearly dependent on the structure of the graph. For graphs that are fairly “uniform”, i.e., the variance in the sizes of the cliques is *not much*, first-fit outperforms the other algorithms, while if the graphs are “non-uniform”, i.e, there is *considerable* variance between the sizes of the maximal cliques, best-fit out performs the other algorithms, irrespective of whether OPT is the size of the maximum weight clique. What is significant is the performance of first-fit and best-fit as structure of the graph changes from mode 1 to mode 2. For the case when OPT is equal to the maximum weighted clique, the percentage deviation of first-fit changes as  $3.24 \rightarrow 8.91$ , while the deviation for best-fit changes as  $13.98 \rightarrow 4.35$ .
2. This suggests that the algorithm of choice for interval coloring depends on the underlying structure of the graph. However, if we cannot make any assumptions about the structure of the graph, the better algorithm to run would be first-fit, since the average deviation over all runs is still the smallest for first-fit. Approximately 19% for best-fit compared to approximately 12% for first-fit.

## 4 Conclusion

Our goal was to evaluate three algorithms for max-coloring and interval coloring of chordal graphs. For the max-coloring problem, the algorithm of choice would clearly be first-fit. For the interval coloring problem, however, the choice of the best algorithm is dependent on the structure of the graph. If the graph is fairly “regular”, then first-fit is the algorithm of choice, while if the graph is “irregular”, best-fit outperforms all algorithms. It would be interesting to investigate the behaviour of the algorithms for various classes of chordal graphs, where the value of OPT can be computed, and is not equal to the maximum weighted clique. We have also experimented with trees and disjoint cliques, and the results of these experiments are available at <http://www.cs.uiowa.edu/~rraman>. The interesting question here is to understand why best-fit performs better at interval coloring when going from mode-1 to mode-2.

## 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)*, 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] G. Confessore, P. Dell’Olmo, and S. Giordani. An approximation result for the interval coloring problem on claw-free chordal graphs. *Discrete Applied Mathematics*, 120:71–88, 2002.
- [4] J. Fabri. Automatic storage optimization. *ACM SIGPLAN Notices: Proceedings of the ACM SIGPLAN ’79 on Compiler Construction*, 14(8):83–91, 1979.
- [5] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the theory of NP-completeness*. W.H. Freeman and Company, San Fransisco, 1979.
- [6] 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.
- [7] J. Gergov. Algorithms for compile-time memory optimization. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pages S907–S908, 1999.
- [8] 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.
- [9] S. Irani. Coloring inductive graphs on-line. *Algorithmica*, 11(1):53–72, 1994.
- [10] H.A. Kierstead. The linearity of first-fit coloring of interval graphs. *SIAM J. Discrete Math*, 1:526–530, 1988.
- [11] H.A. Kierstead. A polynomial time approximation algorithm for dynamic storage allocation. *Discrete Mathematics*, 88:231–237, 1991.
- [12] H.A. Kierstead and W.T. Trotter. An extremal problem in recursive combinatorics. *Congressus Numerantium*, 33:143–153, 1981.
- [13] S.V. Pemmaraju, R. Raman, and K. Varadarajan. Buffer minimization using max-coloring. To appear in the Proceedings of The ACM-SIAM Symposium on Discrete Algorithms (SODA), 2004.
- [14] T. Rus and S.V. Pemmaraju. Using graph coloring in an algebraic compiler. *Acta Informatica*, 34(3):191–209, 1997.
- [15] R.E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, amd selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13:566–579, 1984.

# Appendix

## Max-Coloring Mode 1 graphs

V	E	OPT	Best Fit	First Fit	Partition	$\frac{(BF-OPT)}{OPT}$	$\frac{(FF-OPT)}{OPT}$	$\frac{(GPM-OPT)}{OPT}$
20	27.94	2120.47	2191.26	2134.01	2633.96	5.23	0.96	29.9
40	63.3	831.91	880.34	836.76	1108.77	7.18	0.81	37.52
60	104.03	902.81	965.87	910.27	1224.64	7.86	0.85	39.96
80	141.08	961.84	1011.41	979.26	1334.86	6.43	1.81	45.61
100	178.64	2268.47	2418.69	2295.36	3110.93	8.84	1.14	44.1
120	222.27	1356.42	1461.67	1380.1	1890.82	8.15	1.79	44.46
140	257.22	774.7	845.37	788.67	1087.79	10.8	2.17	46.02
160	305.18	2167.11	2316.02	2212.62	3105.67	7.96	1.95	49.82
180	330.37	1105.17	1195.43	1126.3	1563.79	8.36	2.39	47.04
200	380.5	139.07	147.43	141.42	196.08	5.99	2.41	45.07
220	419.54	2453.81	2709	2486	3538.46	12.71	1.44	49.67
240	466.02	730.57	806.1	744.58	1068.02	11.18	2.07	51.89
260	502.28	768.78	824.52	781.29	1119.17	9.06	2.01	52.25
280	519.13	1257.41	1404.5	1282.9	1879.96	12.75	1.97	54.61
300	581.73	1122.64	1233.69	1152.24	1666.62	11.59	3.05	52.89
320	627.44	183.9	203.12	188.54	265.49	13.13	2.86	49.22
340	679.18	2064.28	2258.54	2103.69	3084.66	11.55	2.34	55.66
360	704.46	2489.68	2685.77	2536.22	3680.5	8.35	1.86	52.57
380	736.93	559.1	612.24	574.61	828.06	10.41	2.83	51.77
400	791.91	547.11	593.43	559.38	815.3	9.2	3.07	56.13
420	840.38	525.73	579.4	536.9	761.66	11.48	2.69	48.81
440	880.51	2742.24	3043.64	2800.47	4132.78	11.95	2.69	56.89
460	918.46	2652.96	2967.26	2714.62	3974.67	14.18	2.55	55.23
480	969.79	2516.06	2774.93	2556.16	3747.43	11.24	1.55	52.62
500	997.4	1980.37	2170.87	2025.11	3000.66	11.31	2.65	57.75
520	1033.03	783.59	860.1	800.87	1185.09	11.05	2.51	56.24
540	1076.32	2547.63	2849.74	2626.68	3872.62	13.48	3.59	57.63

Figure 7: Results of our main experiments on mode 1 random chordal graphs, evaluating heuristics for the max-coloring problem.

## Max-Coloring Mode 1 graphs with random weights

$ V $	$ E $	LB	Best Fit	First Fit	Partition	$\frac{(BF-OPT)}{OPT}$	$\frac{(FF-OPT)}{OPT}$	$\frac{(GPM-OPT)}{OPT}$
20	28.0111	2376.22	2825	2594.41	3340.67	18.8862	9.18218	40.5873
40	64.7222	2793.91	3346.9	3148.41	4043.36	19.7926	12.6883	44.7203
60	104.944	2979.58	3604.57	3368.21	4463.26	20.9758	13.0432	49.7949
80	144.656	3057.64	3759.9	3499.66	4668.01	22.9672	14.4559	52.6669
100	170.9	3103.28	3816.16	3569.09	4881.24	22.9718	15.0103	57.2932
120	217.022	3177.24	3946.64	3668.24	5015.84	24.216	15.4536	57.8678
140	263.544	3354.43	4168.31	3873.58	5312.7	24.2628	15.4764	58.3785
160	310.744	3263.64	4123.96	3863.77	5306.64	26.3604	18.3881	62.5987
180	349.067	3377.77	4207.78	3946.24	5402.4	24.5728	16.83	59.94
200	369.5	3436.68	4236.58	4049.02	5544.8	23.2754	17.8179	61.3419
220	438.433	3512.17	4420.18	4052.91	5749.17	25.8533	15.3963	63.6929
240	465.822	3555.23	4466.49	4171.19	5732.43	25.6314	17.3253	61.2393
260	506.811	3563.57	4488.81	4185.33	5814.47	25.964	17.4479	63.1642
280	543.056	3670.51	4664.58	4319.58	5981.14	27.0825	17.6833	62.9513
300	619.611	3694.76	4666.72	4341	6132.84	26.3067	17.4909	65.9878
320	616.833	3683.39	4599.39	4257.07	5971.09	24.8684	15.5747	62.1086
340	695.267	3728.1	4707.9	4389.61	6141.16	26.2815	17.7439	64.7261
360	725.544	3675.27	4721.42	4376.39	6133.7	28.4648	19.0768	66.8913
380	785.122	3712.73	4653.92	4379.04	6147.21	25.3503	17.9466	65.571
400	799.322	3718.68	4700.19	4412.16	6158.34	26.3941	18.6485	65.6058
420	821.178	3758.01	4813.5	4440.74	6185.27	28.0864	18.1674	64.5888
440	853.956	3766.96	4733.96	4428.67	6248.27	25.6706	17.5662	65.8705
460	899.822	3789.99	4738.3	4480.22	6259.47	25.0215	18.212	65.1579
480	953.733	3708.09	4750.59	4444.38	6195.4	28.1142	19.8563	67.078
500	999.622	3768.08	4824.71	4468.5	6290.66	28.0417	18.5883	66.946
520	1051.66	3788.9	4818.58	4490.54	6385.32	27.1762	18.5184	68.5271
540	1093.79	3828.27	4839.43	4578.46	6463.62	26.4132	19.596	68.8394

Figure 8: Results of our experiments on mode 1 random chordal graphs, with random weights evaluating heuristics for the max-coloring problem. Here LB refers to the maximum weight clique.

## Max-Coloring Mode 2 graphs

V	E	OPT	Best Fit	First Fit	Partition	$(BF-OPT)$	$(FF-OPT)$	$(GPM-OPT)$
						$\frac{\text{Best Fit}}{OPT}$	$\frac{\text{First Fit}}{OPT}$	$\frac{\text{Partition}}{OPT}$
20	36.86	1007.4	1009.47	1018.61	1211.5	0.15	1.76	27.53
40	113.28	1161.03	1166.3	1186.99	1408.6	0.49	2.57	29.98
60	195.57	2383.88	2384.79	2447.78	2949.69	0.02	3.17	31.5
80	294.37	1354.76	1359.54	1388.53	1668.09	0.59	3.27	32
100	416.98	4254.81	4260.24	4388.09	5268.5	0.17	4.06	32.83
120	530.88	1032.18	1034.67	1062.62	1276.18	0.53	3.98	32.83
140	659.42	964.38	964.69	993.33	1195.96	0.13	4.21	33.19
160	794.01	870.62	871.52	900.9	1078.01	0.06	4.63	33.79
180	908.08	2574.7	2576.42	2668.42	3232.24	0.07	4.46	34.4
200	1058.61	1309.26	1311.86	1351.38	1636.7	0.17	4.56	34.4
220	1229.08	4954	4958.22	5117.67	6194.14	0.14	4.91	35.68
240	1336.02	4829.81	4829.81	5029.49	6122	0	5.23	35.77
260	1513.66	4538.61	4538.61	4715.86	5749.6	0	5.62	37.5
280	1668.77	249.16	249.18	258.97	305.99	0.01	6.72	29.82
300	1806.31	3122.13	3125.16	3272.02	3964.76	0.08	6.56	38.04
320	1966.82	1169.97	1173.08	1205.5	1457.74	0.41	4.38	32.92
340	2111.49	2489.43	2490.08	2561.21	3117.5	0.05	4.37	34.12
360	2207.6	5013.06	5019.56	5202.42	6414.64	0.27	5.66	36.5
380	2443.81	3467.01	3473.06	3599.31	4352.59	0.18	5.38	35.65
400	2628.93	2588.14	2590.09	2688.96	3235.04	0.26	4.98	33.48
420	2797.31	3087.11	3092.89	3196.56	3869.2	0.49	5.03	33.98
440	2944.37	5282.24	5282.24	5473.18	6705.98	0	5.58	36.87
460	3132.72	3554.21	3558.86	3674.63	4448.24	0.37	4.83	35.21
480	3290.86	2139.26	2141.69	2231.1	2715.39	0.22	5.21	36.93
500	3492.46	5202.62	5208.01	5407.56	6561.06	0.22	6.12	35.37
520	3564.52	5360.71	5364.63	5567.66	6794.72	0.16	4.94	36.77
540	3785.6	1052.72	1053.03	1095.91	1319.3	0.08	5.56	34.35

Figure 9: Results of our main experiments on mode 2 random chordal graphs, evaluating heuristics for the max-coloring problem.

## Max-Coloring Mode 2 graphs with random weights

$ V $	$ E $	LB	Best Fit	First Fit	Partition	$\frac{(BFI-OPT)}{OPT}$	$\frac{(FFI-OPT)}{OPT}$	$\frac{(GPI-OPT)}{OPT}$
20	33.9778	2617.92	3135.13	2877.84	3420.83	19.7565	9.92857	30.6698
40	108.989	3579.93	4492.93	4144.81	4864.37	25.5033	15.779	35.8787
60	195.344	4125.07	5278.97	4811.07	5620.78	27.9729	16.63	36.2591
80	300.522	4545.92	5842.42	5302.06	6248.58	28.5201	16.6332	37.4546
100	415.611	4930.46	6340.47	5807.21	6822.6	28.598	17.7824	38.3767
120	534.711	5007.94	6674.97	6027.22	7103.28	33.2876	20.3532	41.8402
140	654.722	5402.37	7088.69	6405.62	7526.36	31.2145	18.5707	39.3159
160	799.078	5477.86	7339.93	6594.4	7700.49	33.9928	20.3829	40.5749
180	917.8	5652.33	7557.41	6722.86	7948.07	33.7043	18.9395	40.6157
200	1032.77	5613.7	7683.96	6792.93	8030.8	36.8786	21.0063	43.0572
220	1204.66	5796.89	8002.99	7022.63	8350.62	38.0566	21.1449	44.0535
240	1351.2	5877.72	8096.71	7077.89	8455.37	37.7525	20.4189	43.8545
260	1529.18	6172.01	8454.62	7396.43	8771.43	36.9833	19.8383	42.1163
280	1670.63	6349.47	8686.23	7656.33	9033.13	36.8026	20.5823	42.266
300	1808.93	6486.83	8804.82	7770.63	9191.58	35.7338	19.7909	41.6959
320	1977.26	6538.11	8888.31	7837.77	9316.49	35.9462	19.8782	42.4951
340	2105.8	6506.6	9002.73	7863.92	9343.47	38.3631	20.8607	43.5998
360	2259.06	6500.41	9195.9	7919.62	9315.26	41.4664	21.8326	43.3026
380	2448.57	6559.79	9295.09	7999.06	9481.11	41.698	21.9407	44.5338
400	2579.76	6623.37	9270.36	8096.2	9643.78	39.9644	22.2369	45.6024
420	2819.09	7024.57	9626.78	8426.91	10017.4	37.0444	19.9634	42.6052
440	2991.97	6919.62	9724.36	8390.56	9996.6	40.533	21.2574	44.4674
460	3097.56	6694.3	9469.39	8189.7	9747.19	41.4545	22.3384	45.6043
480	3315.63	7062.68	9891.43	8462.93	10052.4	40.0522	19.8261	42.3314
500	3491.01	7246.79	9977.18	8754.11	10388.8	37.6772	20.7999	43.3567
520	3615.3	7151.23	10029.5	8588.54	10183.7	40.2487	20.0988	42.405
540	3873.46	7378.29	10098.1	8812.96	10676.6	36.8622	19.4444	44.7035

Figure 10: Results of our main experiments on mode 2 random chordal graphs, with random weights evaluating heuristics for the max-coloring problem.

## Interval Coloring Mode 1 graphs

$ V $	$ E $	OPT	Best Fit	First Fit	Partition	$\frac{(BFI-OPT)}{OPT}$	$\frac{(FFI-OPT)}{OPT}$	$\frac{(GPI-OPT)}{OPT}$
20	24.2556	1885.12	1949.19	1903.39	1965.38	3.39854	0.968991	4.25731
40	67.5333	2292.77	2372.34	2333.63	2469.91	3.47082	1.78242	7.72623
60	98.6111	2479.49	2656.53	2529.43	2633.49	7.14036	2.0143	6.21096
80	138.9	2553.66	2703.09	2592.46	2743.8	5.85174	1.51939	7.44597
100	183.456	2583.14	2778.1	2674.96	2788.21	7.54722	3.55424	7.93864
120	227.156	2674.79	2917.73	2740.39	2929.06	9.08275	2.45253	9.50605
140	259.089	2640.33	2870.84	2704.13	2880.5	8.73038	2.41636	9.09607
160	299.833	2799.21	3133.31	2876.71	3107.34	11.9355	2.76864	11.0079
180	344.156	2684.37	3078.98	2770.38	2995.03	14.7003	3.20415	11.5732
200	386.022	2674.04	2954.73	2749.43	2971.99	10.4968	2.81928	11.1421
220	434.933	2866.38	3288.71	2973.67	3257.5	14.734	3.74301	13.6452
240	456.489	2735.6	3173.8	2837.41	3034.12	16.0184	3.72171	10.9125
260	524.344	2886.42	3239.98	2992.83	3258.82	12.2489	3.68661	12.9018
280	562.333	2915.48	3396.34	3013.74	3261.63	16.4936	3.37052	11.873
300	602.944	2799.39	3255.93	2882.12	3141.19	16.3087	2.95541	12.2098
320	638.456	2917.19	3408.61	3032.81	3241.28	16.8457	3.96348	11.1096
340	691.6	2935.66	3457.93	3070.11	3318.41	17.7908	4.58009	13.0382
360	718.733	3002.07	3543.6	3082.78	3361.53	18.0387	2.68852	11.974
380	793.144	2961.61	3448.62	3067.83	3308.57	16.4441	3.58664	11.7151
400	808.178	3006.5	3555.89	3122.96	3403.43	18.2734	3.87346	13.2025
420	838.811	2895.71	3463.99	3029.21	3239.32	19.6248	4.61027	11.8662
440	875.5	3097.68	3668.82	3216.02	3437.86	18.4378	3.82042	10.9817
460	940.011	3010.78	3622.51	3139.28	3381.62	20.3181	4.268	12.3172
480	987.033	3069.97	3643.03	3194.78	3471.02	18.6669	4.06555	13.0638
500	1016.47	3018.96	3652.63	3126.13	3422.17	20.99	3.55016	13.356
520	1060.01	3068.37	3725.8	3179.87	3400.93	21.4262	3.63386	10.8386
540	1120.18	3189.69	3810.51	3323.84	3628.48	19.4634	4.20591	13.7565

Figure 11: Results of our main experiments on mode 1 random chordal graphs, evaluating heuristics for the interval coloring problem.

## Interval Coloring Mode 1 graphs with random weights

$ V $	$ E $	LB	Best Fit	First Fit	Partition	$\frac{(BFI-OPT)}{OPT}$	$\frac{(FFI-OPT)}{OPT}$	$\frac{(GPI-OPT)}{OPT}$
20	26.8222	2354.47	2514.91	2441.41	2590.94	6.81447	3.69274	10.0438
40	65.4667	2793.79	3182.16	3015.37	3139.42	13.9011	7.93109	12.3715
60	98.1222	2900.13	3459.82	3200.39	3298.09	19.2987	10.3532	13.722
80	138.911	3025.34	3730.73	3361.86	3572.19	23.316	11.1231	18.0754
100	175.989	3137.23	3995.07	3504.59	3698.63	27.3436	11.7095	17.8947
120	221.489	3246.92	4213.96	3573.64	3803.58	29.7831	10.0625	17.1441
140	261.3	3316.77	4237.59	3676.37	3939.13	27.7626	10.8419	18.7643
160	304.589	3349.5	4345.77	3773.23	4031.22	29.7437	12.6506	20.353
180	339.744	3364.79	4489.1	3733.14	3964.03	33.414	10.9474	17.8093
200	388.922	3487.59	4702.46	3960.63	4250.49	34.834	13.5637	21.8747
220	417.511	3439.33	4748.81	3861.61	4116.83	38.0736	12.2779	19.6986
240	465.622	3564.61	4774.09	3985.03	4317.4	33.9301	11.7943	21.1184
260	500.833	3511.46	4831.2	4016.06	4266.38	37.584	14.3701	21.4988
280	547.156	3621.37	4935.97	4079.52	4346.26	36.3012	12.6515	20.017
300	611.878	3641	5147.13	4206.07	4438.81	41.3659	15.5195	21.9119
320	646.844	3632.78	5053.3	4158.73	4420.82	39.1029	14.4781	21.6926
340	677.422	3600.43	5307.17	4112.14	4360.98	47.4036	14.2125	21.1237
360	708.389	3636.84	5215.73	4140.89	4413.28	43.4137	13.8594	21.3491
380	738.656	3653.37	5373.68	4219.02	4455.33	47.0884	15.4831	21.9514
400	800.833	3790.56	5389	4357.7	4544.91	42.1691	14.962	19.9009
420	867.378	3771.04	5433.61	4331.66	4608.32	44.0877	14.8662	22.2028
440	890.633	3751.67	5564.04	4290.47	4549.29	48.3086	14.3616	21.2605
460	899.422	3706.61	5579.24	4293.46	4503.38	50.5214	15.8324	21.4958
480	972.989	3770.38	5594.34	4365.38	4627.62	48.3762	15.7809	22.7363
500	1039.4	3788.44	5705.79	4384.34	4682.8	50.6103	15.7294	23.6075
520	1034.57	3785.07	5645.89	4345.94	4671.27	49.1622	14.8182	23.4131
540	1137.82	3827.2	5764.21	4461.3	4774.82	50.6117	16.5682	24.7602

Figure 12: Results of our main experiments on mode 1 random chordal graphs, with random weights evaluating heuristics for the interval coloring problem.

## Interval Coloring Mode 2 graphs

$ V $	$ E $	OPT	Best Fit	First Fit	Partition	$\frac{(BFI-OPT)}{OPT}$	$\frac{(FFI-OPT)}{OPT}$	$\frac{(GPI-OPT)}{OPT}$
20	35.8667	2432	2500.38	2457.86	2600.98	2.81159	1.06314	6.9481
40	109.556	3335.94	3424.66	3458.87	3629.51	2.65925	3.68478	8.80011
60	203.089	3867.24	3969.92	4049.57	4297.57	2.65506	4.71453	11.1274
80	303.6	3983.06	4048.6	4240.38	4554.28	1.64558	6.46042	14.3413
100	409.578	4241.47	4381.33	4565.77	4863.81	3.2976	7.64594	14.6729
120	523.911	4413.39	4551.53	4784.73	5065.86	3.13012	8.41404	14.7838
140	655.778	4782.32	4966.99	5206.98	5446.64	3.86144	8.87969	13.8912
160	781.756	5152.17	5326.09	5567.19	5957.64	3.37571	8.05529	15.6338
180	926.378	5080.64	5270.76	5473.6	5902.91	3.74187	7.73436	16.1843
200	1072.34	5421.77	5604.29	5881.51	6370.67	3.36647	8.47961	17.5017
220	1213.28	5545.61	5762.98	6056.4	6490.36	3.91962	9.21069	17.0359
240	1339.73	5480.04	5681.98	5995.14	6501.72	3.68488	9.39956	18.6436
260	1480.26	5695.61	5929.38	6313.43	6670.63	4.10433	10.8473	17.1188
280	1654.29	5857.57	6085.37	6418.36	6874.07	3.88899	9.57375	17.3536
300	1827.76	6031.32	6382.63	6590.99	7079.63	5.82478	9.27934	17.3811
320	1949.17	5572.78	5844.69	6109.32	6568	4.87927	9.62795	17.8586
340	2128.6	6031.9	6333.58	6647.39	7091.83	5.00137	10.2039	17.5721
360	2280.6	5898.62	6168.66	6535.21	6969.13	4.57791	10.7922	18.1485
380	2403.33	5873.04	6177.52	6476.66	6934.98	5.18433	10.2777	18.0815
400	2632.37	6066.16	6387.16	6677.3	7189.98	5.29165	10.0747	18.5261
420	2756.43	6362.79	6630.59	7050.39	7570.03	4.20885	10.8066	18.9735
440	2917.2	6182.98	6605.76	6808.89	7305.22	6.83777	10.1231	18.1505
460	3147.48	6431.33	6743.31	7137.57	7612.78	4.8509	10.9811	18.3701
480	3256.69	6334.67	6676.6	7065.41	7610.31	5.39781	11.5356	20.1375
500	3422.84	6395.4	6750.6	7157.43	7744.3	5.55399	11.9153	21.0917
520	3666.49	6630.22	6965.03	7285.79	7931.58	5.04977	9.88755	19.6276
540	3782.43	6403.79	6797.48	7138.03	7662.57	6.14775	11.4658	19.6568

Figure 13: Results of our main experiments on mode 2 random chordal graphs, evaluating heuristics for the interval coloring problem.

## Interval Coloring Mode 2 graphs with random weights

$ V $	$ E $	LB	Best Fit	First Fit	Partition	$\frac{(BFI-OPT)}{OPT}$	$\frac{(FFI-OPT)}{OPT}$	$\frac{(GPI-OPT)}{OPT}$
20	35.4111	2736.94	2940.54	2944.44	3070.48	7.43895	7.58145	12.1863
40	109.3	3619.4	3960.23	4130.19	4353.67	9.41685	14.1125	20.287
60	201.567	4016.52	4514.96	4710.27	4988.93	12.4096	17.2723	24.2103
80	305.778	4599.7	5216.71	5446.9	5729.69	13.4142	18.4186	24.5666
100	407.733	4779.99	5574.44	5734.87	5927.17	16.6204	19.9766	23.9996
120	534.244	5080.24	5892.5	6148.76	6414.76	15.9885	21.0327	26.2686
140	661.222	5373.57	6282.02	6399.18	6753.03	16.906	19.0862	25.6713
160	791.522	5379.14	6460.83	6590.64	6915.19	20.1089	22.5222	28.5556
180	903.644	5659.76	6740.32	6815.61	7167.44	19.0921	20.4224	26.6388
200	1076.37	5854.59	6830.21	7224.81	7488.83	16.6642	23.4042	27.9139
220	1211.64	5968.71	7054.93	7201.4	7571.26	18.1986	20.6525	26.8491
240	1338.13	5969.88	7236.98	7329.67	7760.87	21.2249	22.7775	30.0004
260	1527.03	6363.59	7484.54	7754.19	8114.28	17.6151	21.8524	27.511
280	1685.17	6280.87	7541.18	7765.64	8109.21	20.0659	23.6397	29.1097
300	1789.98	6490.52	7801.33	7968.41	8312.38	20.1958	22.77	28.0695
320	1911.62	6281.07	7692.9	7842.99	8220.39	22.4776	24.8671	30.8757
340	2112.26	6648.44	7955.83	8209.59	8683.2	19.6646	23.4813	30.605
360	2308.61	6597.4	8010.04	8071.09	8564.29	21.4121	22.3374	29.8131
380	2462.37	6857.11	8311.03	8466.91	8828.51	21.2031	23.4764	28.7497
400	2584.28	6800	8210.2	8375.16	8740.73	20.7382	23.1641	28.5402
420	2730.77	6693.14	8319.68	8223.09	8801.21	24.3015	22.8584	31.4959
440	2958.31	6940.89	8431.81	8543.13	9072.14	21.4803	23.0841	30.7058
460	3107.87	6895.26	8450.82	8488.9	8977.16	22.56	23.1122	30.1932
480	3251.54	7010.97	8681.73	8574.38	9276.71	23.8308	22.2995	32.3171
500	3452.28	7058.18	8744.14	8646.87	9213.89	23.8867	22.5085	30.542
520	3625.52	7366.5	9047.21	9139.46	9543.09	22.8156	24.0678	29.5471
540	3822.16	6998.47	8739.58	8706.7	9230.33	24.8785	24.4087	31.8908

Figure 14: Results of our main experiments on mode 2 random chordal graphs with random weights, evaluating heuristics for the interval coloring problem.

Times opt BF,FF,lg :157 369 50

$ V $	$ E $	OPT	Best Fit	First Fit	Partition	$\frac{(BFI-OPT)}{OPT}$	$\frac{(FFI-OPT)}{OPT}$	$\frac{(GPI-OPT)}{OPT}$
10	9	1407.62	1444.64	1466.14	1967.14	2.63012	4.15752	39.7495
20	19	1584.57	1739.28	1810.56	2485.18	9.76362	14.2619	56.8364
30	29	1642.99	1839.51	1955.1	2712.39	11.9613	18.9965	65.0887
40	39	1739.31	1892.96	2095.71	2929.83	8.83364	20.4909	68.4479
50	49	1783.21	1898.3	2171.6	3056.08	6.45402	21.7803	71.3806
60	59	1790.3	1928	2239.78	3117.48	7.69145	25.1063	74.1316
70	69	1825.59	1945.98	2238.13	3229.71	6.59452	22.5979	76.9134
80	79	1845.38	1933.09	2332.14	3291.1	4.75302	26.3776	78.3429
90	89	1816.9	1944.28	2255.67	3332.64	7.01072	24.1492	83.4248
100	99	1842.19	1944.78	2246.8	3375.39	5.56886	21.9636	83.2271
110	109	1856.69	1956.18	2323.83	3427.48	5.3584	25.1601	84.6016
120	119	1873.88	1962.83	2346.18	3454.28	4.74714	25.2044	84.3385
130	129	1865.06	1965.78	2369.87	3425.9	5.40049	27.0668	83.6889
140	139	1870.02	1972.47	2356.72	3507.84	5.47825	26.0264	87.583
150	149	1886.34	1977.24	2435.9	3489.44	4.81884	29.1334	84.9845
160	159	1888.1	1977.33	2396.86	3512.9	4.72609	26.9454	86.0548
170	169	1906.5	1980.38	2479.39	3526.92	3.87505	30.0492	84.9946
180	179	1911.77	1992.51	2527.34	3581.28	4.22355	32.1994	87.3282
190	189	1893.93	1985.66	2517.33	3579.98	4.84295	32.9156	89.0234
200	199	1913.12	1992.78	2591.06	3607.7	4.16364	35.436	88.5766
210	209	1915.13	1995.27	2567.26	3572.73	4.18422	34.051	86.5527
220	219	1915.32	1994.73	2498.57	3613.63	4.1461	30.4515	88.6697
230	229	1911.81	1995.58	2562.14	3617.41	4.38153	34.0166	89.2138
240	239	1920.57	1998	2629.8	3625.22	4.0318	36.9283	88.7579
250	249	1917.19	2002.43	2601.5	3644.9	4.44632	35.6935	90.1169
260	259	1925.26	2000.91	2595.97	3656.08	3.92964	34.8375	89.9009
270	269	1917.37	1998.68	2653.68	3654.97	4.24077	38.4022	90.6243
280	279	1915.38	2001.28	2634.32	3661.32	4.48475	37.5354	91.1541
290	289	1919.93	2002.5	2608.59	3668.22	4.3005	35.8687	91.0599
300	299	1936.2	2004.19	2655.42	3675.36	3.51146	37.1461	89.8231
310	309	1937.28	2003.7	2676.61	3685.98	3.42864	38.1635	90.2658
320	319	1937.98	2006.59	2718.33	3698.38	3.54035	40.2665	90.837
330	329	1933.53	2006.36	2668.64	3695.32	3.76628	38.0191	91.1176
340	339	1938.5	2007.64	2765.94	3714.08	3.5669	42.6848	91.5954
350	349	1941.34	2006.87	2727.22	3698.37	3.3751	40.4811	90.5054
360	359	1939.92	2005.52	2692.7	3724.62	3.38158	38.8045	91.9985
370	369	1938.6	2005.66	2781.06	3719.06	3.45897	43.4569	91.8423
380	379	1929.51	2002.64	2706.23	3720.53	3.79025	40.2549	92.8226
390	389	1939.81	2005.11	2722.36	3730.97	3.36631	40.3413	92.3366
400	399	1949.52	2007.97	2749.46	3748.51	2.99789	41.0323	92.2784
410	409	1937.59	2005.53	2745.9	3740.19	3.50665	41.7174	93.0332
420	419	1945.28	2007.11	2837.24	3734.14	3.17864	45.8529	91.9594
430	429	1950.82	2009.16	2764.42	3750.61	2.99019	41.7055	92.258
440	439	1955.81	2009.77	2785.59	3742.98	2.75873	42.4263	91.3773
450	449	1941.96	2007.59	2816.48	3757.68	3.37975	45.0331	93.4997
460	459	1947.37	2007.34	2749.26	3764.59	3.07994	41.1781	93.3169
470	469	1951.58	2004.59	2767.99	3750.99	2.71632	41.8334	92.2029
480	479	1953.38	2008.09	2872.06	3766.39	2.80085	47.0302	92.8142
490	489	1955.29	2010.2	2841.41	3756.98	2.80834	45.3192	92.1444
500	499	1951.11	2008.06	2845.92	3772.78	2.91856	45.8616	93.3656

Figure 15: Max-coloring on random trees with maximum degree = 20

$ V $	$ E $	OPT	Best Fit	First Fit	Partition	$\frac{(BFI-OPT)}{OPT}$	$\frac{(FFI-OPT)}{OPT}$	$\frac{(GPI-OPT)}{OPT}$
10	9	1407.62	1493.08	1453.31	1604.38	6.07092	3.24582	13.9779
20	19	1584.57	1844.12	1799.97	1887.97	16.3802	13.5936	19.1472
30	29	1642.99	2016.58	1932.18	2041.47	22.7384	17.6014	24.2532
40	39	1739.31	2129.44	2047.86	2183.02	22.4303	17.7395	25.5107
50	49	1783.21	2242.09	2132.04	2212.32	25.7332	19.5621	24.064
60	59	1790.3	2268.31	2175.56	2246.4	26.7001	21.5191	25.4762
70	69	1825.59	2243.17	2188.82	2312.14	22.8736	19.8968	26.652
80	79	1845.38	2374.13	2282.5	2348.9	28.653	23.6874	27.2856
90	89	1816.9	2287.37	2266.44	2359.54	25.8939	24.7424	29.8665
100	99	1842.19	2285.48	2252.54	2344.88	24.0632	22.2754	27.2876
110	109	1856.69	2346.5	2280.82	2370.92	26.3809	22.8435	27.6963
120	119	1873.88	2349.57	2267.61	2430.8	25.3853	21.0117	29.7203
130	129	1865.06	2434.34	2345.7	2400.49	30.524	25.7711	28.7087
140	139	1870.02	2342.84	2284.79	2470.67	25.2843	22.1798	32.1196
150	149	1886.34	2405.47	2377.96	2459.32	27.52	26.0616	30.375
160	159	1888.1	2476.32	2425.27	2475.41	31.1542	28.4501	31.1059
170	169	1906.5	2450.44	2367.58	2483.49	28.531	24.1845	30.2643
180	179	1911.77	2479.83	2449.71	2504.62	29.7142	28.1386	31.0109
190	189	1893.93	2486.59	2435.48	2522.56	31.2923	28.5936	33.1914
200	199	1913.12	2471.67	2523.47	2473.58	29.1954	31.9031	29.2953
210	209	1915.13	2542.81	2454.66	2512.63	32.7746	28.1715	31.1989
220	219	1915.32	2457.07	2424.88	2509.74	28.2848	26.6042	31.0351
230	229	1911.81	2463.63	2482.2	2585.47	28.8638	29.835	35.2365
240	239	1920.57	2529.57	2540.16	2530.62	31.7094	32.2607	31.7644
250	249	1917.19	2552.31	2604.39	2566.07	33.1278	35.8441	33.8453
260	259	1925.26	2496.41	2521.51	2600.73	29.6665	30.9702	35.0851
270	269	1917.37	2551.66	2539.57	2581.18	33.0813	32.4508	34.621
280	279	1915.38	2489.48	2571.38	2558.17	29.9732	34.2491	33.5594
290	289	1919.93	2561.84	2527.73	2580.63	33.434	31.6573	34.4127
300	299	1936.2	2570.07	2649.94	2556.01	32.7377	36.8632	32.0117
310	309	1937.28	2558.47	2612.09	2605.79	32.065	34.833	34.5078
320	319	1937.98	2576.44	2619.44	2556.33	32.945	35.1638	31.9073
330	329	1933.53	2594.36	2655.22	2582.82	34.1769	37.3249	33.5804
340	339	1938.5	2560.4	2674.49	2616.94	32.0815	37.9669	34.9984
350	349	1941.34	2541.79	2609.2	2645.29	30.9293	34.4017	36.2607
360	359	1939.92	2585.74	2696.62	2598.17	33.2911	39.0067	33.9315
370	369	1938.6	2580.88	2666.09	2671.04	33.131	37.5265	37.7821
380	379	1929.51	2584.36	2582.33	2638.17	33.9384	33.8336	36.7272
390	389	1939.81	2591.3	2624.58	2640.88	33.5852	35.3007	36.141
400	399	1949.52	2620.07	2671.24	2666.82	34.3953	37.0205	36.7936
410	409	1937.59	2588.68	2664.17	2643.04	33.603	37.4991	36.4089
420	419	1945.28	2608.32	2807.77	2631.03	34.0848	44.3376	35.2523
430	429	1950.82	2610.1	2772.8	2711.31	33.7949	42.1349	38.983
440	439	1955.81	2600.1	2695.13	2677.67	32.9423	37.8013	36.9082
450	449	1941.96	2604.71	2751.47	2697.63	34.1283	41.6854	38.9132
460	459	1947.37	2611.68	2630.02	2691.31	34.1133	35.0553	38.2026
470	469	1951.58	2636.02	2800.1	2711.54	35.0713	43.4788	38.9411
480	479	1953.38	2633.03	2727.57	2675.49	34.7939	39.6333	36.9673
490	489	1955.29	2627.47	2762.7	2746.79	34.3774	41.2937	40.48
500	499	1951.11	2618.07	2831.74	2710.73	34.1834	45.135	38.9328

Figure 16: Interval coloring on random trees with maximum degree = 20