On Spectrum Assignment in Elastic Optical Tree-Networks

Jean-Claude Bermond, Fatima Zahra Moataz

▶ **To cite this version:**

Jean-Claude Bermond, Fatima Zahra Moataz. On Spectrum Assignment in Elastic Optical Tree-Networks. [Research Report] Inria Sophia Antipolis; Université Nice Sophia Antipolis. 2015. hal-01116321v4

HAL Id: hal-01116321

https://hal.inria.fr/hal-01116321v4

Submitted on 5 Nov 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On Spectrum Assignment in Elastic Optical Tree-Networks*

Jean-Claude Bermond and 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. The efficiency and flexibility are, however, 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) [10] 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 EONs is referred to as Routing and Spectrum Assignment (RSA). In 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 request has to be contiguous (contiguity constraint), it has to be the same over all links of the routing path (continuity constraint) and requests with paths 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 The SA problem is a generalization of the well studied problem of Wavelength Assignment (WA) (WA is the special case of SA in which all requests have equal demands). 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

*This work has been partly supported by ANR project Stint under reference ANR-13-BS02-0007 and ANR program "Investments for the Future" under reference ANR-11-LABX-0031-01

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

tractable, particularly in path networks. Indeed, SA has been proved to be equivalent to other problems studied in the literature which we describe in details in Section 2. Using the results obtained for the equivalent problems, SA is NP-complete in paths even if the requests' demands do not exceed 2 slots [2]. Furthermore, SA is NP-complete in paths with 4 links and unidirectional rings with 3 links [24]. On the positive side, SA can be approximated within a factor of $2+\epsilon$ in paths, a factor of $4+\epsilon$ in rings, and a factor of $O(log(k))$ in binary trees where k is the number of requests [23].

Contribution In this report, we study the SA problem in trees. We focus on special cases where the tree is a star or a binary tree. By studying these special cases, we hope to gain more insight into the general problem in trees and design a constant-factor approximation algorithm or prove that such algorithm does not exist. We prove that SA is NP-hard in undirected stars of 3 links and in directed stars of 4 links, and show that in general stars it can be approximated within a factor of 4. Afterwards, we use the equivalence of SA with a graph coloring problem (interval coloring) to find constant-factor approximation algorithms for SA on binary trees with special demand profiles. Namely, we examine the cases where the demands are in a set $\{k, kX\}$ ($k, X \in \mathbb{N}^*$), in a set $\{kX, k(X+1)\}$ ($k, X \in \mathbb{N}^*$), or bounded by D. For the latter case, we give a general approximation when the demands are bounded by $D \in \mathbb{N}$ and then give better approximations for the cases where the demands are bounded by $D \in \{3, 4, 5, 6\}$.

This report is organized as follows. In Section 2, we formally define the SA problem and survey its relation to other problems and its complexity in path networks in particular. Afterwards, we present our results in stars and binary trees in Sections 3 and 4, respectively.

2 Problem statement and related problems

In this section, we first define the problem of Spectrum Assignment (SA) and then present some related problems and highlight their relation to SA. In the last subsection, we list the results implied by these relations for the complexity of SA in paths.

2.1 Spectrum Assignment

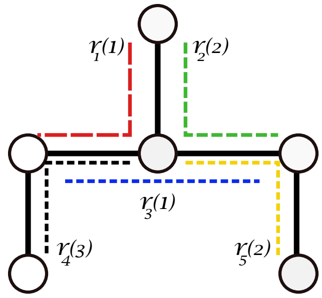
An instance $(\mathcal{N}, \mathcal{R})$ of the problem consists of a graph $\mathcal{N} = (N, L)$ and a set of requests $\mathcal{R}$. The graph $\mathcal{N}$ models an optical network with N as the set of nodes and L as the set of links. A request $r \in \mathcal{R}$ consists of a path $P(r)$ in $\mathcal{N}$ and a spectrum demand $d(r) \in \mathbb{N}$ (number of spectrum slots). We say that two requests $r, r' \in \mathcal{R}$ are *conflicting* if their paths $P(r)$ and $P(r')$ share a link. A spectrum assignment of $(\mathcal{N}, \mathcal{R})$ is a mapping f from $\mathcal{R}$ to $\mathbb{N}^*$ such that for every pair of conflicting requests $r, r' \in \mathcal{R}$, we have $\{f(r), \ldots, f(r) + d(r) - 1\} \cap \{f(r'), \ldots, f(r') + d(r') - 1\} = \emptyset$. We say that all the slots in $\{f(r), \ldots, f(r) + d(r) - 1\}$ are *occupied* by r. In this report, we consider slots as integers (which will be useful for the relation with colors in interval colorings); however other authors consider slots as intervals of unit length. In fact the set of slots $\{f(r), \ldots, f(r) + d(r) - 1\}$ corresponds to the spectrum interval $]f(r) - 1, f(r) + d(r) - 1]$ and we sometimes use them interchangeably.

The *span* of a spectrum assignment f, denoted $s(f)$, is the smallest integer s such that for each request $r \in \mathcal{R}$, $f(r) + d(r) - 1 \leq s$. The span of an instance $(\mathcal{N}, \mathcal{R})$, denoted by $s(\mathcal{N}, \mathcal{R})$ is the minimum of the spans over all possible spectrum assignments. We formulate the Spectrum Assignment problem as follows:

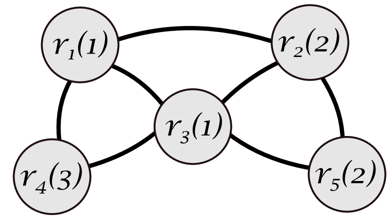
Problem 1 (Spectrum Assignment (SA))**.** *Given an instance* $(\mathcal{N}, \mathcal{R})$*, compute* $s(\mathcal{N}, \mathcal{R})$*.*

For an instance $(\mathcal{N}, \mathcal{R})$ of SA, the *load of a link* ℓ, denoted by $\pi(\ell)$, is the sum of the demands of the requests using ℓ and the *load of an instance*, denoted by $\Pi(\mathcal{N}, \mathcal{R})$, is the maximum load over all its links. It is straightforward that $\Pi(\mathcal{N}, \mathcal{R}) \leq s(\mathcal{N}, \mathcal{R})$. In the approximations we obtain for SA in this report, the span is usually upper bounded by a function of the maximum load.

The *greedy algorithm* for SA is an algorithm which assigns spectrum to requests ordered in a given order $r_1, \ldots, r_n$ such that a request r_i is assigned the smallest positive integer $g(r_i)$ such that $\{g(r_i), \ldots, g(r_i) +$



(a) Instance of SA in a tree



(b) Corresponding conflict graph

Figure 1: Example of the construction of the conflict graph

$d(i) - 1\} \cap \{g(r_j), \ldots, g(r_j) + d_j - 1\} = \emptyset$ for each r_j in $\{r_1, \ldots, r_{i-1}\}$ conflicting with r_i. We will use this algorithm many times in the rest of this report.

Figure 1a illustrates an instance of SA on a binary tree with 5 requests ($r_i(j)$ is request r_i with demand j). Note that the order of the requests with which the greedy algorithm is applied has a direct impact on the number of spectrum slots used. In the example of Figure 1a, applying the greedy algorithm in the order r_1, r_2, r_3, r_4, and then r_5 results in the use of 7 spectrum slots, while applying the algorithm in the order r_3, r_1, r_5, r_2, and then r_4 results in the use of only 5 spectrum slots: slots $\{1\}$, $\{2\}$, $\{2, 3\}$, $\{4, 5\}$, and $\{3, 4, 5\}$ for r_3, r_1, r_5, r_2, and r_4, respectively. The span of this instance is exactly 5, as the load is equal to 5 on the two links used by r_3.

2.2 Related problems

2.2.1 Scheduling Tasks on Multiprocessor Systems

It has been proved in [24] that SA in a network of k links can be reduced to the problem of Scheduling Tasks on Multiprocessor Systems (STMS) with k multiprocessors. In the STMS problem, we are given a set of n tasks and a set of identical processors, a processing time $d(j)$ and a prespecified set P_j of processors for each task j, $j \in \{1, \ldots, n\}$. The objective is to 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 P_j, and (3) each processor can work on at most one task at a time.

Given an instance $(\mathcal{N}, \mathcal{R})$ of SA, an instance of STMS is constructed as follows. For each link ℓ of $\mathcal{N}$, we associate a processor w_ℓ, and for each request r in $\mathcal{R}$ with path $P(r)$ and demand $d(r)$, we associate a task t_r with processing time $d(r)$ and a set of processors $\{w_\ell \mid \ell \in P(r)\}$. The makespan is then the span of the instance of SA.

Complexity of STMS Note that the relation above is only in one direction as there exist instances of STMS for which there is no corresponding instance of the SA problem. However for 3 processors we can associate to an instance of STMS an instance of SA in an unidirectional ring with 3 links (each processor being associated to one of the links). It has been shown in [15] that the problem of STMS is strongly NP-complete even if the number of used processors is at most 3. Using this result, it is proved in [24] that the SA problem is strongly NP-complete in an unidirectional ring with 3 links. On the positive side, it has been proved in [11] and [16] that STMS can be approximated within $\frac{7}{6}$ and 1.5 when the number of processors is 3 and 4, respectively. Theorem 1 follows from these approximations.

Theorem 1. *There are approximation algorithms with ratios $\frac{7}{6}$ and 1.5 for the Spectrum Assignment problem in networks with 3 and 4 links, respectively.*

2.2.2 Dynamic Storage Allocation

When the network is a path, the SA problem is equivalent to the problem of Dynamic Storage Allocation (DSA). In the DSA problem, we are given a set A of items to be stored, each $a \in A$ having size $d(a)$, an arrival time $\alpha(a)$, and a departure time $\beta(a)$ (with $\beta(a) > \alpha(a)$). A storage allocation for A is a function $f : A \to \mathbb{N}^*$ which associates to each item $a \in A$ a storage interval $I(a) =]f(a) - 1, \ldots, f(a) + d(a) - 1]$ such that for all $a, a' \in A$ with $a \neq a'$, if $]\alpha(a), \beta(a)] \cap]\alpha(a'), \beta(a')]$ is not empty, then $I(a) \cap I(a')$ is empty. The storage size used by a storage allocation f denoted by $s(f)$ is the smallest integer s such that for each item $a \in A$, $f(a) + d(a) - 1 \leq s$. The objective in DSA is to find a storage allocation which minimizes the used storage size.

If we consider the time interval as a path network and each of the items to be stored as a request, we can see the equivalence between the problem of SA in paths and the DSA problem. In more details, given an instance of SA on a path $(v_1, \ldots, v_k)$, we associate to each request r with demand $d(r)$, an item a_r of size $d(r)$. We also associate to each vertex v_i of the path network time i. Let v_i and v_j be the endvertices of the path $P(r)$ of the request r $(i < j)$, then we choose for the associated item a_r the arrival time $\alpha(a_r) = i$ and the departure time $\beta(a_r) = j$. The fact that two requests r and r' are conflicting corresponds to the fact that the time intervals $]\alpha(a_r), \beta(a_r)]$ and $]\alpha(a_{r'}), \beta(a_{r'})]$ intersect. Then a spectrum assignment with span γ corresponds to a storage allocation using a storage size γ. Conversely using the opposite transformation we can associate to an instance of DSA an instance of SA on a path.

Complexity of DSA The problem of DSA has been extensively studied. It has been proved that DSA is strongly NP-complete, even when restricted to instances where the storage size of all items is in $\{1, 2\}$. The proof of NP-completeness is by reduction from the 3-PARTITION problem and can be found in the appendix of [2]. On the positive side, many approximation algorithms have been proposed to solve DSA. The first proposed algorithms are based on a greedy algorithm called First Fit (FF) and its performance for online coloring of interval graphs. The relation between online coloring of interval graphs and dynamic storage allocation can be found in [5]. Using FF a linear approximation was proved in [17] and a ratio of 6 is given in [18]. Gergov has adopted another approach not using FF, yielding an approximation of 5 and 3 sequentially in [8] and [9]. In his approach, Gergov defines and uses a 2-allocation which is a storage allocation where two items but not three are allowed to overlap. A better approximation has been achieved in [3] where the authors use the idea of boxing items to design a $2+\epsilon$-approximation algorithm. Better approximations were achieved for DSA with restricted item sizes. In [19], the authors present a $\frac{4}{3}$-approximation algorithm when the maximum size is 2, and a 1.7-approximation algorithm when the maximum size is 3. In [20], it is proved that for instances with sizes of 1 and X, an approximation of ratio $2 - \frac{1}{X}$ can be guaranteed. All these results established for DSA apply, by equivalence, to SA in paths.

2.2.3 Interval Coloring

As pointed out in [23], the problem of SA is also equivalent to a graph coloring problem called Interval Coloring (IC). An interval coloring or a contiguous coloring [12] of a vertex-weighted graph $(G = (V, E), w)$ is a mapping $f : V \to \mathbb{N}^*$ such that for every $v, v' \in V$, if $(v, v') \in E$ then $\{f(v), \ldots, f(v) + w(v) - 1\} \cap \{f(v'), \ldots, f(v') + w(v') - 1\} = \emptyset$. The number of colors used by an interval coloring f, denoted by $\chi_f(G, w)$ is the smallest integer s such that for each vertex $v \in V$, $f(v) + w(v) - 1 \leq s$. The interval chromatic number of a weighted graph (G, w), denoted by $\chi(G, w)$, is the smallest number of colors needed to color the vertices with intervals, i.e. it is the minimum of $\chi_f(G, w)$ among all possible interval colorings f of (G, w). The interval coloring problem is defined as follows.

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

To see the equivalence between SA and IC we do the following. For an instance $(\mathcal{N}, \mathcal{R})$ of SA, we create a weighted graph $(G = (V, E), w)$ modeling the dependency between the different requests called *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' are conflicting. The weight $w(v_r)$ of each vertex v_r is equal to the demand of the corresponding request r (i.e. $w(v_r) = d(r)$). Figure 1b ($r_i(j)$ is vertex r_i with weight j) shows the conflict graph associated to the SA instance of Figure 1a.

If $(\mathcal{N}, \mathcal{R})$ is an instance of SA and (G, w) is its conflict graph, then finding a spectrum assignment of $(\mathcal{N}, \mathcal{R})$ is equivalent to finding an interval coloring of (G, w) and $s(\mathcal{N}, \mathcal{R}) = \chi(G, w)$.

Complexity of IC The problem of IC has been introduced in [12] where its relation to other problem such as DSA has been highlighted. It has also been proved in [12] that IC is equivalent to the problem of finding, for a vertex-weighted graph, an acyclic orientation which minimizes the weight of the longest path, where the length of a path is the sum of the weights of its vertices. The complexity of DSA implies that IC is strongly NP-complete in interval graphs. IC is also strongly NP-complete in proper interval graphs [22]. On the positive side, IC is polynomial in comparability graphs [12] and can be approximated within a factor of $2 + \epsilon$ in interval graphs [3], a factor of 2 for proper interval graphs [22] and claw-free chordal graphs [6], and a factor of $O(log(n))$ in chordal graphs where n is the number of vertices [21].

Since the conflict graph associated to an instance of SA in a path is an interval graph (and vice versa), all the results established for IC in interval graphs apply to SA in paths. In section 4, we will use the fact that the conflict graph associated to a binary tree is a chordal graph to obtain results for SA in binary trees using interval coloring of chordal graphs.

2.3 Spectrum Assignment in paths

The results deduced from the equivalence to the problems defined above can be summarized as follows for path networks.

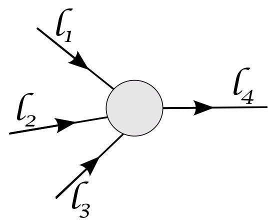
- With respect to the number of links, SA is NP-complete in path networks with 4 links and polynomial in paths with at most 3 links [24], and it can be approximated within a factor of 1.5 in paths with 4 or 5 links [24].
- With respect to the demands, SA is strongly NP-complete even if the requests have demands in the set $\{1, 2\}$ [2]. It can be approximated within a factor of $\frac{4}{3}$ and a factor of 1, 7 when the maximum demand is 2 and 3, respectively [19]. It also can be approximated within a factor of $2 - \frac{1}{X}$ when the demands are in the set $\{1, X\}$ [20].
- In general, SA in paths can be approximated in paths within a factor of $2 + \epsilon$ [3] and it can be approximated within a factor of 2 when the paths of the requests are such that no path is strictly included in another [22].

3 Spectrum Assignment in stars: hardness 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]. We prove in this section that SA is not only NP-complete 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.

Theorem 2. *The problem of Spectrum Assignment is strongly NP-complete in undirected stars with 3 links.*

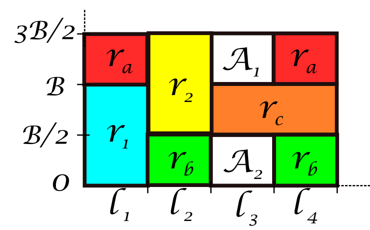
Proof. It was shown in [24] that the SA problem is strongly NP-complete in a 3-link unidirectional ring (see subsection 2.2.1). Let us consider an instance of SA in a 3-link ring $C = (l_1, l_2, l_3)$ with a request set $\mathcal{R}$. Let us build a star S with three links l'_1, l'_2 and l'_3, and a 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 in C is $P(r) = l_i$, $i \in \{1, 2, 3\}$, then the path of r' in S is $P(r') = l'_i$, and if $P(r) = l_i l_j$, then $P(r') = l'_i l'_j$. Solving SA in $(C, \mathcal{R})$ is equivalent to solving SA in $(S, \mathcal{R}')$. □



(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
$r_3^i,\ _{1 \leq i \leq k}$	a_i	l_3

(b) Requests $\mathcal{R}$



(c) Spectrum assignment

Figure 2: Reduction from 2-PARTITION to SA in a directed star

Theorem 3. *The problem of Spectrum Assignment is weakly NP-complete in directed stars with 3 ingoing links and one outgoing link or 3 outgoing links and one ingoing link.*

Proof. The proof is by reduction from the 2-PARTITION problem. In the 2-PARTITION problem, we are given a set A of k integers $a_1, a_2, \ldots, a_k$ such that $B = \sum_{j=1}^{k} a_j$ and the objective is to decide whether A can be partitioned into two disjoint sets A_1 and A_2 such that $\sum_{a_j \in A_1} a_j = \sum_{a_j \in A_2} a_j$.

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 the Spectrum Assignment problem in a 4-link directed star network S (Figure 2a) and a set of requests $\mathcal{R}$. The star S has 3 ingoing links l_1,l_2, and l_3 and one outgoing link l_4. The set of requests $\mathcal{R}$ consists of the requests presented in Figure 2b: requests r_a, r_b, r_c, r_1, and r_2 and for every integer a_i in the set A, a request r_3^i with demand a_i and 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 2c. 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 size $\frac{B}{2}$. □

Theorem 4. *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 all of these cases, the span is equal to the maximum 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 any conflicting requests use the same link 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. Indeed let A_{i3} be the sum of the demands of the requests using l_i and l_3 for $i \in \{1, 2\}$ and let A_i be the sum of the demands of the requests using only link l_i for $i \in \{1, 2, 3\}$. Then the span of the spectrum used on link l_1 is $A_{13} + A_1 = \pi(l_1)$, and that on links l_2 and l_3 is equal to $\max(A_3^1 + A_3, A^2) + A_3^2 = \max(\pi(l_2), \pi(l_3))$, and so the span of this spectrum assignment is equal to the maximum load of the instance.

- When the star has 2 ingoing links l_1 and l_2 and 2 outgoing links l_3 and l_4, let A_{ij} be the sum of the demands of the requests using l_i and l_j for $i \in \{1,2\}$ and $j \in \{3,4\}$ and let A_i be the sum of the demands of the requests using only link l_i for $i \in \{1,2,3,4\}$. First, the requests using l_1 and l_3 and the requests using l_2 and l_4 are assigned with the greedy algorithm; the span of the spectrum used on links l_1 and l_3 is equal to A_{13}, and the span of the spectrum used on links l_2 and l_4 is equal to A_{24}. Afterwards, the requests using only one link are assigned; the span of the spectrum used on links l_1, l_2, l_3, and l_4 is $A_{13}+A_1$, $A_{24}+A_2$, $A_{13}+A_3$, and $A_{24}+A_4$, respectively. Finally, the requests using l_1 and l_4 and the requests using l_2 and l_3 are assigned with the greedy algorithm; the span of the spectrum used on links l_1 and l_4 is equal to $\max(A_3^1 + A_1, A_4^2 + A_4) + A_4^1 = \max(\pi(l_1), \pi(l_4))$, and the span of the spectrum used on links l_2 and l_3 is equal to $\max(A_3^1 + A_3, A_4^2 + A_2) + A_3^2 = \max(\pi(l_2), \pi(l_3))$. This means that the span of this spectrum assignment is equal to the maximum load of the instance.

□

Theorem 5. *Let $(\mathcal{N}, \mathcal{R})$ be an instance of SA. If the length of the paths associated to the requests in $\mathcal{R}$ is at most α, then the greedy algorithm gives a 2α-approximation for the Spectrum Assignment problem. In particular there is a 4-approximation polynomial-time algorithm for the Spectrum Assignment problem in stars.*

Proof. Let $(\mathcal{N}, \mathcal{R})$ be an instance of SA. Let the requests of $\mathcal{R}$ be ordered in the non-increasing order of demands $r_1, r_2, \dots, r_q$ (i.e., $d(r_1) \geq d(r_2) \geq \dots \geq d(r_q)$). Let Π be the maximum load. We will use at most $2\alpha\Pi$ slots to allocate spectrum to the requests of $\mathcal{R}$. Suppose that we have already assigned spectrum to the first requests r_j, $j < i$ with the span $2\alpha\Pi$ and consider the request r_i with demand $d(r_i) = d$. For each link l of the path $P(r_i)$, let $R_i(l)$ be the set of requests already assigned conflicting with r_i on the link l. As the load of the link l is at most Π, the sum of the demands of the requests of $R_i(l)$ is at most $\Pi - d$. Since each of these requests has demand at least d, we have at most $\frac{\Pi - d}{d}$ requests in $R_i(l)$. This implies that the path $P(r_i)$ has at most $\frac{\alpha(\Pi-d)}{d}$ requests conflicting with r_i which have been already assigned spectrum. Consider the slots not occupied by these requests (available slots). If there exists an interval of d or more available slots below these requests or between two requests, we can assign to request r_i the first such interval.

Otherwise, between slot 1 and the first slot occupied by the conflicting requests and between the last slot occupied by a request and the first slot of the next request there are at most $d-1$ available slots. As there are at most $\frac{\alpha(\Pi-d)}{d}$ requests conflicting with r_i, we have at most $\frac{\alpha(\Pi-d)}{d}$ such intervals. As the requests in $R_i(l)$ occupy at most $(\Pi - d)$ slots, we have at most $\alpha(\Pi - d)$ slots occupied by the conflicting requests and at most $\frac{\alpha(\Pi-d)}{d}(d-1)$ slots available where we cannot provision r_i. Altogether, we have a number of non usable slots equal to $\alpha(\Pi - d) + \frac{\alpha(\Pi-d)}{d}(d-1) = 2\alpha\Pi - 2\alpha d - \frac{\alpha(\Pi-d)}{d}$. So, there is an interval of $2\alpha d + \frac{\alpha(\Pi-d)}{d}$ available contiguous slots above all the requests conflicting with r_i. Then, we can allocate to r_i d contiguous slots in this interval. In particular, in the case of stars where $\alpha = 2$, we obtain a 4-approximation. □

This approximation algorithm for stars together with the $2+\epsilon$-approximation algorithm for paths presented in [3], imply a constant factor approximation for tree networks which are spiders. A *spider* is a tree with one vertex of degree at least 3 and all others with degree at most 2.

Theorem 6. *There is a $(6+\epsilon)$-approximation for the Spectrum Assignment problem in spiders.*

Proof. Let $(S, \mathcal{R})$ be an instance of SA in a spider tree. Let v be the vertex of S of degree 3 and let S' be the star induced by v and all the vertices of S which are at distance 1 from v. We first consider the set of requests R_1 using an edge of S'. For a request r of R_1 we associate the restricted request r' in S' with path the subpath of r restricted to S' and same demand as r. We use the 4-approximation presented in Theorem 5, to allocate spectrum to these restricted requests using the star S'. That induces a spectrum assignment to the requests of R_1, as two such requests intersect if and only if their restricted requests intersect. Now we consider the other set of requests R_2 which are included in some path P of the paths of $S \setminus S'$. We use the $2+\epsilon$-approximation algorithm to allocate spectrum to the requests of R_2 using P. We can use the same spectrum range for two different paths as they have no link in common. So altogether we get a $(6+\epsilon)$-approximation algorithm. □

4 Spectrum Assignment in binary trees

The SA problem in binary trees (trees in which each node has degree at most three) has been studied in [23]. It has been proved that SA can be approximated within a ratio of $O(log(k))$ where k is the number of requests. The proof is based on the equivalence between SA and the problem of Interval Coloring (IC). In fact, the conflict graph of an instance of SA in a binary tree corresponds to an edge intersection graph of paths in a binary tree. These graphs have been proved to be chordal graphs in [13, 14]. Using the problem of interval coloring in chordal graphs, we give in this section some constant-factor approximation algorithms for the problem of spectrum assignment in binary trees with special demand profiles. Namely, we examine the cases where the demands are in a set $\{k, kX\}$ $(k, X \in \mathbb{N}^*)$, in a set $\{kX, k(X+1)\}$ $(k, X \in \mathbb{N}^*)$, or bounded by D. For the latter case, we give a general approximation when the demands are bounded by $D \in \mathbb{N}$ and then give better approximations for the cases where the demands are bounded by $D \in \{3, 4, 5, 6\}$. It is important to recall here that even if the network is a path and the demands are bounded by 2, SA is strongly NP-complete. We first start by giving some definitions and then we state our results.

4.1 Definitions

A chord of a cycle C in a graph is an edge of the graph connecting two vertices that are not adjacent in C. A graph G is chordal if every cycle of G with at least 4 vertices has a chord. One important property of chordal graphs is their perfect elimination order. The *perfect elimination order (PEO)* of a graph is an ordering $x_1, x_2, \ldots, x_n$ of the vertices of the graph such that for $i = 1, \ldots, n-1$, the neighbors of x_i in $G[\{x_{i+1}, \ldots, x_n\}]$[1] form a clique. It is well known that a graph is chordal if and only if it has a perfect elimination order. Paper [25] 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 use *the reverse perfect elimination order (RPEO)* in the design of some algorithms. 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. Another tool we will be using is the *greedy algorithm* for IC. Defined similarly to the greedy algorithm for SA, the greedy algorithm for IC, also called the *First Fit algorithm* (FF) is an algorithm which assigns colors to vertices in a given order $v_1, \ldots, v_n$ such that a vertex v_i is assigned the smallest positive integer $g(v_i)$ 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 $\{v_1, \ldots, v_{i-1}\}$ which is adjacent to v_i.

In a weighted graph $(G = (V, E), w)$, we define the weight of a subset $S \subseteq V$ to be the quantity $w(S) = \sum_{v \in S} w(v)$. The maximum weighted clique is a clique with the biggest weight. The density of $(G = (V, E), w)$ is the weight of the maximum weighted clique and is denoted by $\Delta(G, w)$. It is straightforward that $\Delta(G, w) \leq \chi(G, w)$, where $\chi(G, w)$ is the interval chromatic number of (G, w).

In the remainder of this section, we present our results for SA in binary trees with bounded demands as corollaries after proving theorems for IC in weighted chordal graphs with bounded weights. We note here that even though every conflict graph of an instance of SA in a binary tree is a chordal graph, the opposite is not true [13].

4.2 Demands k and kX

In this section, we present an approximation algorithm for the SA problem when the demand of each request is either k or kX, with $k, X \in \mathbb{N}^*$. We start by proving the following theorem for interval coloring in chordal graphs.

Theorem 7. *Let (G, w) be a weighted chordal graph with weights in the set $\{1, X\}$. There exists a polynomial-time algorithm that finds an interval coloring of (G, w) with $2\Delta(G, w) - \lfloor \frac{\Delta(G,w)}{X} \rfloor$ colors.*

[1] For $S \subseteq V$, we define $G[S]$ as the subgraph of G induced by the vertices of S, i.e. the subgraph of G containing the vertices of S and all the edges of G which have both endpoints in S.

Proof. It has been proved in [20] that there is an algorithm to find a $(2 - \frac{1}{X})$-approximation for the problem of interval coloring in interval graphs whenever there are only two weights 1 and X. We generalize this algorithm for chordal graphs as follows.

Let (G, w) be a weighted chordal graph with weights in $\{1, X\}$ and let $\Delta = \Delta(G, w)$ be its density. We will use $2\Delta - \lfloor \frac{\Delta}{X} \rfloor$ colors to color (G, w) as follows. We partition the colors into two sets. The first set S_1 contains colors from 1 to Δ and the second set S_2 contains colors from $\Delta + 1$ to $2\Delta - \lfloor \frac{\Delta}{X} \rfloor$.

We order the vertices of G in the reverse perfect elimination order. Let $v_1, \dots, v_n$ be the obtained ordering. Recall that the neighbors of v_i in $\{v_1, \dots, v_{i-1}\}$ form a clique in the graph induced by $\{v_1, \dots, v_{i-1}\}$. We use the greedy algorithm to assign colors to the vertices in this order with the additional property that colors assigned to a vertex are either included in S_1 or S_2 (we cannot use colors from both sets). We prove that with this algorithm, all vertices will be assigned colors in S_1 or S_2.

- All vertices of weight 1 will have a color in S_1. In fact, if a vertex v_i of weight 1 cannot be assigned a color in S_1, then its neighbors in $\{v_1, \dots, v_{i-1}\}$ occupy all colors of S_1. This implies that v_i and its neighbors in $\{v_1, \dots, v_{i-1}\}$form a clique of size $\Delta + 1$ a contradiction.

- For vertices of weight X, suppose that there is a vertex v_j of weight X to which we cannot assign colors neither in S_1 nor in S_2. The minimum number of colors used in S_1 that can make it not possible to color v_j with colors from S_1 is $\lfloor \frac{\Delta}{X} \rfloor$ ($X - 1$ free colors then 1 occupied color, then $X - 1$ free colors and 1 occupied color ...). The weight of the neighbors of v_j in $\{v_1, \dots, v_{j-1}\}$ which use colors in S_1 is at least $\lfloor \frac{\Delta}{X} \rfloor$. Since we cannot assign colors from S_2 to v_j and knowing that only vertices of the same weight X use colors from S_2 with the greedy algorithm, we deduce that the sum of the weights of the neighbors of v_j in $\{v_1, \dots, v_{j-1}\}$ which use colors in S_2 is at least $|S_2| - (X - 1)$. So v_j and its neighbors form a clique of size $X + \lfloor \frac{\Delta}{X} \rfloor + |S_2| - (X - 1) \geq \Delta + 1$ as $|S_2| = \Delta - \lfloor \frac{\Delta}{X} \rfloor$. This implies that the density of G is at least $\Delta + 1$, which is not possible.

□

Corollary 1. *Let (G, w) be a weighted chordal graph with weights in the set $\{k, kX\}$. There exists a polynomial time algorithm that finds an interval coloring of (G, w) with $2\Delta(G, w) - k\lfloor \frac{\Delta(G,w)}{kX} \rfloor$ colors.*

Proof. Note that to color a graph (G, w) with weights in $\{k, kX\}$, we can transform it to a graph (G, w') with weights in $\{1, X\}$, color (G, w') and then transform the colors we found into intervals of colors of size k. The number of colors used for G will be then at most k times the number of colors used for G'. Note also that the density of (G, w) is k times the density of (G, w'). By Theorem 1 we can color (G, w') with $2\Delta(G, w') - \lfloor \frac{\Delta(G,w')}{X} \rfloor$ colors and so we can color (G, w) $2k\Delta(G, w') - k\lfloor \frac{\Delta(G,w')}{X} \rfloor = 2\Delta(G, w) - k\lfloor \frac{\Delta(G,w)}{kX} \rfloor$ colors.

□

Remark : This implies that when the weights are in the set $\{k, kX\}$, we can color the graph with less than $(2 - \frac{1}{X})\Delta(G, w) + k$ colors.

Thanks to Corollary 1, we can deduce the following corollary.

Corollary 2. *Let $\mathcal{I}$ be an instance of SA in a binary tree such that the demands of requests are in the set $\{k, kX\}$ and the span of $\mathcal{I}$ is $\mathcal{OPT}$. There is a polynomial-time algorithm that finds a spectrum assignment for $\mathcal{I}$ with span less than $(2 - \frac{1}{X})\mathcal{OPT} + k$.*

Now, in what follows of this subsection, we find a lower bound on the number of colors that can be used to find an interval coloring of a weighted chordal graph with weights in $\{k, kX\}$.

Theorem 8. *There exists a family of weighted graphs $(G_m)_{m \in \mathbb{N}^*}$, with weights in the set $\{k, kX\}$, for which the ratio between the interval chromatic number and the density tends to $2 - \frac{1}{X}$ when m tends to infinity.*

Proof. For $m > 0$, we build the weighted graph G_m of density $k(mX^2 + 1)$ as follows.

- $mX^2 + 1$ vertices of weight k each forming a "big" clique.
- For each subset S of $mX + 1$ vertices of the big clique, we add $m(X-1)$ new vertices of weight kX each. These vertices form a clique with the vertices of S.

In any contiguous coloring of G_m, there exists an integer λ in $\{0, \ldots, kX-1\}$ such that the "big" clique uses $mX+1$ colors congruent to λ modulo kX. Suppose that this is not true and that the big clique uses for each integer i in $\{0, \ldots, kX-1\}$ at most mX colors which are congruent to i modulo kX. This means that the number of colors used is at most kmX^2. This is not possible since this maximum clique has weight $k(mX^2+1)$. Let S be a subset of $mX+1$ vertices of the big clique using colors that are congruent to λ modulo kX. Vertices of S form a clique with $m(X-1)$ vertices of weight kX. Each of these vertices uses a color congruent to λ modulo kX. In total, $m(2X-1)+1$ colors which are congruent to λ are used. This means that the total number of colors used is at least $kmX(2X-1)+1$. The ratio between the chromatic number and the density is then at least $\frac{kmX(2X-1)+1}{k(mX^2+1)} = 2 - \frac{1}{X} - \frac{(2k-1)X-k}{kX(mX^2+1)}$. When m goes to infinity, this ratio goes to $2 - \frac{1}{X}$.

□

4.3 Demands kX and $k(X+1)$

In this section, we present an approximation algorithm for the SA problem when the demand of each request is either kX or $k(X+1)$. We start by proving the following theorem for interval coloring in chordal graphs.

Theorem 9. *Let (G, w) be a weighted chordal graph with weights in $\{kX, k(X+1)\}$. There is a polynomial time algorithm to color G with at most $\frac{X+1}{X}\Delta(G, w)$ colors.*

Proof. Let (G, w) be a weighted chordal graph with weights in $\{kX, k(X+1)\}$. Let $m = \lfloor \frac{\Delta(G,w)}{kX} \rfloor$, we prove that we can color (G, w) with $k(X+1)m$ colors. We partition the set of colors $\{1, \ldots, k(X+1)m\}$ into m contiguous intervals I_i, $1 \leq i \leq m$ of size $k(X+1)$ each. Let us order the vertices of (G, w) in the RPEO order. We use the greedy algorithm to color the vertices in this order using for each vertex colors from exactly one interval I_i, $1 \leq i \leq m$. Suppose that we cannot color some vertex v_j, this means that each interval I_i, $1 \leq i \leq m$, contains a neighbor of v_j with weight at least kX (recall that the weights are either kX or $k(X+1)$). Since the neighbors of v_j which appear first in the RPEO form a clique with v_j, we have a clique of weight at least $mkX + kX > \Delta(G, w)$ which is not possible. Therefore we can color all the vertices.

□

Theorem 9 implies the following corollary.

Corollary 3. *There is a $\frac{X+1}{X}$-approximation algorithm for the Spectrum Assignment problem in binary trees when the demands of the requests are in the set $\{kX, k(X+1)\}$.*

4.4 Maximum demand D

In this section, we present an approximation algorithm for the SA problem when the maximum demand is D.

Theorem 10. *Let (G, w) be a weighted chordal graph with maximum weight W. There is a polynomial time algorithm that finds an interval coloring of (G, w) with at most $2\log_2(W)\Delta(G, w)$ colors.*

Proof. The proof is inspired from that of [21] which proposes a $O(log_2(n))$-approximation where n is the number of vertices.

Let (G, w) be a vertex-weighted chordal graph with maximum weight W. Let us partition the set of vertices V into k subsets S_i, $i \in \{1, \ldots, k\}$ such that for each vertex $v \in S_i$, $w(v) \in [a_i, b_i]$, with $a_1 = 1$, $a_{i+1} = b_i + 1$ and $b_k = W$. We first ignore the weights and optimally color each graph G_i induced by the subset S_i. As the graphs are chordal, we can color the vertices of G_i with $\omega(G_i)$ colors where $\omega(G_i)$

is the clique number of G_i. Afterwards, we replace the color of each vertex $v \in G_i$ by an interval of $w(v)$ colors. This way, we obtain an interval coloring of G_i with at most $b_i\omega(G_i)$ colors. Therefore, the vertices of G can be colored with c colors where $c = \sum_{i=1}^{k} b_i\omega(G_i)$. Note that $a_i\omega(G_i) \leq \Delta(G, w)$ which implies that $c \leq \sum_{i=1}^{k} \frac{b_i}{a_i}\Delta(G, w)$.

Let us choose $b_i = 2a_i$ for $i < k$. We will have then $a_i = 2^i - 1$ for $i \leq k$ and $b_i = 2^{i+1} - 2$ for $i < k$. If $2^h \leq W \leq 2^{h+1} - 2$, then we choose $k = h = \lfloor \log_2(W) \rfloor$ and we will have $c \leq 2\lfloor \log_2(W) \rfloor \Delta(G, w)$. If $W = 2^{h+1} - 1$, then we choose $k = h + 1$. In this case, we have $b_k = W = a_k = 2^{h+1} - 1$ and $c \leq (2h + 1)\Delta(G, w)$. Since $2^{2h+1} \leq (2^{h+1} - 1)^2$, we have $2h + 1 \leq 2 \log_2(W)$ and therefore we always have $c \leq 2 \log_2(W)\Delta(G, w)$.

Note that we can obtain slightly better approximations by using other ratios for $\frac{b_i}{a_i}$. If $\frac{b_i}{a_i} = x$, we obtain an $x \log_x(W)$ approximation. For $x = 3$, we obtain a $3 \log_3(W) \simeq 1.893 \log_2(W)$ approximation. The best ratio value we can take is $x = e$ (which minimizes the function $x \log_x(W)$); for this value we obtain an $e \ln(W) \simeq 1.884 \log_2(W)$ approximation. □

Theorem 10 implies the following corollary.

Corollary 4. *There is a $2 \log_2(D)$-approximation for the Spectrum Assignment problem in binary trees where D is the maximum demand.*

4.5 Maximum demand at most 6

In the previous subsection, an approximation algorithm for the SA problem in binary trees where the maximum demand is at most D has been presented. This approximation is achieved by partitioning the requests into subsets of close demands. This technique is used not only in binary trees but also in general graphs as a heuristic [26]. In what follows, we use different techniques to find better approximations for SA in binary trees for some given values of the maximum demand D. The techniques we use were introduced in [19] to approximate DSA. Results in [19] 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 by 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.

We first prove the following theorem for interval coloring.

Theorem 11. *Let (G, w) be a weighted chordal graph. There are polynomial-time algorithms which find an interval coloring of (G, w) with at most $\frac{3}{2}\Delta(G, w) + \frac{1}{2}$, $\frac{19}{10}\Delta(G, w) + \frac{8}{5}$, $\frac{59}{27}\Delta(G, w) + \frac{67}{27}$, $\frac{859}{336}\Delta(G, w) + \frac{229}{56}$ and $\frac{287}{100}\Delta(G, w) + \frac{885}{200}$ colors when the maximum weight is bounded by 2, 3, 4, 5 and 6, respectively.*

Proof. As in the previous sections, $\Delta(G, w)$ refers to the density of the weighted graph (G, w) and will be abbreviated in this proof to Δ.

Let $\mathcal{C}(d, S)$ denote the set of instances of IC in which the graph is chordal, the density is at most d and the weights are in the set S. Let $c(d, S)$ denote the smallest integer α such that for each instance of $\mathcal{C}(d, S)$, there is an interval coloring with at most α colors (if such α exists).

We present first the general approach to solve the problem for any maximum weight W before presenting the cases $W \in \{3, 4, 5, 6\}$ in details.

General Approach Let (G, w) be a weighted chordal graph with maximum weight W. To color the graph G, we proceed in two phases as follows.

- *Partitioning the vertices into multi-level blocks:* in this phase, the vertices are partitioned into blocks. We will have for each $i \in \{1, \ldots, W\}$, a set $\mathcal{B}_i$ of n_i level-i blocks $B_i^1, \ldots, B_i^{n_i}$ each of density d_i. We order the blocks in the lexicographic order: block B_i^j is before block $B_{i'}^{j'}$ if $i < i'$ or $i = i'$ and $j < j'$.

Our algorithm consists in considering successively the vertices in the RPEO order and assigning a new vertex v to the first available block (in the block's order). In more details, we assign a vertex v to a block B if the weight of the clique induced by v and its neighbors in B does not exceed the density of the block. The vertex v and its neighbors in B indeed form a clique since the graph is chordal and we consider the vertices in the RPEO order.

We will choose the parameters d_i and n_i (see details after) in such a way that the following property is satisfied:

Property *: Each vertex of weight i is assigned to some block in the set $\mathcal{B}_l$ such that $l \leq i$.

In particular, this means that at the end of the algorithm each vertex is assigned to some block.

- *Solving the problem of interval coloring for each block:* in this second phase, the vertices of each block of $\mathcal{B}_i$ are colored using an algorithm to solve instances with density d_i and weights in $S_i = \{i, \dots, W\}$ (the possible weights of the vertices in B_i^j). Note that the vertices of a block of $\mathcal{B}_i$ induce a graph which belongs to $\mathcal{C}(d_i, S_i)$. The algorithm we use is designed to use no more than $c(d_i, S_i)$ colors.

Therefore, the total number of colors used to color the whole graph is at most

$$\sum_{i=1}^{W} n_i c(d_i, \{i, \dots, W\})$$

The total number of colors depends on n_i and d_i. In fact, we will proceed as follows. For a chosen set of values of the densities d_i, we will choose the smallest possible n_i such that Property* is satisfied. Afterwards, we will compute $c(d_i, \{i, \dots, W\})$ and therefore the total number of colors for the chosen values of d_i. We will do this for many values of the densities d_i and keep the set of values which minimize the total number of colors.

Choice of the n_i Note that, if for some i, $d_i < i$, then $n_i = 0$ as a block of $\mathcal{B}_i$ cannot be used to assign a vertex of weight $\geq i$ (recall that the vertices of weight $< i$ are by Property * all assigned to blocks of $\mathcal{B}_l$ with $l < i$). So, in the following claims, we suppose $d_i \geq i$ for all i.

Claim 1. $n_1 = \left\lceil \frac{\Delta}{d_1} \right\rceil$

Proof. Suppose that a vertex v of weight 1 cannot be assigned to any block of $\mathcal{B}_1$. This means that, for each block B of $\mathcal{B}_1$, vertex v and its neighbors in B form a clique of size $> d_1$ and so the weight of the neighbors of v in B is at least d_1. This implies that the weight of the neighbors of v in all of the blocks in $\mathcal{B}_1$ is at least $n_1 d_1$. Since we are considering the vertices in the RPEO, this implies that the clique induced by v and its neighbors in $\mathcal{B}_1$ is of weight $n_1 d_1 + 1$ which exceeds Δ for $n_1 = \left\lceil \frac{\Delta}{d_1} \right\rceil$. This is not possible. □

Claim 2. $n_2 = \left\lceil \frac{\Delta - 1 - n_1(d_1 - 1)}{\Delta_2^2} \right\rceil$ *where* $\Delta_2^2 = \max\{2, d_2 - 1\}$.

Proof. Suppose that a vertex v of weight 2 cannot be assigned to any block of $\mathcal{B}_1$ or $\mathcal{B}_2$. This means that, for each block B of $\mathcal{B}_1$ (resp. $\mathcal{B}_2$), vertex v and its neighbors in B form a clique of size $> d_1$ (resp. $> d_2$) and so the weight of the neighbors of v in B is at least $d_1 - 1$ (resp. $d_2 - 1$). However, if $d_2 = 2$, as all the vertices of weight 1 are assigned to blocks of $\mathcal{B}_1$, v has necessarily one neighbor of weight 2 in each block of $\mathcal{B}_2$. Therefore, if we let $\Delta_2^2 = \max\{2, d_2 - 1\}$, the clique induced by v and its neighbors in the RPEO has a weight at least $n_1(d_1 - 1) + n_2 \Delta_2^2 + 2$ which exceeds Δ for $n_2 = \left\lceil \frac{\Delta - 1 - n_1(d_1 - 1)}{\Delta_2^2} \right\rceil$. □

Example: Consider the case W=2 and let $d_1 = 2$ $d_2 = 2$. Applying the formula we get $n_1 = \lceil \frac{\Delta}{2} \rceil$ and $n_2 = \lceil \frac{\Delta - 1 - n_1}{2} \rceil$. Using the fact that $c(2, \{1, 2\}) = c(2, \{2\}) = 2$ the number of colors is $2n_1 + 2n_2$ that is $6p$ for $\Delta = 4p$; $6p + 2$ for $\Delta = 4p + 1$ and for $\Delta = 4p + 2$ and $6p + 4$ for $\Delta = 4p + 3$ that we can express as $2\Delta - 2\lceil \frac{\Delta - 1}{4} \rceil$. That is slightly better than the value obtained in Theorem 7 more precisely one less when

$\Delta = 4p+2$ (resp. $4p+3$) where we get $6p+2$ (resp. $6p+4$) colors instead of $6p+3$ (resp. $6p+5$). In summary we get:

For maximum weight 2, we obtain an approximation with a multiplicative ratio of $\frac{3}{2}$ and an additive constant of $\frac{1}{2}$.

Claim 3. $n_i = \left\lceil \frac{\Delta+1-i-\sum_{l=1}^{i-1} n_l \Delta_i^l}{\Delta_i^i} \right\rceil$ *where* $\Delta_i^l = \max\{l, d_l + 1 - i\}$.

Proof. Suppose that a vertex v of weight i cannot be assigned to any block of $\mathcal{B}_l$ with $l \leq i$. This means that, for each block B of $\mathcal{B}_l$, vertex v and its neighbors in B form a clique of size $> d_l$ and so the weight of the neighbors of v in B is at least $d_l + 1 - i$. Furthermore, as all the vertices of weight $< l$ are assigned to blocks of $\mathcal{B}_j$ for $j < l$, v has necessarily one neighbor of weight at least l in any block of $\mathcal{B}_l$. Therefore, if we let $\Delta_i^l = \max\{l, d_l + 1 - i\}$, the clique induced by v and its neighbors in the RPEO has a weight at least $\sum_{l=1}^{i} n_l \Delta_i^l + i$ which exceeds Δ for $n_i = \left\lceil \frac{\Delta+1-i-\sum_{l=1}^{i-1} n_l \Delta_i^l}{\Delta_i^i} \right\rceil$. □

Maximum weight 3 Let $W = 3$. We choose some values for d_i and using the claims above, we obtain the following values of n_i:

- $d_1 = d_2 = d_3 = 3$. $n_1 = \lceil \frac{\Delta}{3} \rceil$ and $n_2 = \lceil \frac{\Delta-1-2n_1}{2} \rceil$ and $n_3 = \lceil \frac{\Delta-2-n_1-2n_2}{3} \rceil$.
- $d_1 = 5, d_2 = d_3 = 3$. $n_1 = \lceil \frac{\Delta}{5} \rceil$ and $n_2 = \lceil \frac{\Delta-1-4n_1}{2} \rceil$ and $n_3 = \lceil \frac{\Delta-2-3n_1-2n_2}{3} \rceil$.
- $d_1 = d_2 = 5, d_3 = 3$. $n_1 = \lceil \frac{\Delta}{5} \rceil$ and $n_2 = \lceil \frac{\Delta-1-4n_1}{4} \rceil$ and $n_3 = \lceil \frac{\Delta-2-3n_1-3n_2}{3} \rceil$.

To compare the values of the total number of colors we need to compute $c(3, S)$ for some basic sets S. We recall that $c(d, S)$ is the minimum number of colors which can be used in an interval coloring of any chordal graph with density d and weights in S.

- $\boldsymbol{c(3, \{1,2,3\}) = 4}$.

  We first prove that $c(3, \{1,2\}) \geq 4$. Let us consider the example presented in Figure 3 in which the density is 3 and the maximum weight is 2. The graph in the example consists of a clique of 3 vertices of weight one, such that each vertex of weight one is joined to a vertex of weight 2. This graph cannot be colored using only 3 colors. If we suppose that it can be colored with 3 colors $\{1,2,3\}$, then one of the vertices of weight one will have to be assigned color 2. For this vertex, the neighbor of weight 2 cannot be colored since the only available colors are 1 and 3 which are not contiguous.

  To prove that $c(3, \{1,2,3\}) \leq 4$, we use the First Fit algorithm in the RPEO which needs at most 4 colors .

- $\boldsymbol{c(3, \{2,3\}) = c(3, \{3\}) = 3}$.

  In fact in an instance of $\mathcal{C}(3, \{2,3\})$, all vertices are isolated and we can hence easily color them with at most 3 colors.

- $\boldsymbol{c(4, \{1,2,3\}) = 6}$.

  We first prove that $c(4, \{1,2,3\}) \geq 6$. Let us consider the example presented in Figure 4 which consists of a clique of four vertices of weight one. Each vertex of weight one is joined to a vertex of weight 3 and each pair of vertices of weight one is joined to a vertex of weight 2. Suppose that we only use 5 colors $\{1,2,3,4,5\}$ to color this graph. If one of the vertices of weight 1 uses color 3, then its neighbor which has weight 3 cannot be colored. Otherwise, if the vertices of weight 1 use colors $\{1,2,4,5\}$, then

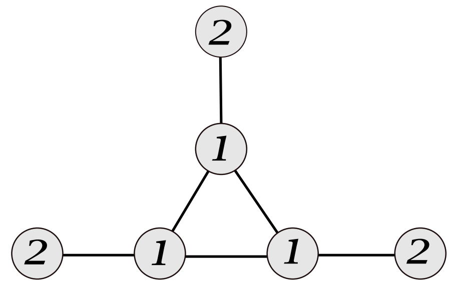


Figure 3: An example showing that $c(3,\{1,2\}) \neq 3$

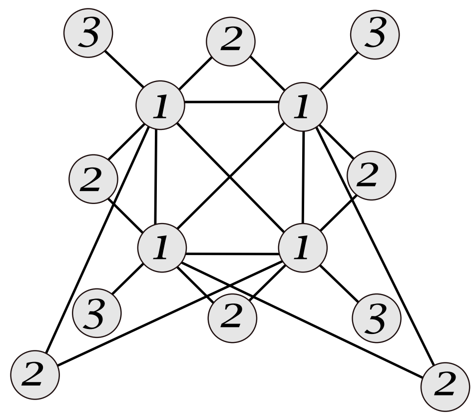


Figure 4: An example showing that $c(4,\{1,2,3\}) \neq 5$

the vertex of weight 2 which is adjacent to the vertices of weight 1 which have colors 2 and 4 cannot be colored.

To color any instance in $\mathcal{C}(4,\{1,2,3\})$ with at most 6 colors, we use the first fit algorithm in the RPEO.

- $\boldsymbol{c(4,\{2,3\})=4}$.

  The First Fit algorithm in the RPEO, colors any instance in $\mathcal{C}(4,\{2,3\})$ with at most 4 colors.

- $\boldsymbol{c(5,\{1,2,3\})=7}$.

  We first prove that $c(5,\{1,3\}) \geq 7$. Let us consider the graph G consisting of a clique of 5 vertices each of weight 1 and such that each pair of vertices of the clique is connected to a vertex of weight 3. The graph G is chordal with density 5 and weights in $\{1,3\}$. Let us suppose that we can color G with only six colors. There are either two vertices of weight one colored with colors 2 and 4 or two vertices of weight one colored with colors 3 and 5. In both cases the vertex of weight 3 adjacent to these two vertices cannot be colored.

  Now let us describe an algorithm that takes an instance of $\mathcal{C}(5,\{1,2,3\})$ and colors it with at most 7 colors. The algorithm is a First Fit algorithm in the RPEO of the vertices with the additional feature that colors 5 and 6 are forbidden for vertices of weight 1.

  - If a vertex v of weight 3 is considered, then if v has a neighbor of weight 2 colored with $\{\alpha,\alpha+1\}$, we color v with $\{1,2,3\}$ if $\alpha \geq 4$ or $\{5,6,7\}$ if $\alpha \leq 3$. If v has two neighbors of weight 1; if color 7 is not used we color v with $\{5,6,7\}$. If color 7 is used, but not color 4 we color v with $\{4,5,6\}$. If both colors 4 and 7 are used, we color v with $\{1,2,3\}$.

– If a vertex v of weight 2 is considered, then if v has 3 neighbors of weight 1, we color v with $\{5,6\}$. If it has one neighbor of weight 2 colored $\{\alpha, \alpha+1\}$ and one of weight 1 colored β, then we color v with $\{5,6\}$ if $\alpha \leq 3$; with $\{1,2\}$ if $\alpha \geq 4$ and $\beta \geq 3$; with color $\{6,7\}$ if $\alpha = 4$ and $\beta \leq 2$ or $\{3,4\}$ if $\alpha \geq 5$ and $\beta \leq 2$.

- $\boldsymbol{c(5, \{2,3\}) = 5}$.

  The First Fit algorithm in the RPEO in which we forbid color 3 to vertices of weight 2 uses at most 5 colors (note that a vertex of weight 3 cannot be colored with $\{2,3,4\}$).

Now, we can compute the number of colors for the 3 cases considered above.

- If we set $d_1 = d_2 = d_3 = 3$, the number of colors used is $n_1 c(3, \{1,2,3\}) + n_2 c(3, \{2,3\}) + n_3 c(3, \{3\}) = 4n_1 + 3n_2 + 3n_3$. As $n_3 \leq \frac{\Delta - n_1 - 2n_2}{3}$ the number of colors is at most $\Delta + 3n_1 + n_2$ and as $n_2 \leq \frac{\Delta - 2n_1}{2}$ it is at most $\frac{3\Delta}{2} + 2n_1$. Finally, as $n_1 \leq \frac{\Delta}{3} + \frac{2}{3}$, the number of colors used is at most $\frac{13}{6}\Delta + \frac{4}{3}$.
- If we set $d_1 = 5, d_2 = d_3 = 3$, the number of colors used is $n_1 c(5, \{1,2,3\}) + n_2 c(3, \{2,3\}) + n_3 c(3, \{3\}) = 7n_1 + 3n_2 + 3n_3$. As $n_3 \leq \frac{\Delta - 3n_1 - 2n_2}{3}$ the number of colors is at most $\Delta + 4n_1 + n_2$ and as $n_2 \leq \frac{\Delta - 4n_1}{2}$ it is at most $\frac{3\Delta}{2} + 2n_1$. Finally, as $n_1 \leq \frac{\Delta}{5} + \frac{4}{5}$, the number of colors used is at most $\frac{19}{10}\Delta + \frac{8}{5}$.
- If we set $d_1 = d_2 = 5$, and $d_3 = 3$, the number of colors used is $n_1 c(5, \{1,2,3\}) + n_2 c(5, \{2,3\}) + n_3 c(3, \{3\}) = 7n_1 + 5n_2 + 3n_3$. As $n_3 \leq \frac{\Delta - 3n_1 - 3n_2}{3}$ the number of colors is at most $\Delta + 4n_1 + 2n_2$ and as $n_2 \leq \frac{\Delta - 4n_1 + 2}{4}$ it is at most $\frac{3\Delta}{2} + 2n_1 + 1$. Finally, as $n_1 \leq \frac{\Delta}{5} + \frac{4}{5}$, the number of colors used is at most $\frac{19}{10}\Delta + \frac{13}{5}$.

We have tried other values of d_i and n_i but we obtained bigger numbers of colors. For example:

- If we set $d_1 = 4$, $d_2 = d_3 = 3$, using $c(4, \{1,2,3\}) = 6$, the number of colors is at most $\frac{17}{8}\Delta + \frac{15}{8}$.
- If we set $d_1 = d_2 = 4$, and $d_3 = 3$, using $c(4, \{2,3\}) = 4$ the number of colors is at most $\frac{13}{6}\Delta + \frac{13}{6}$.
- If we set $d_1 = 6$, and $d_2 = d_3 = 3$, then using $c(6, \{1,2,3\}) = 9$ (to be computed after), the number of colors is at most $\frac{23}{12}\Delta + \frac{25}{12}$.
- If we set $d_1 = 6$, $d_2 = 5$ and $d_3 = 3$, then using $c(5, \{2,3\}) = 5$, the number of colors is at most $\frac{23}{12}\Delta + \frac{37}{12}$.

For maximum weight 3, we obtain an approximation with a multiplicative ratio of $\frac{19}{10}$ and an additive constant of $\frac{8}{5}$.

Maximum weight 4 Let us first compute $c(4, S)$ for some basic sets S.

- $\boldsymbol{c(4, \{1,2,3,4\}) = 6}$.

  Since in an instance of $\mathcal{C}(4, \{1,2,3,4\})$, the vertices of weight 4 are isolated, we color them and then use the algorithm used to prove that $c(4, \{1,2,3\}) = 6$ to color the other vertices.

- $\boldsymbol{c(4, \{2,3,4\}) = 4}$.

  A First Fit algorithm in the RPEO uses at most 4 colors. In fact, in an instance of $\mathcal{C}(4, \{2,3,4\})$, vertices of weights 3 or 4 are isolated and it suffices to color vertices of weight 2 with $\{1,2\}$ or $\{3,4\}$.

- $\boldsymbol{c(4, \{3,4\}) = c(4, \{4\}) = 4}$.

  In an instance of $\mathcal{C}(4, \{3,4\})$, all vertices are isolated and can be colored independently.

- $\boldsymbol{c(5, \{1,2,3,4\}) = 8}$.

  We first prove that $c(5, \{1, 2, 3, 4\}) \geq 8$. Let us consider the graph G which consists of a clique of 5 vertices of weight 1 each and such that each vertex of weight one is connected to a new vertex of weight 4 and each pair of vertices of weight 1 is connected to a new vertex of weight 3. The graph G is chordal and has density 5 and maximum weight 4. Suppose that only 7 colors can be used to color G. Color 4 cannot be used for any vertex of weight 1, otherwise its neighbor of weight 4 cannot be colored. Furthermore we can use for vertices of weight 1 at most one of the pair of colors $\{2, 5\}$ and $\{3, 6\}$, otherwise the neighbor of weight 3 connected to this pair cannot be colored. So we have altogether 3 colors forbidden for vertices of weight 1 and so only 4 available colors which is impossible.

  Now let us describe an algorithm that takes an instance of $\mathcal{C}(5, \{1, 2, 3, 4\})$ and colors it with at most 8 colors. The algorithm uses first fit in the RPEO with the additional feature that colors 3 and 6 are forbidden to vertices of weight 1. Let us check that this algorithm uses indeed at most 8 colors.

  - If a vertex v of weight 4 is considered, then if v has a neighbor of weight 1 which has been already colored α, we color v with $\{1, 2, 3, 4\}$ if $\alpha \geq 5$ or $\{5, 6, 7, 8\}$ if $\alpha \leq 4$.
  - If a vertex v of weight 3 is considered, then if v has a neighbor of weight 2 colored with $\{\alpha, \alpha+1\}$, we color v with $\{1, 2, 3\}$ if $\alpha \geq 4$ or with $\{6, 7, 8\}$ if $\alpha \leq 3$. If v has two neighbors of weight 1 colored with $\alpha < \beta$, we color v with $\{1, 2, 3\}$ if $\alpha \geq 4$ or $\{3, 4, 5\}$ if $\alpha \leq 2$ and $\beta \geq 7$ or $\{6, 7, 8\}$ if $\alpha \leq 2$ and $\beta \leq 5$ (recall that $\alpha \neq 3$ and $\beta \neq 6$).
  - If a vertex v of weight 2 is considered, then if v has 3 neighbors of weight 1 colored with $\alpha < \beta < \gamma$, we color v with $\{1, 2\}$ if $\alpha \geq 4$, or $\{3, 4\}$ if $\alpha \leq 2$ and $\beta \geq 5$, or $\{5, 6\}$ if $\beta \leq 4$ and $\gamma \geq 7$, or $\{7, 8\}$ if $\gamma \leq 5$. If v has one neighbor of weight 2 colored with $\{\alpha, \alpha + 1\}$ and one of weight 1 colored β, we color v with $\{1, 2\}$ if $\alpha \geq 3$ and $\beta \geq 4$, or $\{4, 5\}$ if $\alpha \leq 2$ and $\beta \geq 7$, or $\{7, 8\}$ if $\alpha \leq 2$ and $\beta \leq 5$, or $\{3, 4\}$ if $\alpha \geq 5$ and $\beta \leq 2$, or $\{7, 8\}$ if $\alpha \leq 4$ and $\beta \leq 2$.

- $\boldsymbol{c(5, \{2,3,4\}) = 5}$.

  In an instance of $\mathcal{C}(5, \{2, 3, 4\})$, vertices of weight 4 are isolated. We can color them then all with colors $\{1, 2, 3, 4\}$. For the vertices of weights 2 and 3, we use the algorithm which achieves $c(5, \{2, 3\}) = 5$.

- $\boldsymbol{c(6, \{1,2,3,4\}) = c(6, \{1,2,3\}) = 9}$.

  We first prove that $c(6, \{1, 3\}) \geq 9$. Let us consider the graph G consisting of a clique of 6 vertices of weight 1 each and such that the vertices of each triple of the clique are connected to a vertex of weight 3. The graph G is chordal with density 6 and weights in $\{1, 3\}$. Let us suppose that we can color G with only 8 colors. There are three vertices of weight 1 using colors $\{1, 4, 7\}$ or three vertices of weight 1 using colors $\{2, 5, 8\}$ or two vertices of weight 1 using colors $\{3, 6\}$. In any of these three cases, a vertex of weight 3 cannot be colored.

  Now let us describe an algorithm that takes an instance of $\mathcal{C}(6, \{1, 2, 3, 4\})$ and colors it with at most 9 colors. The algorithm is a First Fit algorithm in the RPEO of the vertices with two additional features: colors 6,7 and 8 are forbidden to vertices of weight 1, and each vertex of weight 2 is assigned colors $\{1, 2\}$, $\{3, 4\}$, $\{5, 6\}$, or $\{7, 8\}$ and not any other contiguous combination of two colors. This algorithm uses at most 9 colors.

  - If a vertex v of weight 4 is considered, then if v has a neighbor of weight 2 which has been already colored, the possible sets of color used by this neighbor are $\{1, 2\}$, $\{3, 4\}$, $\{5, 6\}$, or $\{7, 8\}$. In any case, v can be colored with 4 contiguous colors. If v has two neighbors of weight 1 each that have been already colored, then either one of the colors 9 or 5 is not used by this neighbor, and in this case v can use it along with the colors $\{6, 7, 8\}$ (which are forbidden for vertices of weight 1), or both colors 9 and 5 are used and v can use colors $\{1, 2, 3, 4\}$.
  - If a vertex v of weight 3 is considered, then if v has a neighbor of weight 3 colored with $\{\alpha, \alpha + 1, \alpha + 2\}$, we color v with $\{1, 2, 3\}$ if $\alpha \geq 4$ or with $\{7, 8, 9\}$ if $\alpha \leq 3$. If v has two neighbors one of

weight 2 colored with $\{\alpha, \alpha+1\}$ and one of weight 1 colored with β, then we color v with $\{6,7,8\}$ if $\alpha = 1$ or 3; $\{1,2,3\}$ if $\alpha = 5$ or 7 and $\beta \geq 4$; $\{7,8,9\}$ if $\alpha = 5$ and $\beta \leq 3$; $\{4,5,6\}$ if $\alpha = 7$ and $\beta \leq 3$.

  – If a vertex v of weight 2 is considered. If all its neighbors that have been already colored are of weight 1, then v can be assigned colors $\{7,8\}$. If v has two colored neighbors of weight 2 each, or one colored neighbor of weight 2 and two other colored neighbors with weight 1, or one colored neighbor of weight 3 and another of weight 1, or one vertex of weight 4, then one of the channels $\{1,2\}$, $\{3,4\}$, $\{5,6\}$, or $\{7,8\}$ is necessarily free to be used.

- $\boldsymbol{c(6, \{2,3,4\}) = 8.}$

  We first prove that $c(6, \{2,4\}) \geq 8$. Let us consider the graph G which consists of a clique of 3 vertices of weight 2 each, and such that each vertex of weight 2 is connected to a vertex of weight 4. The graph G is chordal and has density 6 and weights in $\{2,4\}$. Let us suppose that we can color G with only 7 colors. In any possible coloring of the vertices of weight 2, a vertex v of weight 2 has to use either colors $\{3,4\}$ or $\{4,5\}$. In both cases, the neighbor of v which has weight 4 cannot be colored.

  Now let us describe the algorithm that colors an instance of $\mathcal{C}(6, \{2,3,4\})$. The algorithm uses First Fit in the RPEO with the additional feature that the possible combinations of colors for vertices of weight 2 are the following: $\{1,2\}$, $\{3,4\}$, $\{5,6\}$, and $\{7,8\}$.

- $\boldsymbol{c(6, \{3,4\}) = 6.}$

  Vertices of weight 4 are isolated and can be all assigned colors $\{1,2,3,4\}$ and other vertices can be colored using first fit in the RPEO with at most 6 colors.

Now, we can compute the number of colors for $d_1 = 6$ and $d_2 = d_3 = d_4 = 4$.

The number of colors used is $n_1 c(6, \{1,2,3,4\}) + n_2 c(4, \{2,3,4\}) + n_3 c(4, \{3,4\}) + n_4 c(4, \{4\}) = 9n_1 + 4n_2 + 4n_3 + 4n_4$.

We have $n_1 = \lceil \frac{\Delta}{6} \rceil$; $n_2 = \lceil \frac{\Delta - 1 - 5n_1}{3} \rceil$; $n_3 = \lceil \frac{\Delta - 2 - 4n_1 - 2n_2}{3} \rceil$, and $n_4 = \lceil \frac{\Delta - 3 - 3n_1 - 2n_2 - 3n_3}{4} \rceil$.

As $n_4 \leq \frac{\Delta - 3n_1 - 2n_2 - 3n_3}{4}$ the number of colors is at most $\Delta + 6n_1 + 2n_2 + n_3$. As $n_3 \leq \frac{\Delta - 4n_1 - 2n_2}{3}$ the number of colors is at most $\frac{4\Delta}{3} + \frac{14}{3}n_1 + \frac{4}{3}n_2$, and as $n_2 \leq \frac{\Delta - 5n_1 + 1}{3}$ it is at most $\frac{16\Delta}{9} + \frac{22}{9}n_1 + \frac{4}{9}$. Finally, as $n_1 \leq \frac{\Delta+5}{6}$, the number of colors is at most $\frac{59}{27}\Delta + \frac{67}{27}$.

We have computed the number of colors for other choices of the d_i but the values are bigger.

- For $d_1 = d_2 = d_3 = d_4 = 4$, then the number of colors used is $6n_1 + 4n_2 + 4n_3 + 4n_4$ which is at most $\frac{91}{36}\Delta + \frac{97}{36}$ colors.
- For $d_1 = 5$ and $d_2 = d_3 = d_4 = 4$, then the number of colors used is $8n_1 + 4n_2 + 4n_3 + 4n_4$ which is at most $\frac{109}{45}\Delta + \frac{176}{45}$ colors.
- For $d_1 = d_2 = 5$ and $d_3 = d_4 = 4$, then the number of colors used is $8n_1 + 5n_2 + 4n_3 + 4n_4$ which is at most $\frac{73}{30}\Delta + \frac{17}{5}$ colors.
- For $d_1 = d_2 = d_3 = 6$, $d_4 = 4$, then the number of colors used is $9n_1 + 8n_2 + 6n_3 + 4n_4$ which is at most $\frac{139}{60}\Delta + \frac{167}{60}$ colors.
- For $d_1 = d_2 = 6$ and $d_3 = d_4 = 4$, then the number of colors used is $9n_1 + 8n_2 + 4n_3 + 4n_4$ which is at most $\frac{67}{30}\Delta + \frac{91}{30}$ colors.

For maximum weight 4, we obtain an approximation with a multiplicative ratio of $\frac{59}{27}$ and an additive constant of $\frac{67}{27}$.

Maximum weight 5 Let us first compute $c(5, S)$ for some basic sets S.

- $\boldsymbol{c(5, \{1, 2, 3, 4, 5\}) = 8.}$

  In fact, we know that $c(5, \{1, \ldots, 4\}) = 8$ and for any chordal graph with density 5 and maximum weight 5, vertices of weight 5 are isolated and can be colored independently from the others.

- $\boldsymbol{c(5, \{2, 3, 4, 5\}) = 5.}$

  In a chordal graph of density 5 and weights in $\{2, \ldots, 5\}$, vertices of weight 4 or 5 are isolated and can be easily colored. For other vertices, we know that $c(5, \{2, 3\}) = 5$.

- $\boldsymbol{c(5, \{3, 4, 5\}) = c(5, \{4, 5\}) = 5.}$

  All vertices are isolated and can be easily colored.

- $\boldsymbol{c(6, \{1, 2, 3, 4, 5\}) = 10.}$

  We first prove that $c(6, \{1, \ldots, 5\}) = 10$. Let us consider the graph G which consists of a clique C of 6 vertices of weight 1 such that each of the vertices of C is connected to a vertex of weight 5, and each pair of vertices of C is connected to a vertex of weight 4. Let us suppose that we can color G with 9 colors. Color 5 cannot be used for any vertex of weight 1, otherwise its neighbor of weight 5 cannot be colored. Furthermore we can use for vertices of weight 1 at most one of the pair of colors $\{2, 6\}$, $\{3, 7\}$, $\{4, 8\}$, otherwise the neighbor of weight 4 connected to this pair cannot be colored. So we have altogether 4 colors forbidden for vertices of weight 1 and so only 5 available colors which is impossible.

  The First Fit in the RPEO with the additional feature of forbidding colors $\{7, 8, 9, 10\}$ to vertices of weight 1 colors any instance of $\mathcal{C}(6, \{1, \ldots, 5\})$ with at most 10 colors.

- $\boldsymbol{c(6, \{2, 3, 4, 5\}) = 8.}$

  Vertices of weight 5 are isolated and can be easily colored. As for other vertices we have already proved that $c(6, \{2, 3, 4\}) = 8$.

- $\boldsymbol{c(6, \{3, 4, 5\}) = 6.}$

  Vertices of weight 4 and 5 are isolated and vertices of weight 3 can be colored using First Fit in the RPEO with either the colors $\{1, 2, 3\}$ or $\{4, 5, 6\}$.

- $\boldsymbol{c(7, \{1, 2, 3, 4, 5\}) \leq 12.}$

  To obtain a coloring with 12 colors, we use the First Fit algorithm in the RPEO with the additional feature of forbidding colors $\{8, 9, 10, 11, 12\}$ to vertices of weight 1 and colors $\{8, 9\}$ and $\{9, 10\}$ for vertices of weight 2. The proof that the algorithm works is done by considering the various possibilities when a new vertex is added.

  - If a vertex v of weight 5 is considered, then if v has:
    - 2 neighbors of weight 1, we color v with $\{8, 9, 10, 11, 12\}$.
    - 1 neighbor of weight 2 colored with $\{\alpha, \alpha + 1\}$, we color v with $\{1, 2, 3, 4, 5\}$ if $\alpha \geq 6$ or with $\{8, 9, 10, 11, 12\}$ if $\alpha \leq 5$.
  - If a vertex v of weight 4 is considered, then if v has:
    - 3 neighbors of weight 1, we color v with $\{9, 10, 11, 12\}$.
    - 1 neighbor of weight 2 colored with $\{\alpha, \alpha + 1\}$ and 1 neighbor of weight 1 colored β, we color v with $\{9, 10, 11, 12\}$ if $\alpha \leq 7$; otherwise $\alpha \geq 10$ and we color v with $\{1, 2, 3, 4\}$ if $\beta \geq 5$ or with $\{5, 6, 7, 8\}$ if $\beta \leq 4$.(Note that we use the fact that $\alpha \neq 8$; otherwise with $\alpha = 8$ and $\beta = 4$ we could not have colored v).
    - 1 neighbor of weight 3 colored with $\{\alpha, \alpha + 1, \alpha + 2\}$, we color v with $\{1, 2, 3, 4\}$ if $\alpha \geq 5$ or $\{9, 10, 11, 12\}$ if $\alpha \leq 4$.

– If a vertex v of weight 3 is considered, then if v has:
  * 4 neighbors of weight 1, we color v with $\{10, 11, 12\}$.
  * 1 neighbor of weight 2 colored with $\{\alpha, \alpha+1\}$ and 2 neighbors of weight 1 colored $\beta < \gamma$, we color v with $\{10, 11, 12\}$ if $\alpha \leq 7$; otherwise $\alpha \geq 10$, and we color v with $\{1, 2, 3\}$ if $\beta \geq 4$ or with $\{4, 5, 6\}$ if $\beta \leq 3$ and $\gamma \geq 7$ or with $\{7, 8, 9\}$ if $\beta \leq 3$ and $\gamma \leq 6$. (Note that we use the fact that $\alpha \neq 9$; otherwise with $\alpha = 9$, $\beta = 3$ and $\gamma = 6$ we could not have colored v).
  * 2 neighbors of weight 2 colored with $\{\alpha, \alpha+1\}$ $\{\beta, \beta+1\}$ with $\alpha < \beta$, we color v with $\{1, 2, 3\}$ if $\alpha \geq 4$, or with $\{5, 6, 7\}$ if $\alpha \leq 3$ and $\beta \geq 10$, or with $\{10, 11, 12\}$ if $\alpha \leq 3$ and $\beta \leq 7$.
  * 1 neighbor of weight 3 colored with $\{\alpha, \alpha+1, \alpha+2\}$ and 1 neighbor of weight 1 colored β, we color v with $\{10, 11, 12\}$ if $\alpha \leq 7$ or $\{1, 2, 3\}$ if $\alpha \geq 8$ and $\beta \geq 4$ or with $\{4, 5, 6, \}$ if $\alpha \geq 8$ and $\beta \leq 3$.
  * 1 neighbor of weight 4 colored with $\{\alpha, \alpha+1, \alpha+2, \alpha+3\}$, we color v with $\{1, 2, 3\}$ if $\alpha \geq 4$ or $\{10, 11, 12\}$ if $\alpha \leq 3$.

– If a vertex v of weight 2 is considered, then if v has:
  * 5 neighbors of weight 1, we color v with $\{11, 12\}$.
  * 1 neighbor of weight 2 colored with $\{\alpha, \alpha+1\}$ and 3 neighbors of weight 1 colored $\beta < \gamma < \delta$, we color v with $\{11, 12\}$ if $\alpha \leq 7$; otherwise if $\alpha \geq 10$ we color v with $\{1, 2\}$ if $\beta \geq 3$, or with $\{3, 4\}$ if $\beta \leq 2$ and $\gamma \geq 5$, or with $\{5, 6\}$ if $\beta \leq 2$, $\gamma \leq 4$ and $\delta \geq 7$, or with $\{7, 8\}$ if $\beta \leq 2$, $\gamma \leq 4$ and $\delta \leq 6$.
  * 2 neighbors of weight 2 colored with $\{\alpha, \alpha+1\}$ and $\{\beta, \beta+1\}$ with $\alpha < \beta$, and a neighbor of weight 1 colored γ, we color v with $\{11, 12\}$ if $\beta \leq 7$; otherwise if $\beta \geq 10$ we color v with $\{1, 2\}$ if $\alpha \geq 3$ and $\gamma \geq 3$, or with $\{4, 5\}$ if $\alpha \leq 2$ and $\gamma \geq 6$, or with $\{6, 7\}$ if $\alpha \leq 2$ and $\gamma \leq 5$, or with $\{3, 4\}$ if $\alpha \geq 5$ and $\gamma \leq 2$, or with $\{6, 7\}$ if $\alpha \leq 4$ and $\gamma \leq 2$.
  * 1 neighbor of weight 3 colored with $\{\alpha, \alpha+1, \alpha+2\}$ and 2 neighbors of weight 1 colored $\beta < \gamma$, we color v with $\{11, 12\}$ if $\alpha \leq 8$; otherwise if $\alpha \geq 9$ we color v with $\{1, 2\}$ if $\beta \geq 3$ or with $\{4, 5\}$ if $\beta \leq 2$ and $\gamma \geq 6$ or with $\{6, 7\}$ if $\beta \leq 2$ and $\gamma \leq 5$.
  * 1 neighbor of weight 3 colored with $\{\alpha, \alpha+1, \alpha+2\}$ and 1 neighbor of weight 2 colored $\{\beta, \beta+1\}$, we color v with $\{1, 2\}$ if $\alpha \geq 3$ and $\beta \geq 3$, or with $\{11, 12\}$ if $\alpha \leq 2$ and $\beta \leq 7$, or with $\{5, 6\}$ if $\alpha \leq 2$ and $\beta \geq 10$, or with $\{4, 5\}$ if $\alpha \geq 6$ and $\beta \leq 2$, or with $\{11, 12\}$ if $\alpha \leq 5$ and $\beta \leq 2$.
  * 1 neighbor of weight 4 colored with $\{\alpha, \alpha+1, \alpha+2, \alpha+3\}$ and 1 neighbor of weight 1 colored β, we color v with $\{11, 12\}$ if $\alpha \leq 7$ or $\{1, 2\}$ if $\alpha \geq 8$ and $\beta \geq 3$ or with $\{3, 4\}$ if $\alpha \geq 8$ and $\beta \leq 2$.
  * 1 neighbor of weight 5 colored with $\{\alpha, \alpha+1, \alpha+2, \alpha+3, \alpha+4\}$, we color v with $\{1, 2\}$ if $\alpha \geq 3$ or $\{11, 12\}$ if $\alpha \leq 2$.

Now we can compute the number of colors for $d_1 = 7$ and $d_2 = d_3 = d_4 = d_5 = 5$.

The number of colors used is $n_1 c(7, \{1, 2, 3, 4, 5\}) + n_2 c(5, \{2, 3, 4, 5\}) + n_3 c(5, \{3, 4, 5\}) + n_4 c(5, \{4, 5\}) + n_5 c(5, \{5\}) = 12n_1 + 5n_2 + 5n_3 + 5n_4 + 5n_5$.

We have $n_1 = \lceil \frac{\Delta}{7} \rceil$, $n_2 = \lceil \frac{\Delta-1-6n_1}{4} \rceil$, $n_3 = \lceil \frac{\Delta-2-5n_1-3n_2}{3} \rceil$, $n_4 = \lceil \frac{\Delta-3-4n_1-2n_2-3n_3}{4} \rceil$, and $n_5 = \lceil \frac{\Delta-4-3n_1-2n_2-3n_3-4n_4}{5} \rceil$.

As in the preceding cases, we, successively, use upper bounds for the n_i. The number of colors is at most $\Delta + 9n_1 + 3n_2 + 2n_3 + n_4$, then $\frac{5\Delta}{4} + 8n_1 + \frac{5}{2}n_2 + \frac{5}{4}n_3$, then $\frac{5\Delta}{3} + \frac{71}{12}n_1 + \frac{5}{4}n_2$, then $\frac{95\Delta}{48} + \frac{97}{24}n_1 + \frac{5}{8} \leq \frac{859}{336}\Delta + \frac{229}{56}$.

We have computed the number of colors for other choices of the d_i but we obtained bigger values as we present in what follows.

- If we set $d_1 = d_2 = d_3 = d_4 = d_5 = 5$, then the number of colors used is $8n_1 + 5n_2 + 5n_3 + 5n_4 + 5n_5$ which is at most $\frac{679}{240}\Delta + \frac{161}{40}$.

- If we set $d_1 = d_2 = d_3 = 6, d_4 = d_5 = 5$, then the number of colors used is $10n_1 + 8n_2 + 6n_3 + 5n_4 + 5n_5$ which is at most $\frac{659}{240}\Delta + \frac{343}{60}$.
- If we set $d_1 = 6, d_2 = d_3 = d_4 = d_5 = 5$, then the number of colors used is $10n_1 + 5n_2 + 5n_3 + 5n_4 + 5n_5$ which is at most $\frac{763}{288}\Delta(G, w) + \frac{1275}{288}$.

For maximum weight 5, we obtain an approximation with a multiplicative ratio of $\frac{859}{336}$ and an additive constant of $\frac{229}{56}$.

Maximum weight 6 Let us first compute $c(6, S)$ for some basic sets S. Note that with a density at most 6, any vertices of weight 6 are isolated. We can then deduce the following from what we have computed for a maximum weight of 5.

- $\boldsymbol{c(6, \{1, 2, 3, 4, 5, 6\}) = 10.}$
- $\boldsymbol{c(6, \{2, 3, 4, 5, 6\}) = 8.}$
- $\boldsymbol{c(6, \{3, 4, 5, 6\}) = c(6, \{4, 5, 6\}) = c(6, \{5, 6\}) = c(6, \{6\}) = 6.}$
- $\boldsymbol{c(7, \{1, 2, 3, 4, 5, 6\}) = 12.}$

  We use the algorithm which gives $c(7, \{1, 2, 3, 4, 5, \}) \leq 12$. If a vertex v of weight 6 is added it is joined to at most one vertex of weight 1 of color β. If $\beta \leq 6$, we color v with colors $\{7, 8, 9, 10, 11, 12\})$ and if $\beta = 7$ with colors $\{1, 2, 3, 4, 5, 6\})$ and so $c(7, \{1, 2, 3, 4, 5, 6\}) \leq 12$.

  To show that $c(7, \{1, 2, 3, 4, 5, 6\}) \geq 12$, we consider the chordal graph consisting of a clique of 7 vertices of weight one such that each vertex of this clique is joined to a vertex of weight 6. Furthermore we join each pair of vertices of weight 1 to a vertex of weight 5. Suppose that we can color the graph with 11 colors. Color 6 cannot be used for any vertex of weight 1, otherwise its neighbor of weight 6 cannot be colored. Furthermore we can use for vertices of weight 1 at most one color of each of the following pairs of colors $\{2, 7\}$, $\{3, 8\}$, $\{4, 9\}$, $\{5, 10\}$, otherwise the neighbor of weight 5 connected to this pair cannot be colored. So we have altogether 5 colors forbidden for vertices of weight 1 and so only 6 available colors which is impossible.

Now we can compute the number of colors for $d_1 = 7$ and $d_2 = d_3 = d_4 = d_5 = d_6 = 6$.

The number of colors used is $n_1 c(7, \{1, 2, 3, 4, 5, 6\}) + n_2 c(6, \{2, 3, 4, 5, 6\}) + n_3 c(6, \{3, 4, 5, 6\}) + n_4 c(6, \{4, 5, 6\}) + n_5 c(6, \{5, 6\}) + n_6 c(6, \{6\}) = 12n_1 + 8n_2 + 6n_3 + 6n_4 + 6n_5 + 6n_6$.

We have $n_1 = \lceil \frac{\Delta}{7} \rceil$, $n_2 = \lceil \frac{\Delta - 1 - 6n_1}{5} \rceil$, $n_3 = \lceil \frac{\Delta - 2 - 5n_1 - 4n_2}{4} \rceil$, $n_4 = \lceil \frac{\Delta - 3 - 4n_1 - 3n_2 - 3n_3}{4} \rceil$, $n_5 = \lceil \frac{\Delta - 4 - 3n_1 - 2n_2 - 3n_3 - 4n_4}{5} \rceil$, and $n_6 = \lceil \frac{\Delta - 5 - 2n_1 - 2n_2 - 3n_3 - 4n_4 - 5n_5}{6} \rceil$.

Like in the preceding cases, we obtain upperbounds on n_i. The number of colors is at most $\Delta + 10n_1 + 6n_2 + 3n_3 + 2n_4 + n_5$, then $\frac{6\Delta}{5} + \frac{47}{5}n_1 + \frac{28}{5}n_2 + \frac{12}{5}n_3 + \frac{6}{5}n_4$, then $\frac{3\Delta}{2} + \frac{41}{5}n_1 + \frac{47}{10}n_2 + \frac{3}{2}n_3$, then $\frac{15\Delta}{8} + \frac{253}{40}n_1 + \frac{16}{5}n_2 + \frac{3}{8}$, then $\frac{503\Delta}{200} + \frac{497}{200}n_1 + \frac{459}{200} \leq \frac{287}{100}\Delta + \frac{885}{200}$.

If we set $d_1 = d_2 = d_3 = 6 = d_4 = d_5 = d_6 = 6$, the number of colors used is $10n_1 + 8n_2 + 6n_3 + 6n_4 + 6n_5 + 6n_6$ which is at most $\frac{603}{200}\Delta(G, w) + O(1)$.

We could improve the value if we could prove that $c(8, \{1, 2, 3, 4, 5, 6\}) \leq 14$ but that seems not possible. We can only prove $c(8, \{1, 2, 3, 4, 5, 6\}) \leq 15$ which gives a bigger number of colors.

For maximum weight 6, we obtain an approximation with a multiplicative ratio of $\frac{287}{100}$ and an additive constant of $\frac{885}{200}$. □

Theorem 11 implies the following corollary.

Corollary 5. *Let $\mathcal{I}$ be an instance of SA in a binary tree. Let $\mathcal{OPT}$ be the span of $\mathcal{I}$. There are polynomial-time algorithms which find a spectrum assignment for $\mathcal{I}$ with a span less than $\frac{3}{2}\mathcal{OPT} + \frac{1}{2}$, $\frac{19}{10}\mathcal{OPT} + \frac{8}{5}$, $\frac{59}{27}\mathcal{OPT} + \frac{67}{27}$, $\frac{859}{336}\mathcal{OPT} + \frac{229}{56}$ and $\frac{287}{100}\mathcal{OPT} + \frac{885}{200}$ when the maximum request demand is bounded by 2,3,4,5 and 6, respectively.*

5 Conclusion

We have studied in this report the problem of Spectrum Assignment (SA) in tree networks. We have proved that SA is NP-complete in undirected stars with 3 links and directed stars with 4 links. We have also shown that there is a 4-approximation algorithm to solve the problem in general stars. Afterwards, we have focused on SA in binary trees with special demand profiles and we have designed constant approximation algorithms for several cases. As future work, we would like to find approximation algorithms for interval coloring in chordal graphs in general and to SA in binary trees in particular. Towards this objective, we believe the following directions might be useful.

- It would be interesting to try to use the clique graph of the chordal graph [7] to find an acyclic orientation where the number of maximal cliques to which a path belongs is bounded. In fact, finding a k-approximation for interval coloring is equivalent to finding an acyclic orientation in which the longest directed path has vertices in at most k maximal cliques [12]. This approach has been used to find a 2-approximation for interval coloring in claw-free chordal graphs [6].
- It would be also helpful to try to use ideas from the approximation algorithms used for interval coloring in interval graphs. These algorithms were developed for the problem of Dynamic Storage Allocation (DSA) as we mentioned in Section 2.2.2 and they use mainly three techniques.
  - 2-coloring (2-allocation) [9]: in this technique, which yields a 3-approximation for Interval Coloring (IC) in interval graphs, first, a 2-coloring is found where 2 adjacent vertices but not three might use the same color. This 2-coloring is transformed afterwards to a normal coloring. Is it possible to find a 2-coloring for chordal graphs in polynomial time?
  - Boxing vertices [3]: in this technique, which yields a $2+\epsilon$ approximation for IC in interval graphs, vertices are modeled as rectangles (the dimensions of a rectangle corresponding to a vertex v are the weight of v and the interval corresponding to v in the interval representation of the graph). These rectangles are cleverly boxed or gathered in larger rectangles. Afterwards an exact algorithm is used to color these large rectangles. Is it possible to adapt such technique to chordal graphs and find a clever way to box the vertices?
  - Buddy-decreasing-size algorithm [5]: in this algorithm, which yields a 6-approximation for IC in interval graphs, vertices are colored in the decreasing order of their weights. Some of the challenges in this direction is that using it as it is for chordal graphs cannot give better than a $log(n)$-approximation; there is a tight example in [21]. In the tight example however all the vertices have the same weight which means that there is an exponential number of possible orders. Is there a clever order (something similar to lexicographic order?) which can give a better approximation ratio?

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 *Proc. 18th Symp. on Discrete Algorithms,ACM-SIAM* (2007), p. 1056–1065.

[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 (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 (1992), 1171–1182.

[5] Chrobak, M., and Ślusarek, 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 (2002), 73–90.

[7] Galinier, P., Habib, M., and Paul, C. Chordal graphs and their clique graphs. In *Graph-Theoretic Concepts in Computer Science*, M. Nagl, Ed., vol. 1017 of *Lecture Notes in Computer Science*. Springer, Heidelberg, 1995, pp. 358–371.

[8] 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 Heidelberg, 1996, pp. 52–61.

[9] 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.

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

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

[12] 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.

[13] Golumbic, M. C., and Jamison, R. E. The edge intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B 38*, 1 (1985), 8 – 22.

[14] 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.

[15] Hoogeveen, J., van de Velde, S., and Veltman, B. Complexity of scheduling multiprocessor tasks with prespecified processor allocations. *Discrete Applied Mathematics 55*, 3 (1994), 259 – 272.

[16] 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 Heidelberg, 2000, pp. 60–71.

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

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

[19] 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 Heidelberg, 2004, pp. 339–348.

[20] 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.

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

[22] Shalom, M. On the interval chromatic number of proper interval graphs. *Discrete Mathematics 338*, 11 (2015), 1907 – 1916.

[23] 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.

[24] 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 (2014), 754–763.

[25] 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.

[26] Wang, R., and Mukherjee, B. Spectrum management in heterogeneous bandwidth optical networks. *Optical Switching and Networking 11, Part A* (2014), 83 – 91.