On Spectrum Assignment in Elastic Optical Tree-Networks

Fatima Zahra Moataz

On Spectrum Assignment in Elastic Optical Tree-Networks

Fatima Zahra Moataz*

Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France

INRIA, France

Abstract

To face the explosion of the Internet traffic, a new generation of optical networks is being developed; the Elastic optical Networks (EONs). The aim with EONs is to use the optical spectrum efficiently and flexibly. This benefit of the flexibility is accompanied by more difficulty in the resource allocation problems. In this report, we study the problem of Spectrum Allocation in Elastic Optical Tree-Networks. In trees, even though the routing is fixed, the spectrum allocation is NP-hard. We survey the complexity and approximability results that have been established for the SA in trees and prove new results for stars and binary trees.

1 Introduction

Elastic Optical Networks (EONs) [9] have been proposed recently as a potential candidate to replace the traditional Wavelength Division Multiplexing (WDM) networks. In EONs, new technologies such as optical OFDM, adaptive modulation techniques, bandwidth variable transponders, and flexible spectrum selective switches are used to ensure an efficient utilization of the optical resources and to enable a flexible grid as opposed to the WDM fixed-grid. In fact, the optical spectrum in EONs, is subdivided into small channels, called slots, which are finer than the 50GHz wavelengths used under WDM. With these slots, small bitrates are not over-provisioned and big bitrates can be satisfied as single entities, under the constraint of contiguity. This constraint dictates that the slots used by a request should be consecutive. This results in an efficient use of the spectrum but it also makes the problems of resource allocation in EONs more difficult than their counterparts in WDM.

The key resource allocation problem in Elastic Optical Networks is referred to as Routing and Spectrum Assignment (RSA). For static RSA, the input is a set of traffic requests and the objective is to allocate to each request, a path in the optical network and an interval of spectrum slots along that path, minimizing the utilized spectrum. The spectrum allocated to a demand has to be contiguous (contiguity constraint), it has to be the same over all links of the routing path (continuity constraint) and demands sharing a link should be assigned disjoint spectrum intervals (non-overlapping constraint). If the routing is fixed, i.e., a path is predefined for each request, RSA reduces to the problem of Spectrum Assignment (SA).

Related work Spectrum Allocation is a generalization of the well studied problem of Wavelength Assignment (WA). Since WA has been proved NP-complete in [4], SA is also NP-complete. In fact, SA remains NP-hard even in networks where WA is tractable, particularly in path networks. When the network is a path, SA is equivalent to the Dynamic Storage Allocation (DSA). Hence, as for DSA [2], SA is strongly

*This author is supported by a grant from the "Conseil régional Provence Alpes-Côte d'Azur".

NP-complete even if the requests demands do not exceed 2 slots. Recent papers have taken advantage of the relation between SA and other problems to draw some hardness and approximation results for restricted cases. In [21], SA is studied from a scheduling perspective. It is proved that SA is NP-hard in path networks with as much as 4 links and unidirectional rings with 3 links, and approximation algorithms of scheduling are used to find approximation for SA in path networks with few links. In [20], SA is studied from an interval coloring point of view. Algorithms for interval coloring are used to provide an $2+\epsilon$-approximation algorithm for SA in path networks, $4+\epsilon$-approximation in ring networks and $log(k)$-approximation for binary trees where k is the number of requests. To the best of our knowledge, no other paper presents results on SA in tree networks.

Contribution In this paper, we study the spectrum assignment problem in trees. We focus on special cases where the tree is a star or where the demands of the requests are bounded by a constant and the tree is binary. By studying these special cases, we hope to gain more insight into the general problem and design a constant-factor approximation algorithm or prove that such algorithm does not exist. In our study, we follow the tendency and use relation to other problems to draw new results. We prove that SA is NP-hard in undirected stars of 3 links and in directed stars of 4 links, and show that it can be approximated within a factor of 4. Afterwards, we use techniques used for the DSA problem to find constant-factor approximation algorithms for SA on binary trees when the demands are bounded.

This report is organized as follows. We define the problem of Spectrum Assignment and other equivalent problems in Section 2. We survey the complexity and approximability results of SA in paths in Section 3. Finally, we present our results on stars and trees in Sections 4 and 5, respectively.

2 Preliminaries

In this section, we first define the problem of spectrum assignment and related notions. Afterwards, we list some problems to which SA can reduce or is equivalent. Finally, we introduce to algorithms that will be of use in the paper.

2.1 Spectrum Assignment

Given a graph N modeling an optical network, and a set of requests $\mathcal{R}$ where each $r \in \mathcal{R}$ has a path p_r and a spectrum demand d_r (number of slots), a spectrum assignment of $(G, \mathcal{R})$ is a mapping f from $\mathcal{R}$ to $\mathbb{N}^*$ such that for every two requests $r, r' \in \mathcal{R}$, if $p_r \cap p_{r'} \neq \emptyset$ then $[f(r), f(r)+d_r-1] \cap [f(r'), f(r')+d_{r'}-1] = \emptyset$. We say that all the slots in $[f(r), f(r)+d_r-1]$ are occupied by r. The ***span*** of a spectrum assignment f, denoted $s(f)$, is the smallest integer b such that for each request $r \in \mathcal{R}$, $f(r)+d_r-1 \leq b$. The span of an instance $(G, \mathcal{R})$, denoted by $s(N, \mathcal{R})$ is the minimum of spans over all possible spectrum assignments. We formulate the spectrum assignment problem as follows:

Problem 1 (Spectrum Assignment (SA)). *Given an instance $(N, \mathcal{R})$, compute $s(N, \mathcal{R})$.*

For an instance of SA, the ***load of an edge*** e is the sum of the demands of the requests using e and the ***load of an instance*** is the maximum load over all its edges. Two requests are **conflicting** if their paths share an edge.

For an instance of SA, we create a weighted graph $G = (V, E, w)$ modeling the dependency between the different requests and call it ***the conflict graph***. We associate to every request $r \in \mathcal{R}$ a vertex v_r in V. We add an edge between two vertices v_r and $v_{r'}$ if the corresponding requests r and r', use paths sharing at least one edge, i.e., $p_r \cap p_{r'} \neq \emptyset$. The weight $w(v)$ of each vertex v_r is equal to the bandwidth requirement of the corresponding request r. We define the weight of a subset $S \subset V$ to be the quantity $w(S) = \sum_{x \in S} w(x)$. A clique of a graph $G = (V, E)$ is a subset $K \subset V$ of vertices that are pairwise adjacent in G. A clique is maximal if it is not contained in any other clique and it is maximum if it contains the biggest number of vertices among all cliques. The clique number of G is the size of its maximum clique and is denoted by $\omega(G)$. In a weighted graph $G = (V, E, w)$, the maximum weighted clique is a maximal clique with the biggest

weight. The weighted clique number of $G = (V, E, w)$ is the weight of the maximum weight clique and is denoted by $\omega_w(G)$.

Let $(N, \mathcal{R})$ be an instance of SA snd $G = (V, E, w)$ its conflict graph. A conflict set in $\mathcal{R}$ corresponds to a clique in $G(V, E, w)$ and the density of $(N, \mathcal{R})$ is equal to the weighted clique number of G, i.e., $d(N, \mathcal{R}) = \omega_w(G)$.

2.2 Spectrum Assignment and equivalent problems

As it has been mentioned in the introduction, SA is closely related to many problems that have been studied in numerous contexts. We highlight in this subsection some of these relations.

Dynamic Storage Allocation (DSA). Given s set A of items to be stored, each $a \in A$ having size $s(a)$, an arrival time $r(a)$, and a departure time $d(a)$ (with $d(a) > r(a)$), and a positive integer storage size D. Is there an allocation of storage for A, i.e. a function $\sigma : A \to \{1, 2, \ldots, D\}$ such that for every $a \in A$ the allocated storage interval $I(a) = [\sigma(a), \sigma(a) + s(a) - 1]$ is contained in $[1, D]$ and such that, for all $a, a' \in A$ with $a \neq a'$, if $I(a) \cap I(a')$ is non empty then $[r(a), d(a)] \cap [r(a'), d(a')]$ is empty? When the network is a path, it is easy to see that the DSA problem is equivalent to SA.

Scheduling tasks on multiprocessor systems ($P|f_{ix_j}C_{max}|$). Givem a set of n tasks and a set of identical processors, a processing time p_j and a prespecified set f_j of processors for task j, $j = 1, \ldots, n$, schedule the tasks so as to minimize the makespan $C_{max} = max_j C_j$, where C_j denotes the completion time of task j, under the following constraints: (1) preemptions (interruptions of a task) are not allowed, (2) each task must be processed simultaneously by all processors in f_j, and (3) each processor can work on at most one task at a time. The problem of SA can be reduced to scheduling tasks on multiprocessor systems as it has been noted in [21].

Interval (or Contiguous) Coloring (IC). As defined in [11] (p 203), An interval coloring of a weighted graph $G = (V, E, w)$ maps each vertex x onto an (open) interval I_x of the real line of width (or measure) $w(x)$ such that adjacent vertices are mapped to disjoint intervals, that is, $xy \in E$ implies $I_x \cap I_y \neq \emptyset$. The number of hues of a coloring is defined to be $|\cup_x I_x|$. The interval chromatic number $\chi_w(G)$ is the least number of hues needed to color the vertices with intervals. The interval coloring problem is defined as follows.

Problem 2 (Interval Coloring (IC))**.** *Given a vertex-weighted graph $G = (V, E, w)$, compute $\chi_w(G)$.*

The problem of Spectrum allocation is equivalent to the problem of Inteval Coloring. In fact, if $(N, \mathcal{R})$ is an instance of SA and $G = (V, E, w)$ is its conflict graph, then $s(N, \mathcal{R}) = \chi_w(G)$.

2.3 Some useful algorithms

The greedy algorithm (or First Fit) The ***greedy algorithm*** for SA is an algorithm which assigns spectrum to requests in a given order $r_1, \ldots, r_n$; a request r_i is assigned the smallest positive integer $g(r)$ such that $[g(r_i), g(r_i)+d_i-1] \cap [g(r_j), g(r_j)+d_j-1] = \emptyset$ for each r_j in $\{r_1, \ldots, r_{i-1}\}$ if $p_{r_j} \cap p_{r_i} \neq \emptyset$. The ***greedy algorithm*** for IC is an algorithm which assigns color intervals to vertices in a given order $v_1, \ldots, v_n$; a vertex v_i is assigned the smallest positive integer $g(v)$ such that $[g(v_i), g(v_i)+w(v_i)-1] \cap [g(v_j), g(v_j)+w(v_j)-1] = \emptyset$ for each v_j in $\{v_1, \ldots, v_{i-1}\}$ if $p_{v_j} \cap p_{v_i} \neq \emptyset$.

Buddy-decreasing-size Algorithm [5] The buddy-decreasing-size operates as follows. We first increase the demands of all requests in $\mathcal{R}$ to the closest power of 2. Afterwards we order the requests in the decreasing order of their new demands. We allocate spectrum to the requests using the greedy algorithm in this order. Let $(N, \mathcal{R})$ be an instance of SA. We define the weighted degree of a request as the sum of its demand and the demands of the requests conflicting with it. We define the maximum weighted degree Δ as the maximum degree over all possible requests. We prove the following lemma.

Lemma 1. *Given an instance $(N, \mathcal{R})$ with maximum weighted degree Δ, the buddy-decreasing-size uses at most 2Δ slots.*

Proof. Let $(N, \mathcal{R})$ be an instance of SA. After increasing the sizes of the requests, the maximum degree is at most 2Δ. Let $r_1, r_2, \ldots, r_q$ be the order of the requests in the decreasing demand. Let g be the spectrum assignment defined by the buddy-decreasing-size algorithm. We prove that for each $i \in \{1, q\}$, $g(r_i) = 1$ or all slots $s < g(r_i)$ are occupied by neighbors of r_i. We proceed by induction. For $i = 1$, $g(r_i) = 1$. We suppose that that for each $i < j$ either $g(r_i) = 1$ or all slots $s < g(r_i)$ are occupied by neighbors of r_i. $d_{r_j} = 2^a$. If no neighbors of r_j have been assigned spectrum then $g(r_j) = 1$. Otherwise, let s be the smallest slot not occupied by neighbors of r_j. If $s = 1$, let r_l be the first neighbor of r_j such that $g(r_l) > s$. Since r_l verifies the induction assumption, $g(r_l) = k2^b$ where $b \geq a$. There is then enough space to have $g(r_j) = 1$. If $s \neq 1$, let r_l be the neighbor of r_j occupying slot $s-1$. If there is no other neighbor of r_j occupying slots $> s$ then we can have $g(r_j) = s$ and we know that all the slots $< s$ are occupied by neighbors of r_j. Otherwise, let $r_{l'}$ be the be the first neighbor of r_j such that $g(r_{l'}) > s$. With the induction assumption we know that $g(r_{l'}) = k'2^{b'}$ and $s = k2^b + 1$. The space between r_l and $r_{l'}$ is $k'2^{b'} - k2^b - 1$ is enough to have $g(r_j) = s$.

We deduce from this that the number of slots does not exceed 2Δ. □

3 Spectrum assignment in paths: survey

It has been proven in [21], that the spectrum assignment remains NP-complete even if the network is a path of four edges. The proof is by reduction from the problem of 2-PARTITION. In the paper, the authors examine the spectrum allocation problem as a multiprocessor scheduling problem and propose an algorithm to solve the problem when the path is of length 3. Some approximation algorithms are also presented: 1.5-approximation for four and five edges and , a two-stage algorithm that yields a constant approximation ratio for a path with a fixed number of edges.

3.1 SA in paths vs DSA

As we have mentioned beforehand, the problem of spectrum allocation is equivalent to the problem of Dynamic Storage Allocation. This latter has been extensively studied. We survey in what follows some of the results that have been established for DSA and by equivalence for SA in paths.

3.1.1 NP-completeness

Dynamic Storage Allocation is NP-complete, even when restricted to instances where $s(a) \in \{1, 2\}$. The proof of NP-completeness is by reduction from the 3-PARTITION problem and it can be found in the appendix of [2]. Dynamic Storage Allocation has been extensively studied. The best approximation algorithm has a ratio of $2 + \epsilon$.

Problem 3 (3-PARTITION)**.** ***Input:*** *Set W of $3m$ elements, a bound B, and a positive integer size $z(w)$ for each $w \in W$, such that $\sum_{w \in W} z(w) = mB$, and $B/4 < z(w) < B/2$ for all w.*
Question: *Can W be partitioned into disjoint sets $W_1, \ldots, W_m$ such that $\sum_{w \in W_i} = B$ for $1 \leq i \leq m$*

3.1.2 Approximation

Many algorithms have been designed to approximate the problem of Dynamic Storage Allocation. The first algorithms are based on the First Fit (FF) algorithm and its performance for online coloring of interval graphs. In fact, the relation between online coloring of interval graphs and contiguous coloring of interval graphs can be found in [5]. In the paper, they propose an algorithm called Buddy-decreasing-size where they sort the items in the descending size (after increasing the size of each item to a power of 2) and then place each item as low as possible in the memory. This algorithm performs as well as online first fit for chunk items of size one arriving in the same order as the decreasing order of the big items.

The approach using First Fit has yielded a 6-approximation. The linearity of first fit is proven in [15] and the ratio of 6 in [16].

Gregov has adopted another approach not using FF, yielding an approximation of 5 and 3 sequentially in [7] and [8] . In his approach, Gregov defines a 2-allocation where two items but not three are allowed to overlap. This 2-allocation is achieved by allocating to every item r (in the decreasing order of the size after rounding them to powers of 2) the biggest stable position, which is a position a such that all the positions below a have been used by a clique of requests conflicting with r. Gregov proves after that the graph of conflict of requests in the 2-allocation has a property that allows to have the approximation.

A better approximation has been achieved in [3]. The authors use the idea of boxing items to design a $2+\epsilon$-approximation algorithm: they gather items, which are modeled as rectangles, in larger rectangles so that the total wasted space in the rectangles remains small and then use an algorithm to assign memory to the large rectangles.

Paper [17] presents better approximation algorithms when the items have restricted sizes. A $\frac{4}{3}$-approximation when the maximum size is 2, and a $1,7$-approximation when the maximum size is 3. The authors start by solving the problem exactly for small basic cases with small clique numbers and then use these small cases to approximate the general problem. The key ideas they use are : channel flip, partitioning of the memory into many channels and the partitioning of items into subsets of small density. Each subset can be provisioned in a channel.

In [18], it is proved that for instances with sizes of 1 and X, ordered by starting times, First Fit has a guarantee of $2-\frac{1}{X}$.

3.2 SA in paths vs Interval Coloring

Ihe conflict graph of an instance $(N,\mathcal{R})$ where N is a path is an interval graph. A graph is an **interval graph** if every $v \in V(G)$ corresponds to an interval I_v such that $\{u,v\} \in E(G)$ if and only if $I_u \cap I_v \neq \emptyset$. The set of intervals $\{I_v : v \in VG\}$ is the interval representation of G. It is known that an interval representation of a given interval graph can be found in polynomial time. **A proper interval graph** is an interval graph having an interval representation such that no interval is properly contained in another. The problem of SA in paths is equivalent to the Interval Coloring of Interval graphs.

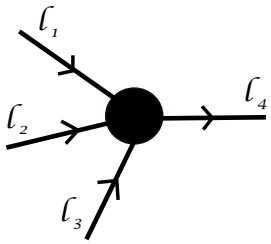
Interval coloring in a proper interval graph It has been proven in [On the Interval Chromatic Number of Proper Graphs] by Mordechai Shalom that the problem is strongly NP-hard in Proper Interval Graphs.

In [**?**], the authors propose a 2-approximation algorithm for the problem of spectrum allocation in proper interval graphs.

The approximation comes from the fact that the interval chromatic number is at most 2 times the circular chromatic number and that in the case of proper interval graphs the circular chromatic number is equal to the max clique. Actually **the interval chromatic number is at most the circular chromatic number plus the** $max(weight)-1$ **which gives good additive approximation when the maximum size is bounded.**

4 Spectrum assignment in stars: complexity and approximability results

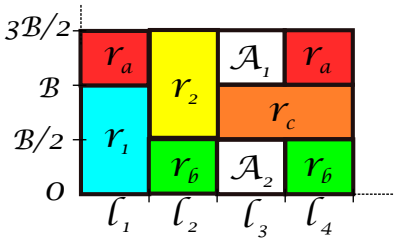
A star is a tree-network with at most one node of degree at least 2. The problem of wavelength assignment (WA) is NP-complete in undirected stars but polynomial in directed stars [1]. The polynomiality of WA in directed stars was useful because optical networks are symmetrically directed and because it helped in the design of constant-factor approximation algorithms for WA in directed trees [1]. Such algorithm cannot be extended to SA since we prove in this section that SA is not only NP-compelte in undirected stars but also in directed stars with 4 links. On the positive side, we prove the existence of a 4-approximation algorithm for the general case and show that there are better approximation algorithms for stars with few links.



(a) Star S

Request	Demand	Path
r_a	$B/2$	$l_1 l_4$
r_b	$B/2$	$l_2 l_4$
r_c	$B/2$	$l_3 l_4$
r_1	B	l_1
r_2	B	l_2

(b) Requests R



(c) Spectrum assignment

Figure 1

Theorem 1. *The problem of Spectrum Assignment is strongly NP-complete in undirected stars with 3 links.*

Proof. It is shown in [21] that the SA problem is NP-complete in a 3-link unidirectional ring. Let us consider an instance of SA in a 3-link ring $C = (l_1, l_2, l_3)$ with a request set $\mathcal{R}$. Let us build a star S with three edges e_1, e_2 and e_3, and a set of requests $\mathcal{R}'$ defined as follows. For each request $r \in \mathcal{R}$ using at most 2 links, we create a request r' in $\mathcal{R}'$ such that if the path of r is $p_r = l_i$, $i \in \{1, 2, 3\}$, then the path of r' is $p_{r'} = e_i$, and if $p_r = l_i l_j$, then $p_{r'} = e_i e_j$. Solving SA in $(C, \mathcal{R})$ is equivalent to solving SA in $(S, \mathcal{R}')$. □

Theorem 2. *The problem of Spectrum Assignment is weakly NP-complete in directed stars with 4 links.*

Proof. The proof is by reduction from the 2-PARTITION problem. Given an instance of the 2-PARTITION problem with a set of k integers $A = \{a_1, a_2, \ldots, a_k\}$ such that $B = \sum_{j=1}^{k} a_j$, we create an instance of spectrum assignment in a 4-links directed star network S (Figure 1a) and a set of requests $\mathcal{R}$. The set of requests $\mathcal{R}$ consists of the requests R in Figure 1b plus a request of size a_i for every integer a_i in the set A, all using link l_3. We prove that finding a spectrum assignment for $(S, \mathcal{R})$ with span $\frac{3}{2}B$ is equivalent to finding a partition of A into two sets A_1 and A_2 such that $\sum_{a_j \in A_1} a_j = \sum_{a_j \in A_2} a_j = \frac{B}{2}$. In fact, if there is a partition of A into A_1 and A_2 such that $\sum_{a_j \in A_1} a_j = \sum_{a_j \in A_2} a_j = \frac{B}{2}$, then we can assign spectrum as shown in Figure 1c. Now let us suppose there is a spectrum assignment for $(S, \mathcal{R})$ with span $\frac{3}{2}B$. There are two possible symmetric assignments to the requests on links l_1 and l_2. We suppose we assign to r_1, r_a, r_2 and r_b spectrum intervals $[0, B]$, $[B, \frac{3}{2}B]$, $[\frac{B}{2}, \frac{3}{2}B]$, and $[0, \frac{B}{2}]$, respectively (the analysis is similar for the other assignment). This assignment forces request r_c to use the interval $[\frac{B}{2}, B]$ and the other requests on link l_3 will have to be partitioned into two sets of the same width $\frac{B}{2}$. □

Theorem 3. *The problem of Spectrum Assignment in directed stars with at most 3 links or exactly 2 ingoing links and 2 outgoing links can be solved in polynomial time.*

Proof. In any of these cases, the span is equal to the load and the greedy algorithm with specific orders can achieve the optimal span.

- When the star has only ingoing or outgoing links, the problem is trivial since there is no conflict between the requests and the greedy algorithm with any order can achieve the optimal span.
- For the case where the star is a directed path of length 2, an optimal spectrum assignment consists in using the greedy algorithm with an order where the requests using two links come first. This way, the spectrum span will be defined by the link with the maximum load.
- For the case where the star has two ingoing links l_1 and l_2 and one outgoing link l_3 (or the opposite), an optimal spectrum assignment consists in using the greedy algorithm with an order where the requests using l_1 and l_3 come first and the requests using l_2 and l_3 come last.

- When the star has 2 ingoing links l_1 and l_2 and 2 outgoing links l_3 and l_4, an optimal assignment consists in using the greedy algorithm with the following order: requests using l_1 and l_3, then requests using l_2 and l_4, then requests using only one link, then requests using l_1 and l_4, and finally requests using l_2 and l_3.

□

Theorem 4. *There is a 4-approximation algorithm for the problem of Spectrum Assignment in stars (directed or not).*

Proof. Let $(S, \mathcal{R})$ be an instance of SA on a star S. Let l be its load. We have proved that the buddy-decreasing-size algorithm does not use more than 2Δ colors, where Δ is the maximum weighted degree. A request in $\mathcal{R}$ does not use more that 2 links, the maximum degree is then bounded by $2l$. Since $l \leq s(S, \mathcal{R})$ we deduce that the buddy-decreasing-size algorithm achieves a 4-approximation in stars. □

Theorem 5. *There are approximation algorithms with ratios $\frac{7}{6}$ and 1.5 when the star has 3 and 4 links, respectively.*

Proof. It has been proved in [21] that SA in a network of k links transforms to to the problem of Scheduling tasks on multiprocessor systems $(P|f_{ix_j}|C_{max})$ with k multiprocessors. The reduction is easy, given an instance $(G, \mathcal{R})$ of SA, an instance of $P|f_{ix_j}|C_{max}$ is constructed as follows. for each link l of G, there is a processor w_l, and for each request r in $\mathcal{R}$, there is a task t_r with processing time d_r and a set of processors $\{w_l \mid l \in p_r\}$. It has been proven in [10] and [13] that $P|f_{ix_j}|C_{max}$ can be approximated within $\frac{7}{6}$ and 1.5 when the number of processors is 3 and 4, respectively. Our results follow. □

5 Spectrum assignment in binary trees: approximability results

It is proven in [20] that there is an approximation algorithm with performance bound of $O(log(k))$ for SA when G is a binary tree where $k = |R|$ is the number of requests. This comes from the fact that the intersection graph of paths in a tree is chordal [12] and there is an approximation algorithm for interval coloring of chordal graphs in [19]. In the remainder of this section, we adress SA in binary trees as an interval coloring problem in chordal graphs. Let us first define **chordal graphs**. A chord of a cycle C in a graph is an edge of the graph connecting two vertices that are non-adjacent in C. A graph is chordal (or triangulated) if every cycle of it with at least 4 vertices has a chord. The conflict graph of an instance $(N, \mathcal{R})$ where N is a binary tree is a chordal graph [12].

One important property of chordal graphs is their perfect elimination order. The **perfect elimination order (PEO)** of a chordal graph is an ordering $v_1, v_2, \ldots, v_n$ of the vertices of the graph such that for $i = 1, \ldots, n-1$, the neighbors of v_i in $G[\{v_{i+1}, \ldots, v_n\}]$ form a clique. Paper [22] describes a linear time algorithm called maximum cardinality search that can be used to determine if a given graph has a perfect elimination order and construct such an ordering if it exists. Throughout the remainder of this report, we will be using **the reverse perfect elimination order (RPEO)**. Note that if $v_1, v_2, \ldots, v_n$ is a RPEO of the vertices of a chordal graph, then for $i = 2, \ldots, n$, the neighbors of v_i in $G[\{v_1, \ldots, v_{i-1}\}]$ form a clique.

In [19], several $O(log(n))$-approximation algorithms are presented for contiguous coloring in chordal graphs (First Fit in the decreasing order of the size, partitioning of requests into sets of similar size). Their experiments show that algorithm that do well for max-coloring do well for contiguous coloring .

There is a 2-approximation algorithm for contiguous coloring of claw-free chordal graphs in [6]. They study the equivalent problem of finding an acyclic orientation of an undirected graph G such that each oriented path is covered by a limited number of maximal cliques of G.

The problem of whether contiguous coloring has a constant factor approximation is still open (as mentioned in [19]).

In this section, we present constant-factor approximation results for SA in trees when the demands are bounded by a constant. It is important to recall here that routing on trees is unique and that even if the network is a path and the demands are bounded by 2, SA is still NP-complete. Let us also note again, that

the SA problem in binary tree is equivalent to the problem of Interval Coloring in chordal graphs [20]. This equivalence together with an approximation algorithm proposed for Interval Coloring in chordal graphs in [19] allow to prove the following theorem.

Theorem 6. *There is a $2log(W)$-approximation for SA where W is the maximum demand.*

Proof. We set $k = \log_2(W)$ Let us partition the set of vertices $\mathcal{R}$ into k subsets S_i, $i \in \{1, \ldots, k\}$ such that for each request $r \in S_i$, $d_r \in [\frac{W}{2^{i-1}}, \frac{W}{2^i}]$. We ignore the weight and optimally color each graph G_i induced by the subset S_i (the graphs are chordal and hence easy to color and the number of used colors will be at most $\omega(G_i)$ where $\omega(G)$ is the clique number of G). We transform this coloring into an interval coloring with at most $\frac{W}{2^i}\omega(G_i)$ colors. The number of colors used for all sets is:

$$c = \sum_{i=1}^{k} \frac{W}{2^i}\omega(G_i)$$

Note that $\frac{W}{2^{i-1}}\omega(G_i) \leq \chi_w(G)$. We have then

$$c \leq \sum_{i=1}^{k} 2\chi_w(G) = 2k\chi_w(G)$$

The $2log(W)$-approximation follows. □

We aim at finding better approximations. For this purpose, we use techniques introduced in [17] to approximate DSA. Results in [17] can extend directly to SA in path networks giving approximation algorithms with factors $\frac{4}{3}$ and 1.7 when the spectrum demands are bounded with 2 and 3, respectively. In what follows we use the same techniques to design constant-factor approximations for SA in binary trees when the spectrum demand is bounded by 6.

Theorem 7. *There are approximation algorithms for the problem of Spectrum Assignment in binary tree networks of factors $\frac{3}{2}$, $\frac{19}{10}$, $\frac{67}{30}$, $\frac{659}{240}$ and $\frac{603}{200}$ when the maximum request demand is bounded by 2,3,4,5 and 6, respectively.*

Proof. Let $G = (V, E, w)$ be the conflict graph of an instance of SA on a binary tree. The graph G is chordal. We do the proof for the problem of interval coloring in chordal graphs.

We use $d(G)$ to refer to the maximum weighted clique number of G and we call it the density of G. The density with respect to a vertex $v \in V$ is the weight of the maximum weighted clique that contains v. We denote it by $d_v(G)$. It is straightforward that $\omega_v(G) \leq \omega(G)$ for each $v \in V$.

Let $CC(d, h)$ denote the class of weighted chordal graphs where the maximum weight is h and the weighted clique number d. Let $L(d, h)$ denote the smallest W such that for each instance S of $CC(d, h)$, there is an interval coloring with at most W colors (if such W exists).

Key idea. The idea of the algorithms is to first compute $L(d, h)$ for small values of d and then use the results to solve the general cases as follows. In an instance of density D and maximum weight h, we partition the vertices into multi-level blocks (subsets) with small weighted clique numbers. Namely, n_i level-i blocks of density d_i, $i \in \{1, \ldots, k\}$. Afterwards, we use the algorithm used to compute $L(d_i, h)$ to color each level-i block. The number of colors used at the end will be equal to $\sum_{i=1,2,3} n_i L(d_i, h)$. To minimize this number, one has to carefully choose the d_i and L_i.

We detail the proof for the cases we have studied.

The maximum weight is two: We first solve the basic cases:

- $L(2, 2) = 2$. First fit in the reverse perfect elimination order can be used for an optimal color assignment to any instance with a clique weight of 2.

- $L(3,2) = 4$. We can see from Figure 2 that $L(3,2) \neq 3$. In fact, if we only use three colors, one of the vertices of weight 1 will be assigned the color in the middle and it will not be possible to color its neighbor of weight 2 with contiguous colors. Using First Fit in the reverse perfect elimination order, we prove $L(3,2) = 4$. In fact, when a new vertex arrives (in the reverse PEO), its neighbors that arrived earlier form a clique. Thus if the vertex has weight one, its neighbors will occupy at most 2 colors and if the vertex has weight two, its neighbors occupy at most one color, in both cases other colors can be assigned.
- $L(4,2) = 5$. We can see from Figure 3 that $L(4,2) \neq 4$. Let v_1 be the vertex of weight 1 and degree 4, v_2 and v_3 be the other two vertices of weight 1 and w_1 and w_2 the two vertices of weight 2 and degree 2. If we suppose that we have only four colors and w_1 is assigned colors 1 and 2, then its neighbors v_1 and v_2 will be assigned colors 3 and 4. This implies that vertex v_3 has either color 1 or 2. This constrains vertex w_2 to take colors 3 and 4 which is not possible since one of its neighbors v_1 already has one of them. If we take a set of 5 contiguous colors labeled $1, 2, 3, 4, 5$, vertices in the RPEO and try to color them, we will have a problem only when a vertex of weight 2 arrives and colors 2 and 4 are used by its neighbors of weight 1. To remedy to this, we choose one of these colors and forbid it to vertices of weight 1. This way, whenever there is a vertex of weight 1 it can be assigned and when there is a vertex of weight 2 it cannot be blocked because the only fragmenting combination is not possible.

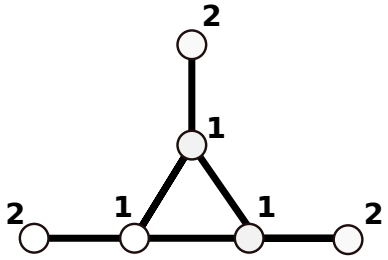


Figure 2: An example showing that $L(3,2) \neq 3$

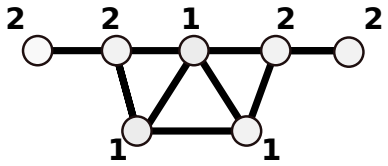


Figure 3: An example showing that $L(4,2) \neq 4$

Now let us design a Multi-level Block Partitioning algorithm as in [17].

- **Phase 1:** Let $n_1 = \lceil \frac{M}{3} \rceil$. We define n_1 level-1 subsets (blocks) of vertices denoted by $B_1^1, B_2^1, \ldots, B_{n_1}^1$, each with maximum density (recall that the maximum density is the weight of the maximum weighted clique) $m_1 = 3$. Each vertex v is assigned using first fit (in reverse perfect elimination order) to the first subset that can accommodate it. If v does not fit into any of these subsets, it is left unassigned (for Phase 2).
- **Phase 2:** Let $n_2 = \lceil \frac{M}{6} \rceil$ level-2 subsets denoted by $B_1^2, B_2^2, \ldots, B_{n_2}^2$ each with maximum density $m_2 = 2$. It is trivial to assign the remaining vertices using first fit to these blocks.

  Proof. We first prove the correctness of the algorithm. All vertices of weight 1 are assigned in phase 1. Suppose that this is not true and that a vertex v of weight 1 cannot be assigned to any of the

level-1 blocks. That means that the density with respect to v in each of these blocks is 3. Since vertices are assigned using the RPEO (all neighbors of v that have been assigned form a clique), we have $d_v(G) > 3n_1 \geq D$ which is not possible. Hence, there are only vertices of weight 2 left in Phase 2. Suppose that a vertex r of demand 2 cannot be assigned to any one of the level-2 subsets. This implies that when v is considered, each of the n_2 subsets has density 2 with respect to v (since there are no requests of demand 1). Furthermore, each of the level-1 subsets must have load at least 2 (with respect to v). Again, since vertices are assigned using the RPEO, this implies that $d_v(G) > 2n_1 + 2n_2 \geq D$, which is not possible. All the vertices are then assigned to the blocks. As for the number of colors needed. Each of the n_1 level-1 subsets can be solved using the algorithm for $L(3,2)$ and each of the n_2 level-2 subsets with the algorithm for $L(2,2)$. Thus the total number of needed colors is $4n_1 + 2n_2 \leq \frac{5}{3}D + 6$ (the constant 6 is introduced by the rounding of some values). □

Using the 2-phase algorithm differently, we can obtain a better approximation of $\frac{3}{2}$. We use $n_1 = \frac{M}{4}$ blocks of density 4, and $n_2 = \frac{M}{8}$ blocks of density 2, again all vertices of size 1 are assigned in the first phase, if we cannot assign a vertex of size two in the second phase that means that all blocks of phase 1 are of density at least 3 and that the density of the graph is bigger than $3\frac{M}{4} + 2\frac{M}{8}$. As for the number of colors needed, each of the n_1 level-1 subsets can be solved using $L(4,2)$ and each of the n_2 level-2 subsets with $L(2,2)$. **Thus the total number of tracks needed is** $5n_1 + 2n_2 \leq \frac{3}{2}D + c$..

This is the best approximation we can have with respect to the density. In fact there is a family of graphs where the ratio of the interval chromatic number to the density tends to $\frac{3}{2}$. In more details, we have $L(4k+1,2) > 6k$. Let us consider the following example. We build a clique of $4k+1$ vertices of weight 1 each. For each $2k+1$ vertices of size one, we add k vertices of size 2 building a new clique of weight $2k+1$. Let us suppose that we use at most $6k$ colors. The vertices of size one will use $4k+1$ colors. At least $2k+1$ odd (or equivalently even) colors will be used. Let us consider the $2k+1$ vertices of weight 1 use all odd colors. They form a clique of weight $2k+1$ with k vertices of weight 2 each. These vertices need exactly k odd colors to be colored. This means that all in all we will need $3k+1$ odd colors. We only have $3k$ odd colors. This means that $L(4k+1, 2 > 6k)$ and when k goes to infinity, the ratio of the interval chromatic number to the density approaches $\frac{3}{2}$.

In what follows, we will ignore the additive constant.

The maximum weight is 3 Let us start by checking the basic cases.

- $L(3,3) = L(3,2) = 4$. To solve it, we use First fit in reverse perfect elimination order.
- $L(3,\{2,3\}) = 3$.
- $L(4,3) = 6$. We know that $L(4,3) \geq L(4,2) = 5$ and we have $L(4,3) \neq 5$ as can be seen in the example 4. In fact, if we only use 5 colors, one of the vertices of weight 1 will use color 3 (because one of the colors 2, 4 has to be unused if we want to color all vertices of weight 2), the neighbor of that vertex that has size 3 cannot be colored then. We have $L(4,3) = 6$ because if we use First Fit with reverse perfect elimination order; there is no scenario where a vertex cannot be colored.
- $L(4,\{2,3\}) = 4$. We use First Fit with RPEO.
- $L(5,\{1,3\}) \neq 6$. As an illustrating example, we build a clique of 5 vertices o weight 1 each. We connect each pair of vertices of weight 1 to a new vertex of weight 3.
- $L(5,\{2,3\}) = 5$. To solve it we always assign to a vertex of size 2 colors at one end of the color interval (1 and 2 or 4 and 5).
- $L(5,3) = 7$. The algorithm is as follows. Use First Fit and forbid to vertices of weight 1 colors 5 and 6. This way there will be no fragmentation scenario to block the coloring of any new vertex.

- $L(6, \{2, 3\}) = 7$. As illustrated by the example in Figure 5, $L(6, \{2, 3\}) \neq 6$. Let us prove that $L(6, \{2, 3\}) = 7$. We partition the set of colors into two subsets: $S_1 = 1, 2, 3, 4$ and $S_2 = 5, 6, 7$. If a vertex cannot use colors from the first subset, it uses colors from the second (never colors from both subsets). In fact if a vertex v of weight 3 arrives and cannot be colored with colors from the first subset, then a neighbor of v uses colors of S_1, colors in S_2 are then not used by any neighbors of v because otherwise the density will exceed 6. If a vertex v of weight 2 arrives and cannot be colored with colors from S_1, then either a neighbor of v of weight 3 or two neighbors of weight 2 use colors of S_1, colors in S_2 are then not used by any neighbors of v because otherwise the density will exceed 6.

- $L(6, 3) \neq 8$. As an illustrating example we build a clique of 6 vetices of weight 1 each plus a neighbor of weight 3 for each triple of the clique.

- $L(6, 3) = 9$. The algorithm is as follows. Forbid the last three colors to vertices of weight 1. Vertices of weight 2 should be assigned colors 12,34,56 or 78. A vertex of size 1 can always be colored. When a vertex of size 2 arrives, if all its neighbors that arrived before have weight one, the vertex can be assigned colors 7 and 8. If it has one neighbor of weight 2 and two vertices of weight 1, one of the channels 12,34,56 or 78 will be always available. It is the same when it has one neighbor of weight 3 and another of weight 1. Now if a vertex of size 3 arrives, there is no possible fragmented scenario that can make the coloring impossible.

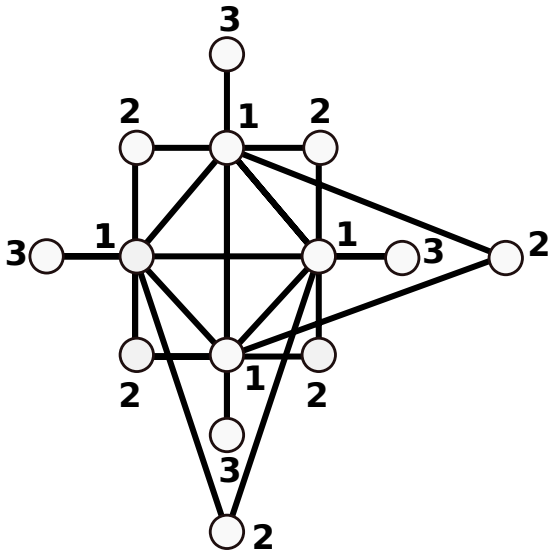


Figure 4: An example showing that $L(4, 3) \neq 5$

Let us now see the approximation we can obtain from the basic cases above. If we set $n_1 = \frac{M}{3}$, $n_2 = \frac{M}{6}$, $n_3 = \frac{M}{9}$ and the densities $m_1 = m_2 = m_3 = 3$. All vertices of weight 1 are allocated in the first group of blocks, all vertices of weight 2 in the second and only requests of size 3 in the third. We use $L(3, 3)$, $L(3, \{2, 3\})$ and $L(3, \{3\})$ **to obtain an approximation of** $4\frac{1}{3} + 3\frac{1}{6} + 3\frac{1}{9} = \frac{13}{6}$.

We can try a different partitioning. We set $n_1 = \frac{M}{3}$, $n_2 = \frac{M}{9}$, $n_3 = \frac{4M}{27}$ with densities $m_1 = 3$, $m_2 = 4$ and $m_3 = 3$. We use $L(3, 3)$, $L(4, \{2, 3\})$ and $L(3, \{3\})$ and we obtain an approximation of $4\frac{1}{3}+4\frac{1}{9}+3\frac{4}{27} = \frac{20}{9}$.

We can try a different partitioning. We set $n_1 = \frac{M}{5}$, $n_2 = \frac{M}{20}$, $n_3 = \frac{M}{12}$ with densities $m_1 = 5$, $m_2 = 5$ and $m_3 = 3$. We use $L(5, 3)$, $L(5, \{2, 3\})$ and $L(3, \{3\})$ and we obtain an approximation of $7\frac{1}{5} + 5\frac{1}{20} + 3\frac{1}{12} = \frac{19}{10}$. **This is the best ratio we have obtained.**

We can try a different partitioning. We set $n_1 = \frac{M}{6}$, $n_2 = \frac{M}{30}$, $n_3 = \frac{M}{15}$ with densities $m_1 = 6$, $m_2 = 6$ and $m_3 = 3$. We use $L(6, 3)$, $L(6, \{2, 3\})$ and $L(3, \{3\})$ and we obtain an approximation of $9\frac{1}{6} + 7\frac{1}{30} + 3\frac{1}{15} = \frac{58}{30}$.

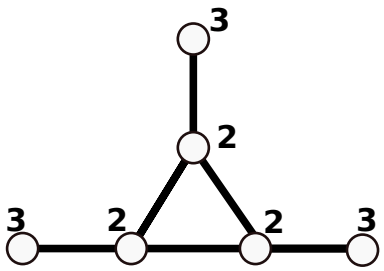


Figure 5: An example showing that $L(6, \{2, 3\}) \neq 6$

The maximum weight is 4**:** Let us solve some of the basic cases.

- $L(4, 4) = L(4, 3) = 6$
- $L(4, \{2, 3, 4\}) = 4$. Vertices of weight two use either colors $1, 2$ or $3, 4$.
- $L(4, \{3, 4\}) = 4$
- $L(5, 4) = 8$. First we have $L(5, 4) \geq L(5, 3) = 7$. To prove that $L(5, 4) \neq 7$, we consider the following example. We build a clique of 5 vertices of weight 1 each, we connect each vertex of weight 1 to a new vertex of weight 4, finally we connect each two vertices of weight 1 to a new vertex of weight 3. We cannot use only 7 colors to color this example. In fact, $L(5, 4) = 8$. We use first fit and forbid colors 3 and 6 to requests of weight 1.
- $L(5, \{2, 3, 4\}) = L(5, \{2, 3\}) = 5$
- $L(5, \{3, 4\}) = 4$
- $L(6, 4) = 9$. Forbid colors 6,7 and 8 to vertices of weight 1. Vertices of weight 2 should be assigned colors 12,34,56 or 78.
- $L(6, \{2, 3, 4\}) = 8$. First, we have $L(6, \{2, 3, 4\}) \geq L(6, \{2, 3\}) = 7$. In fact $L(6, \{2, 3, 4\}) \neq 7$, consider the example where we have a clique of three vertices of weight 2 each. Each vertex of weight 2 is incident to a vertex of weight 4. To have $L(6, \{2, 3, 4\}) = 8$, the algorithm uses First Fit and assigns to vertices of size 2 colors 12, 34, 56, or 78.
- $L(6, \{3, 4\}) = 6$

Let us now see the approximation we can obtain from the basic cases above.

We set $n_1 = \frac{M}{4}$, $n_2 = \frac{M}{12}$, $n_3 = \frac{M}{9}$ and $n_4 = \frac{M}{16}$, and the densities $m_1 = 4$, $m_2 = 4$, $m_3 = 4$ and $m_4 = 4$ **we obtain an approximation of** $6\frac{1}{4} + 4\frac{1}{12} + 4\frac{1}{9} + 4\frac{1}{16} = \frac{91}{36}$**.**

We set $n_1 = \frac{M}{5}$, $n_2 = \frac{M}{20}$, $n_3 = \frac{M}{12}$ and $n_4 = \frac{M}{16}$, and the densities $m_1 = 5$, $m_2 = 5$, $m_3 = 4$ and $m_4 = 4$ **we obtain an approximation of** $8\frac{1}{5} + 5\frac{1}{20} + 4\frac{1}{12} + 4\frac{1}{16} = \frac{73}{30}$**.**

We set $n_1 = \frac{M}{6}$, $n_2 = \frac{M}{30}$, $n_3 = \frac{M}{20}$ and $n_4 = \frac{M}{16}$, and the densities $m_1 = 6$, $m_2 = 6$, $m_3 = 6$ and $m_4 = 4$ **we obtain an approximation of** $9\frac{1}{6} + 8\frac{1}{30} + 6\frac{1}{20} + 4\frac{1}{16} = \frac{139}{60}$**.**

We set $n_1 = \frac{M}{6}$, $n_2 = \frac{M}{30}$, $n_3 = \frac{M}{15}$ and $n_4 = \frac{M}{20}$, and the densities $m_1 = 6$, $m_2 = 6$, $m_3 = 4$ and $m_4 = 4$ **we obtain an approximation of** $9\frac{1}{6} + 8\frac{1}{30} + 4\frac{1}{15} + 4\frac{1}{20} = \frac{67}{30}$**. This is the best ratio we have obtained.**

The maximum weight is 5 Let us consider the following basic cases:

- $L(5,5) = L(5,4) = 8$.
- $L(5,\{2,3,4,5\}) = L(5,\{2,3\}) = 5$
- $L(5,\{3,4,5\}) = 5$
- $L(5,\{4,5\}) = 5$
- $L(6,5) = 10$. First note that $L(6,5) \geq L(6,4) = 9$. To prove that $L(6,5) \neq 9$, let us consider the following illustrating example. We build a clique of 6 vertices of weight 1 each, each of these vertices is incident to a vertex of weight 5 and each pair of vertices of weight 1 is incident to a vertex of weight 4. No vertex of weight 1 can use color 5 because otherwise its neighbor of weight 5 cannot be colored. No matter how we color the vertices excluding the middle color, we will have a fragmentation combination for one of the vertices of weight 4. $L(6,5) = 10$, the algorithm uses First Fit and forbids colors 7, 8, 9 and 10 to vertices of weight 1.
- $L(6,\{2,3,4,5\}) = L(6,\{2,3,4\}) = 8$
- $L(6,\{3,4,5\}) = 6$

Let us now see the approximation we can obtain from the basic cases above.

We set $n_1 = \frac{M}{5}$, $n_2 = \frac{M}{20}$, $n_3 = \frac{M}{12}$, $n_4 = \frac{M}{16}$ and $n_5 = \frac{M}{25}$, and the densities $m_1 = 5$, $m_2 = 5$, $m_3 = 5$, $m_4 = 5$ and $m_5 = 5$ **we obtain an approximation of** $8\frac{1}{5} + 5\frac{1}{20} + 5\frac{1}{12} + 5\frac{1}{16} + 5\frac{1}{25} = \frac{667}{240}$**.** .

We set $n_1 = \frac{M}{6}$, $n_2 = \frac{M}{30}$, $n_3 = \frac{M}{20}$, $n_4 = \frac{M}{16}$ and $n_5 = \frac{M}{25}$, and the densities $m_1 = 6$, $m_2 = 6$, $m_3 = 6$, $m_4 = 5$ and $m_5 = 5$ **we obtain an approximation of** $10\frac{1}{6} + 8\frac{1}{30} + 6\frac{1}{20} + 5\frac{1}{16} + 5\frac{1}{25} = \frac{659}{240}$**. This is the best ratio we have obtained.**

The maximum weight is 6 Let us consider the following basic cases:

- $L(6,6) = L(6,5) = 10$
- $L(6,\{2,3,4,5,6\}) = L(6,\{2,3,4\}) = 8$
- $L(6,\{3,4,5,6\}) = L(6,\{4,5,6\}) = L(6,\{5,6\}) = 6$

Let us now see the approximation we can obtain from the basic cases above. We set $n_1 = \frac{M}{6}$, $n_2 = \frac{M}{30}$, $n_3 = \frac{M}{20}$, $n_4 = \frac{M}{16}$, $n_5 = \frac{M}{25}$ and $n_6 = \frac{1}{36}$ with densities $m_1 = m_2 = m_3 = m_4 = m_5 = m_6 = 6$ **we obtain an approximation of** $10\frac{1}{6} + 8\frac{1}{30} + 6\frac{1}{20} + 6\frac{1}{16} + 6\frac{1}{25} + 6\frac{1}{36} = \frac{603}{200}$**.**

□

The case where we have weights 1 and X

Theorem 8. *There is a* $2 - \frac{1}{W}$ *for SA in binary trees when the demands of the requests are in the set* $\{1, W\}$.

Proof. As for interval graphs in [18], there is an algorithm to find a $2 - \frac{1}{W}$-approximation whenever we have only two weights 1 and X in a chordal graph. We proceed as follows. We partition the colors into two blocks. The first block of width equal to M (the weight of the maximum clique) and the second is equal to a value $N = M - \frac{M}{W}$. We use First Fit in the reverse perfect elimination order (recall that when a vertex arrives in this order, its neighbors that arrived before form a clique). All vertices of weight one will be in the first block because otherwise the density will be bigger than M. When a request of size W arrives, we provision it in the first block and if it is not possible in the second block (not in the between). Let us suppose that a vertex v of weight W arrives and cannot be provisioned in neither of the two blocks; v cannot be provisioned in the second block means that it is all used by its neighbors (neighbors of v using block two have a total weight of $M - \frac{M}{W}$). Since v cannot use the first block, this means that it is fragmented by neighbors of v.

The minimum size of neighbors of r that can fragment block one is $\frac{M}{W}$ (all vertices of weight one separated by $W-1$ free colors). This implies that the density is bigger than M.

This algorithm can apply when we have two sizes and one of them is the multiple of the other.

□

6 Possible directions

To find approximation algorithms for interval coloring in chordal graphs the following directions might be helpful.

- Use the Buddy-decreasing-size algorithm that is equivalent to first fit for online coloring of chordal graphs. Some of the challenges in this direction is that using it as it is cannot give better than a $log(n)$-approximation; there is a tight example in [19]. In the tight example however all the vertices have the same weight which means that there is an exponential number of possible orders. Maybe some clever modification can give a better bound. The other challenge is to find a good bound for first fit in online coloring of chordal graphs with such modification. I did not find much work on the existing bound. All I know is that it uses at most $O(\log(n))\chi(G)$ [14].
- Use the boxing of jobs. The paper using this technique base it on the interval representation of the graph. Somehow, the job boxed have close starting and ending time. The problem here is that the reverse perfect elimination order does not give as much information as that given by the interval representation; it gives an idea about the neighbors of a vertex when it arrives but not on the ones coming after (using the same idea might result in boxing many jobs that are independent).
- Defining a 2-allocation for chordal graphs. The idea used by Gregov does not result in a 2-allocation. Examples can be designed where a color is shared by an unbounded number of vertices (a clique where every node is connected to a request of big size).

References

[1] Beauquier, B. *Communication dans les réseaux optiques par multiplexage en longueur d'onde.* PhD thesis, Université de Nice Sophia Antipolis, 2000.

[2] Buchsbaum, A. L., Efrat, A., Jain, S., Venkatasubramanian, S., and Yi, K. Restricted strip covering and the sensor cover problem. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia, PA, USA, 2007), SODA '07, Society for Industrial and Applied Mathematics, pp. 1056–1063.

[3] Buchsbaum, A. L., Karloff, H., Kenyon, C., Reingold, N., and Thorup, M. Opt versus load in dynamic storage allocation. *SIAM J. Comput. 33*, 3 (Mar. 2004), 632–646.

[4] Chlamtac, I., Ganz, A., and Karmi, G. Lightpath communications: an approach to high bandwidth optical wan's. *Communications, IEEE Transactions on 40*, 7 (Jul 1992), 1171–1182.

[5] Chrobak, Marek, M. On some packing problem related to dynamic storage allocation. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications 22*, 4 (1988), 487–499.

[6] Confessore, G., Dell'Olmo, P., and Giordani, S. An approximation result for the interval coloring problem on claw-free chordal graphs. *Discrete Appl. Math. 120*, 1-3 (Aug. 2002), 73–90.

[7] Gergov, J. Approximation algorithms for dynamic storage allocation. In *Algorithms — ESA '96*, J. Diaz and M. Serna, Eds., vol. 1136 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1996, pp. 52–61.

[8] Gergov, J. Algorithms for compile-time memory optimization. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia, PA, USA, 1999), SODA '99, Society for Industrial and Applied Mathematics, pp. 907–908.

[9] Gerstel, O., Jinno, M., Lord, A., and Yoo, S. Elastic optical networking: a new dawn for the optical layer? *Communications Magazine, IEEE 50*, 2 (February 2012), s12–s20.

[10] Goemans, M. X. An approximation algorithm for scheduling on three dedicated machines. *Discrete Applied Mathematics 61*, 1 (1995), 49 – 59.

[11] Golumbic, M. C. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 2004.

[12] Golumbic, M. C., Lipshteyn, M., and Stern, M. Representing edge intersection graphs of paths on degree 4 trees. *Discrete Mathematics 308*, 8 (2008), 1381 – 1387. Third European Conference on Combinatorics Graph Theory and Applications Third European Conference on Combinatorics.

[13] Huang, J., Chen, J., and Chen, S. A simple linear-time approximation algorithm for multi-processor job scheduling on four processors. In *Algorithms and Computation*, G. Goos, J. Hartmanis, J. van Leeuwen, D. Lee, and S.-H. Teng, Eds., vol. 1969 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000, pp. 60–71.

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

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

[16] Kierstead, H. A polynomial time approximation algorithm for dynamic storage allocation. *Discrete Mathematics 88*, 2–3 (1991), 231 – 237.

[17] Li, S., Leong, H., and Quek, S. New approximation algorithms for some dynamic storage allocation problems. In *Computing and Combinatorics*, K.-Y. Chwa and J. Munro, Eds., vol. 3106 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 339–348.

[18] Murthy, P. K., and Bhattacharyya, S. S. Approximation algorithms and heuristics for the dynamic storage allocation problem. Tech. rep., Univ. Maryland Inst. Adv. Comput. Studies, College Park, 1999.

[19] Pemmaraju, S. V., Penumatcha, S., and Raman, R. Approximating interval coloring and max-coloring in chordal graphs. *J. Exp. Algorithmics 10* (Dec. 2005).

[20] Shirazipourazad, S., Zhou, C., Derakhshandeh, Z., and Sen, A. On routing and spectrum allocation in spectrum-sliced optical networks. In *INFOCOM, 2013 Proceedings IEEE* (April 2013), pp. 385–389.

[21] Talebi, S., Bampis, E., Lucarelli, G., Katib, I., and Rouskas, G. N. Spectrum assignment in optical networks: A multiprocessor scheduling perspective. *J. Opt. Commun. Netw. 6*, 8 (Aug 2014), 754–763.

[22] Tarjan, R. E., and Yannakakis, M. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput. 13*, 3 (July 1984), 566–579.