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. 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 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 Assignment 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 [23], 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 [22], 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.

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 in trees and design a constant-factor approximation algorithm or prove that such algorithm does not exist. In our study, we also 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 in Section 2. In Section **??**, we survey the complexity results in paths. We present our results on stars and binary trees in Sections 4 and 5, respectively.

2 Problem statement

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 in N and a spectrum demand $d_r \in \mathbb{N}$ (number of slots), a spectrum assignment of $(N, \mathcal{R})$ is a mapping f from $\mathcal{R}$ to $\mathbb{N}^*$ such that for every two requests $r, r' \in \mathcal{R}$, if the two paths p_r and $p_{r'}$ share an edge, (i.e., $p_r \cap p_{r'} \neq \emptyset$) 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, denoted by ℓ_e, is the sum of the demands of the requests using e and the ***load of an instance***, denoted by ℓ, is the maximum load over all its edges. Two requests are **conflicting** if their paths share an edge.

The ***greedy algorithm*** for SA is an algorithm which assigns spectrum to requests 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 $[g(r_i), g(r_i) + d_i - 1] \cap [g(r_j), g(r_j) + d_j - 1] = \emptyset$ for each r_j in $\{r_1, \ldots, r_{i-1}\}$ such that $p_{r_j} \cap p_{r_i} \neq \emptyset$.

3 Spectrum assignments in paths: short survey

The problem of Spectrum Assignment in paths has been recently studied in [23] and [22]. In [23], 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 a scheduling problem are used to find approximation for SA in path networks with few links. In [22], 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.

The problem of spectrum allocation in path networks is equivalent to the problem of Dynamic Storage Allocation.

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?

If we consider the time interval as a path network and each of the items to be stored as requests we can see the equivalence between the SA in paths and DSA. The problem of DSA has been extensively studied. We give in what follows some of the results that have been established for DSA and which, by equivalence, apply to SA in paths. In Sections 4 and 5, we will use some of the techniques used DSA not for SA in paths but for SA in stars and binary trees.

NP-completeness 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].

Approximation Many algorithms have been designed to approximate the problem of Dynamic Storage Allocation. The first algorithms are based on the greedy algorithm also called First Fit (FF) 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 [17] and the ratio of 6 in [18].

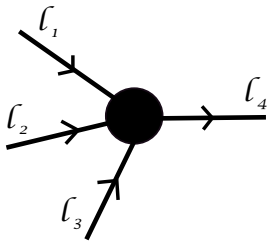
Gregov has adopted another approach not using FF, yielding an approximation of 5 and 3 sequentially in [8] and [9] . In his approach, Gregov defines a 2-allocation where two items but not three are allowed to overlap. This 2-allocation is achieved by allocating to every item i (in the decreasing order of the sizes after rounding them to powers of 2) the biggest stable position, which is a position a such that all the memory slots below a have been used by items conflicting with i (i.e. items whose time intervals intersect with the time interval of i). 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 (the dimensions of a rectangle corresponding to an item i are the duration of i and the size of i), in larger rectangles so that the total wasted space in the rectangles remains small and then use an exact algorithm to assign memory to the large rectangles.

Paper [19] 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 conflicting groups 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 [20], it is proved that for instances with sizes of 1 and X, ordered by starting times, First Fit has a guarantee of $2 - \frac{1}{X}$.

4 Spectrum assignment in stars: hardness and approximability results

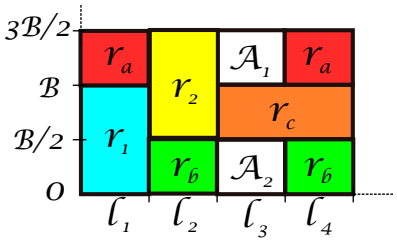
A star is a tree-network with at most one node of degree at least 2. The problem of wavelength assignment (WA) is NP-complete in undirected stars but polynomial in directed stars [1]. The polynomiality of WA in directed stars was useful because optical networks are symmetrically directed and because it helped in the



(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

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.

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

Proof. It is shown in [23] that the SA problem is strongly 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.

2-PARTITION Given a set A of k integers $a_1, a_2, \ldots, a_k$ such that $B = \sum_{j=1}^{k} a_j$. Can A be partitioned into two disjoint sets A_1 and A_2 such that $\sum_{a_j \in A_1} a_j = \sum_{a_j \in A_2}$?

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 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 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 all 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 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.
- 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 (both directed and undirected).*

Proof. We prove that the Buddy-Decreasing-Size algorithm (BDS) presented in [5] for the problem of Dynamic Storage Allocation, achieves an approximation ratio of 4 for the SA problem in stars. The BDS operates as follows. Given an instance $(S, \mathcal{R})$ of SA on a star S, the demand of each request $r \in \mathcal{R}$ is increased from d_r to d'_r the closest power of 2 of d_r. The requests are afterwards ordered in the decreasing order of their new demands. Finally, spectrum is allocated to requests in this order and with the new demands. Nota that any spectrum assignment to the requests with the new demands is a spectrum assignment to the requests with their original demands. Let l be the load of the instance $(S, \mathcal{R})$. We prove in the following that BDS uses at most $4l$ slots.

After increasing the demands of the requests, the new load is at most $2l$. 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 BDS. We prove that for each $i \in \{1, q\}$, $g(r_i) = 1$ or all slots $s < g(r_i)$ are occupied by requests conflicting with r_i (recall that a slot s is occupied by a request r if $s \in [g(r), g(r) + d'_r - 1]$). We proceed by induction. For $i = 1$, we have $g(r_i) = 1$. We suppose that for each $i < j$ either $g(r_i) = 1$ or all slots $s < g(r_i)$ are occupied by requests conflicting with r_i. If no request conflicting with r_j has been assigned spectrum then $g(r_j) = 1$. Otherwise, let s be the smallest slot not occupied by requests conflicting with r_j. We set $d'_{r_j} = 2^a$.

- 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 $b \geq a$. This implies that slots in $[1, 2^a - 1]$ are not occupied by requests conflicting with r_j and hence $g(r_j) = 1$.
- If $s \neq 1$, let r_l be the request conflicting with r_j occupying slot $s - 1$. If there is no other request conflicting with 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 $g(r_j) = s$.

Now if we take the request occupying the highest slot, we know that all slots below are used by the request itself or requests conflicting with it. The number of slots used by a request and requests conflicting with it is bounded by $4l$ since each request uses at most two links and the load on a link is at most $2l$. The approximation ratio follows.

□

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 [23] 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.

Scheduling tasks on multiprocessor systems ($P|f_{ix_j}C_{max}|$). Given 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 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 [11] and [15] 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. Theorem 5 follows. □

5 Spectrum Assignment in binary trees: approximability results

The problem of Spectrum Assignment in binary trees has been studied in [22]. It has been proven that SA can be approximated within a ratio of $log(k)$ where k is the number of requests. The proof is based on the equivalence between SA and the problem of Interval Coloring. Using this equivalence, we give in this section some approximability results for the problem of Spectrum assignment in binary trees with bounded requests. We first start by giving some definitions and presenting the problem of interval coloring.

5.1 Definitions

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 $w(v_r)$ of each vertex v_r is equal to the bandwidth demand of the corresponding request r ($w(v_r) = d_r$). We define the weight of a subset $S \subset V$ to be the quantity $w(S) = \sum_{v \in S} w(v)$.

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 density of $G = (V, E, w)$ is the weight of the maximum weight clique and is denoted by $d(G, w)$.

Interval (or Contiguous) Coloring (IC) [12]. An interval coloring of a 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 $[f(v), f(v)+w(v)-1] \cap [f(v'), f(v')+w(v')-1] = \emptyset$. The number of colors used by an interval coloring f, denoted by $\chi_f(G, w)$ is the smallest integer b such that for each vertex $v \in V$, $f(v) + w(v) - 1 \leq b$. The interval chromatic number of a weighted graph (G, w), denoted by $\chi(G, w)$, is the least number of colors needed to color the vertices with intervals, i.e. it is the minimum $\chi_f(G, w)$ 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 $G = (V, E, w)$, 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 $s(N, \mathcal{R}) = \chi(G, w)$..

The conflict graph of an instance $(N, \mathcal{R})$ where N is a binary tree corresponds to a edge intersection graph of paths in a binary tree. These graphs have been proved to be chordal graphs in [14, 13]. 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 if every cycle of it 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 chordal 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 $G[\{x_{i+1}, \ldots, x_n\}]$ form a clique (for $S \subseteq V$, we define $G[S]$ as the subgraph of G induced by the vertices of S). Paper [24] 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.

The ***greedy algorithm*** for IC, also called the ***first fit algorithm*** 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 $\{v_1, \ldots, v_{i-1}\}$ such that v_i and v_j adjacent.

The problem of interval coloring of chordal graphs has been studied before. In [21], several $O(log(n))$-approximation algorithms are presented for IC in chordal graphs. The techniques used are the greedy algorithm in the decreasing order of the size and partitioning of the vertices into sets of similar weight. The experiments ran in [21] show that algorithms that perform well for the problem of "max-coloring" generally perform well for IC as well. In [6], a 2-approximation algorithm for IC is presented for a special class of chordal graphs; namely the claw-free chordal graphs. To find the approximation, the authors use 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 [21]).

5.2 Approximation results for bounded demands

In the remainder of this section section, we present constant-factor approximation results for SA in binary 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 [22]. We use this equivalence to prove our results. In fact, for each case, we state the theorems for the SA problem but we prove them using lemmas for IC in chordal graphs. In fact, for IC in chordal graphs, we prove even stronger results that the ones we deduce for SA. Besides the upper bounds we obtain on the ratio $\frac{\chi(G,w)}{d(G,w)}$ for chordal graphs and which allow to have the constant approximation, we are able to prove some lower bound on the same ratio for some cases. The results we prove for IC in chordal graphs with respect to upper and lower bounds on $\frac{\chi(G,w)}{d(G,w)}$ for weighted chordal graphs with bounded weights are summarized in Table 1.

Weights	Upper Bound	Lower Bound
$\{k, kX\}$, $k, X \in \mathbb{N}^*$	$2 - \frac{1}{X} + \epsilon$	$2 - \frac{1}{X} - \epsilon$
$\{kX, k(X+1)\}$, $k, X \in \mathbb{N}^*$	$1 + \frac{1}{X} + \epsilon$	
≤ 3	$\frac{19}{10} + \epsilon$	$\frac{3}{2}$
≤ 4	$\frac{67}{30} + \epsilon$	$\frac{3}{2}$
≤ 5	$\frac{659}{240} + \epsilon$	$\frac{5}{3}$
≤ 6	$\frac{603}{200} + \epsilon$	$\frac{5}{3}$
$\leq W$	$2log_2(W)$	

Table 1: Summary of the bounds obtained on the ratio $\frac{\chi}{d}$ for bounded weights

5.2.1 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.

Theorem 6. *There is a $2-\frac{1}{X}+\epsilon$-approximation algorithm for SA in binary trees when the demands of the requests are in the set $\{k, kX\}$.*

To prove Theorem 6 we prove the following lemma.

Lemma 1. *Let (G, w) be a weighted graph with weights in the set $\{k, kX\}$. There is a polynomial time algorithms that finds an interval coloring of (G, w) which uses at most $(2-\frac{1}{X})d(G, w)+k$ 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'). This means that if we can color (G, w') with $ad((G, w'))+b$ colors, then we can color (G, w) with $ad((G, w))+b$ colors. Therefore, we only need to prove the lemma for $k = 1$.

For interval graphs in [20], there is an algorithm to find a $2-\frac{1}{X}$-approximation for the problem of interval coloring whenever we have only two weights 1 and X. We generalize if for chordal graphs as follows.

Let (G, w) be a weighted chordal graph with weights in $\{1, X\}$ and let $d = d(G, w)$ be its density. We will use $2d - \lfloor \frac{d}{X} \rfloor$ colors to color (G, w) as follows. We partition the colors into two blocks. The first block B_1 from color 1 to color d and the second block B_2 from color $d+1$ to color $2d - \lfloor \frac{d}{X} \rfloor$.

We order the vertices of G in the reverse perfect elimination order. Let $v_1, \ldots, v_n$ be the obtained ordering. Recall that neighbors of v_i in $\{v_1, \ldots, v_{i-1}\}$ for a clique in the graph induced by $\{v_1, \ldots, v_{i-1}\}$. We use the greedy algorithm to assign colors to the vertices in this order such that colors assigned to a vertex are either included in B_1 or B_2 and cannot be between the two blocks.

We prove that with this algorithms, all vertices will be assigned colors in B_1 or B_2.

- All vertices of weight 1 will have color in B_1. In fact, if a vertex v_j of color 1 cannot be assigned a color in B_1, then its neighbors in $\{v_1, \ldots, v_{j-1}\}$ occupy all colors of B_1. This implies that the density of the graph is at least $d+1$ which is not possible.

- For vetices of weight X, suppose that there is a vertex v_j of weight X to which we could not assign colors in B_1 nor in B_2. The minimum number of colors used in B_1 that can make it not possible to color v_j with colors from B_1 is $\lfloor \frac{d}{X} \rfloor$ ($X-1$ free colors and then 1 occupied color then $X-1$ free colors and then 1 occupied color ...). The weight of the neighbors of v_j in $\{v_1, \ldots, v_{j-1}\}$ which use colors in B_1 is at least $\lfloor \frac{d}{X} \rfloor$. Since we cannot assign colors from B_2 to v_j and knowing that only vertices of the same weight X use colors from B_2 with the greedy algorithm, we deduce that the neighbors of v_j in $\{v_1, \ldots, v_{j-1}\}$ which use colors in B_2 is at least $d - \lfloor \frac{d}{X} \rfloor - (d-1)$. This implies that the density of G is at least $d+1$, which is not possible.

We have proved that when the weights are in the set $\{1, X\}$, we can color the graph with $(2-\frac{1}{X})d(G, w)+1$ colors. This implies that when the weights are in the set $\{k, kX\}$, we can color the graph with $(2-\frac{1}{X})d(G, w)+k$ colors. □

Lemma 2. *We have a family of graphs G_l, with weights k and kX, for which the ratio between the interval chromatic number and the density goes to $2-\frac{1}{X}$ when l goes to infinity.*

Proof. For $l > 0$, we build the following graph:

- X^2l+1 vertices of weight k forming a "big" clique.

- For each subset S of $Xl+1$ vertices of the big clique, we add $(X-1)l$ new vertices of weight kX each that form a clique with the vertices of S

The density of this graph has weight $k(X^2l+1)$

Let us try to color this graph. There exists an integer λ in $\{0, \ldots, kX-1\}$ such that the "big" clique uses $Xl+1$ colors which are 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 Xl colors which are congruent to i modulo kX. That means that the number of colors used is at most kX^2l. This is not possible since the maximum clique has weight $k(X^2l+1)$. Let S be the subset of vertices of the big clique using colors that are congruent to λ modulo kX. Vertices of S form a clique with $(X-1)l$ vertices of weight kX, each of which uses a color which is congruent to λ. In total, $(2X-1)l+1$ colors which are congruent to λ are used. This means that the number of colors used is at least $kX(2X-1)l+1$. The ratio between the chromatic number and the density is at least $\frac{kX(2X-1)l+1}{k(X^2l+1)} = 2-\frac{1}{X}-\frac{(2k-1)X-k}{X(X^2l+1)}$. When l goes to infinity, this ratio goes to $2-\frac{1}{X}$ □

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

Theorem 7. *There is a $\frac{X+1}{X}+\epsilon$-approximation algorithm for SA in binary trees when the demands of the requests are in the set $\{kX, k(X+1)\}$.*

To prove Theorem 7, we prove the following lemma.

Lemma 3. *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 $\frac{X+1}{X}d(G,w)+k(X+1)$ colors.*

Proof. Let (G,w) be a weighted chordal graph with weights in $\{kX, k(X+1)\}$. We set $m = \lceil \frac{d(G,w)}{kX} \rceil$. We prove that we can color (G,w) with $k(X+1)m$ colors. Let us take consider the interval of colors $\{1,\ldots,k(X+1)m\}$. We partition this interval into m contiguous intervals I_i of size $k(X+1)$ each ($I_i = \{k(X+1)(i-1)+1, k(X+1)i\}$). We order the vertices of (G,w) in the RPEO. Let $v_1,\ldots,v_n$ be the obtained ordering. We use the greedy algorithm in this order to try to color the vertices with color from exactly one of the intervals I_i. We check the intervals in the ascending order of i. To prove that all vertices are colored, let us suppose that a vertex v_j cannot be colored. This means that each interval I_i is used to color a neighbor of v_j with weight at least kX (recall that the weights are either kX or $k(X+1)$). These neighbors form a clique since we are working with the RPEO. This implies that there is a clique with weight at least $kXm+kX$ which is not possible. □

5.2.3 Maximum demand D

The approximation algorithm proposed for Interval Coloring in chordal graphs in [21] allow to prove the following.

Theorem 8. *There is a $2log_2(D)$-approximation for SA where D is the maximum demand.*

To prove Theorem 8, we prove the following lemma.

Lemma 4. *Let (G,w) be a weighted cgordal graph with maximum weight W. There is a polynomial time algorithm that finds an interval coloring of (G,w) which uses at most $2log_2(W)d(G,w)$ colors.*

Proof. The proof is due to [21] which propose a $2log_2(n)$-approximation where n is the number of vertices. We only replace n by the maximum weights. We include it for completeness.

We prove that there is a $2log(W)$-approximation for interval coloring of chordal graphs where W is the maximum weight. Let $(G=(V,E),w)$ be a vertex-weighted chordal graph with maximum weight W. We set $k=\log_2(W)$. Let us partition the set of vertices V into k subsets S_i, $i\in\{1,\ldots,k\}$ such that for each vertex $v\in S_i$, $w(v)\in[\frac{W}{2^{i-1}},\frac{W}{2^i}]$. We ignore the weights 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). This coloring is easily 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 $\frac{W}{2^{i-1}}\omega(G_i)\leq d(G,w)$ which implies that $c\leq\sum_{i=1}^{k}2d(G,w)=2kd(G,w)$. The $2log(W)$-approximation follows. □

5.2.4 Maximum demand ≤ 6

In the previous subsection, a $2log(W)$-approximation algorithm for the SA problem in binary trees where the maximum demand is at most W has been presented. This approximation is achieved using a partitioning of the requests into subsets of close demands. This technique is used not only in binary trees but also in general graphs as a heuristic [25]. In what follows, we use different techniques to find better approximations for SA in binary trees for some given values of the maximum demand W. 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 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 9. *There are approximation algorithms for the problem of Spectrum Assignment in binary tree networks of factors $\frac{19}{10}+\epsilon$, $\frac{67}{30}+\epsilon$, $\frac{659}{240}+\epsilon$ and $\frac{603}{200}+\epsilon$ when the maximum request demand is bounded by 3,4,5 and 6, respectively.*

Again to prove the theorem, we prove an equivalent lemma for interval coloring.

Lemma 5. *There are approximation algorithms for the problem of interval coloring in chordal graphs of factors $\frac{19}{10}+\epsilon$, $\frac{67}{30}+\epsilon$, $\frac{659}{240}+\epsilon$ and $\frac{603}{200}+\epsilon$ when the maximum weight is bounded by 3,4,5 and 6, respectively.*

Proof. As in the previous sections, $d(G,w)$ refers to the density of the weighted graph (G,w). The density $d_v(G,w)$ with respect to a vertex v is the weight of the maximum clique of G containing v. It is straightforward that $d_v(G,w) \leq d(G)$ for each $v \in V$.

Let $\mathcal{C}(d,S)$ denote the set of instances of IC in which the graph is chordal, the density is less than 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 a maximum weight $W \in \{3,4,5,6\}$ before presenting each case in details.

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

- *Partitioning the vertices into multi-level blocks:* in this phase, the vertices are partitioned into blocks. In total, we will have for $i \in \{1,\ldots,W\}$, n_i level-i blocks $B_1^i,\ldots,B_{n_i}^i$ of density d_i (the density of a block is the density of the graph induced by the vertices it contains).

  We start with empty blocks B_j^i, $i \in \{1,\ldots,W\}$, $j \in \{1,\ldots,n_i\}$. We assign vertices to the blocks as follows. We first order the vertices in the RPEO and then in this order, for each vertex v_l, we check the blocks B_j^i in the ascending order of i and the ascending order of j and we add v_l to the first block B_j^i whose density does not exceed d_i after adding v_l.

  With the adequate choices of n_i and d_i, all vertices are assigned to one of the blocks. Furthermore, all vertices of size at most i are assigned to blocks of B_j^l where $l \leq i$.

- *Solving the problem of interval coloring for each block:* in this phase, the vertices of each block B_j^i are colored using an algorithm to solve instances with density d_i and weights in $S_i = \{i,\ldots,W\}$ (the possible weights of the vetices in B_j^i). This algorithm is designed to use not more than $c(d_i,S_i)$ colors.

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

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

The total number of colors depends on n_i and d_i. In fact, it only depends on d_i since we choose the n_i as follows. We set $n_1 = \lceil \frac{d(G,w)}{d_1} \rceil$ and for $i \in \{2, \ldots, W\}$, we set $n_i = \lceil \frac{d(G,w) - \sum_{j=1}^{i-1} (d_j - k_j^i) n_j}{d_i - k_i^i} \rceil$ such that for $i \in \{2, \ldots, W\}$ and $j \in \{1, \ldots, i\}$ k_j^i is the smallest linear combination of $\{j, \ldots, W\}$ such that $d_j - k_j^i < i$.

We prove that with our choices of n_i the algorithm does indeed verify the property which states that all vertices of weight at most i are assigned to blocks of B_j^l where $l \leq i$.

Proof. We do the proof by induction. For $i = 1$, let us suppose that a vertex v of weight 1 cannot be assigned to one of the blocks B_j^1 for $j \in \{1, \ldots, n_1\}$. This means that the density, with respect to v, of each block B_j^1 is exactly d_1. Moreover, since we are using the RPEO, we know that the neighbors of v which are in the blocks B_j^1 form a clique. This implies that $d_v(G, w) \geq d_1 n_1 + 1$. With our choice of n_1 we will have $d_v(G, w) > d(G, w)$, which is not possible. Now let us suppose that for a given i, there exists a vertex v of weight i that cannot be assigned to one of the blocks B_j^l for $l \leq i$ and $j \in \{1, \ldots, n_l\}$ (supposing that the induction assumption is true for $l < i$). This means that each block B_j^l has density at least $d_j - k_j^l$ with respect to vertex v. Since we use the RPEO, all neighbors of v in the blocks form a clique. This implies that $d_v(G, w) \geq \sum_{j=1}^{i} (d_j - k_j^i) n_j + i$ and then $d_v(G, w) > d(G, w)$ which is not possible. □

We will now address each case in details.

Maximum weight 3 Let us first compute $c(3, S)$ for some basic sets S.

- $c(3, \{1, 2, 3\}) = 4$.
  - We first prove that $c(3, \{1, 2, 3\}) \geq 4$. Let us consider the example presented in Figure 2 in which the density is 3 and the maximum weight is 2. The graph in the example 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.
  - The first fit algorithm in the RPEO uses at most 4 colors for each instance where the density is 3 and the maximum weight is 3.

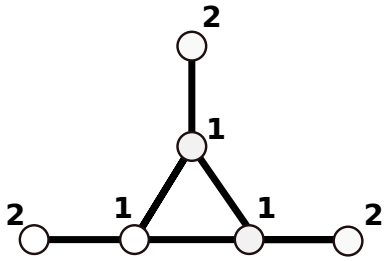


Figure 2: An example showing that $L(3, 3) \neq 3$

- $c(3, \{2, 3\}) = 3$. 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.
- $c(4, \{1, 2, 3\}) = 6$.
  - We first prove that $c(4, \{1, 2, 3\}) \geq 6$. Let us consider the example presented in Figure **??**. Suppose that we only use 5 colors $\{1, 2, 3, 4, 5\}$ to color it. 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 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.

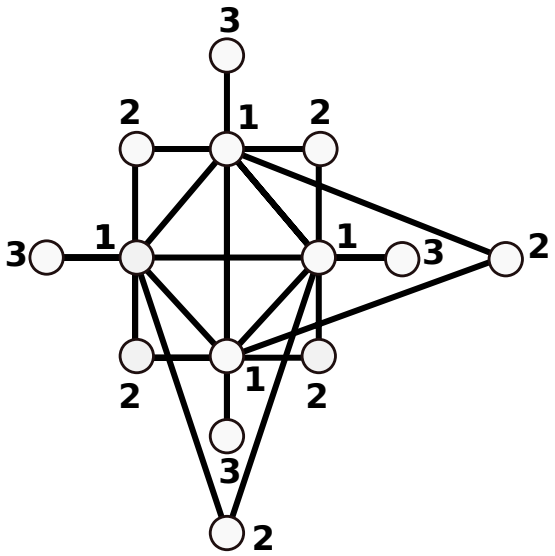


Figure 3: An example showing that $L(4, 3) \neq 5$

- $c(4, \{2, 3\}) = 4$. The first fit algorithm in the RPEO, colors any instance in $\mathcal{C}(4, \{2, 3\})$ with at most 4 colors.
- $c(5, \{1, 2, 3\}) = 7$.
  - We first prove that $c(5, \{1, 2, 3\}) \geq 7$. Let us consider the graph G consisting of a clique of 5 vertices of weight 1 each and such that each pair of two vertices of the clique are 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 one 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. This additional feature prevents any possible "fragmentation" scenario.
- $c(5, \{2, 3\}) = 5$. The first fit algorithm in the RPEO in which we forbid color 3 to vertices of weight 2 uses at most 5 colors.
- $c(6, \{1, 2, 3\}) = 9$.
  - We first prove that $c(6, \{1, 2, 3\}) \geq 9$. 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 $\mathcal{C}(6, \{1, 2, 3\})$ 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 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. In fact, when 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 neighbors of weight 2 and two other colored neighbors with weight 1, or one colored neighbor of weight 3 and another of weight 1, then one of the channels $\{1, 2\}$, $\{3, 4\}$, $\{5, 6\}$, or $\{7, 8\}$ is necessarily free to be used. If a vertex v of weight 3 is considered, there is no possible color assignment for the already colored neighbors that can make coloring v with the possible 9 colors impossible.

- $c(6, \{2, 3\}) = 7$.
  - We first prove that $c(6, \{2, 3\}) \geq 7$. Let us consider the graph presented in Figure 4. This graph cannot be colored using only six colors $\{1, \ldots, 6\}$. If we suppose that it can, then one of the vertices of weight one will have to use colors $\{3, 4\}$ and its neighbor of weight 3 will have no possible contiguous set of three colors.
  - Now let us describe an algorithm that takes an instance of $\mathcal{C}(6, \{2, 3\})$ and colors it with at 7 colors. The algorithm takes a set of 7 colors $\{1, 2, \ldots, 7\}$ and partitions it into two subsets $S_1 = \{1, 2, 3, 4\}$ and $S_2 = \{5, 6, 7\}$. Then, using first fit in the RPEO, we assign to each vertex colors from S_1 if it is possible, otherwise colors from S_2. We prove that all vertices can be colored with this algorithm. When a vertex v of weight 2 is considered, if v cannot be assigned colors from S_1, then the neighbors of v occupying colors from S_1 have weight at least 3. This means that no other neighbor of v is using colors from S_2 because otherwise the density will exceed 6 (recall that the vertices are considered in the RPEO and hence a new vertex forms a clique with its already colored neighbors). Similarly, When a vertex v of weight 3 is considered, if v cannot be assigned colors from S_1, then a of v of weight at least 2 is using colors from S_1 have weight at least 3. This means that no other neighbor of v is using colors from S_2 because otherwise the density will exceed 6.

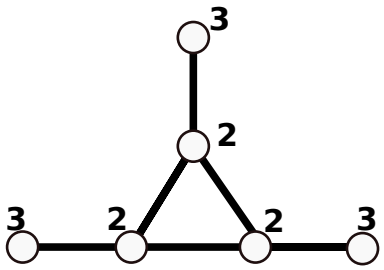


Figure 4: An example showing that $L(6, \{2, 3\}) \neq 6$

Now that we have solved some basic cases with maximum weight 3 and small densities, let us see some possible ways to partition a weighted chordal graph (G, w) with density $d(G, w)$ and maximum weight 3 into small groups with small densities in order to find good approximations.

- If we set $d_1 = d_2 = d_3 = 3$, then $n_1 = \lceil \frac{d(G,w)}{3} \rceil$, $n_2 = \lceil \frac{d(G,w)}{6} \rceil$, $n_3 = \lceil \frac{d(G,w)}{9} \rceil$ and the number of colors used is $n_1 c(3, \{1, 2, 3\}) + n_2 c(3, \{2, 3\}) + n_3 c(3, \{3\}) = 4n_1 + 3n_2 + 3n_3$ which is at most $\frac{13}{6} d(G, w) + 10$
- If we set $d_1 = 3$, $d_2 = 4$, and $d_3 = 3$, then $n_1 = \lceil \frac{d(G,w)}{3} \rceil$, $n_2 = \lceil \frac{d(G,w)}{9} \rceil$, $n_3 = \lceil \frac{4d(G,w)}{27} \rceil$ and the number of colors used is $n_1 c(3, \{1, 2, 3\}) + n_2 c(4, \{1, 2, 3\}) + n_3 c(3, \{3\}) = 4n_1 + 4n_2 + 3n_3$ which is at most $\frac{20}{9} d(G, w) + 11$

- If we set $d_1 = d_2 = 5$, and $d_3 = 3$, then $n_1 = \lceil \frac{d(G,w)}{5} \rceil$, $n_2 = \lceil \frac{d(G,w)}{20} \rceil$, $n_3 = \lceil \frac{d(G,w)}{12} \rceil$ and the number of colors used is $n_1 c(5, \{1,2,3\}) + n_2 c(5, \{1,2,3\}) + n_3 c(3, \{3\}) = 7n_1 + 5n_2 + 3n_3$ which is at most $\frac{19}{10} d(G,w) + 15$

- If we set $d_1 = d_2 = 6$, and $d_3 = 3$, then $n_1 = \lceil \frac{d(G,w)}{6} \rceil$, $n_2 = \lceil \frac{d(G,w)}{30} \rceil$, $n_3 = \lceil \frac{d(G,w)}{15} \rceil$ and the number of colors used is $n_1 c(6, \{1,2,3\}) + n_2 c(6, \{1,2,3\}) + n_3 c(3, \{3\}) = 9n_1 + 7n_2 + 3n_3$ which is at most $\frac{58}{30} d(G,w) + 19$

The best approximation obtained for maximum weight 3 is with a multiplicative ratio of $\frac{19}{10}$ and an additive constant of 15.

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

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

- $c(4, \{2,3,4\}) = 4$. A first fit algorithm in the RPEO uses at most 4 colors (in an instance of $\mathcal{C}(4, \{2,3,4\})$, vertices of weights 3 or 4 are isolated).

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

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

  - 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 two vertices of weight 1 are 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. If one of the vertices of weight 1 uses color 4, then its neighbor of weight 4 cannot be colored. This means that none of the vertices of weight 1 uses color 4 and implies that among the vertices of weight 1 either there are two vertices of using colors 2 and 5 or two vertices using colors 3 and 6. In both cases one vertex of weight 3 cannot be colored.
  - Now let us describe an algorithm that takes an instance of $\mathcal{C}5, \{1,2,3,4\})$ and colors it with at 8 colors. The algorithm uses first fit in the PREO 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. When a vertex v is considered then we have many cases. If v has weight 1 then there is necessarily a free color. If v has weight 4 then it possibly has a neighbor of weight 1 that has been already colored. Whatever color this neighbor occupies, there are 4 contiguous free colors for v. If v has weight 3, then it might have a colored vertex of weight 2 and then there are necessarily 3 contiguous free colors. Otherwise v has two neighbors of weight 1 which are already colored. Since vertices of weight 1 cannot use colors 3 and 6, there are 3 contiguous colors for v. If v has weight 2 then, it either has a neighbor of weight 3, one neighbor of weight 1 and another of weight 2, or three neighbors of weight 1, in all these cases 2 contiguous colors are necessarily free.

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

- $c(6, \{1,2,3,4\}) = 9$. Since $c(6, \{1,2,3\}) = 9$, we conclude $c(6, \{1,2,3,4\}) \geq 9$. The algorithm used to color a chordal graph with density 6 and maximum weight 3 can be used to color the graph if the maximum weight is 4. The number of colors used by the algorithm is still at most 9. 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 $\{1,2,3,4\}$.

- $c(6,\{2,3,4\}) = 8$.
  - We first prove that $c(6,\{2,3,4\}) \geq 8$. Let us consider the graph G which consists of a clique of 3 vertices of weight 3 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 wights in $\{2,3\}$. Let us suppose that we can color G with only 7 colors. In any possible coloring of the vetices of weight 2, a vertex v of weight 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 the possible combinations of colors for vertices of weight 2 are the following $\{1,2\}$, $\{3,4\}$, $\{5,6\}$, and $\{7,8\}$.
- $c(6,\{3,4\}) = 6$. 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 that we have solved some basic cases with maximum weight 4 and small densities, let us see some possible ways to partition a weighted chordal graph (G,w) with density $d(G,w)$ and maximum weight 4 into small groups with small densities in order to find good approximations.

- If we set $d_1 = d_2 = d_3 = d_4 = 4$, then $n_1 = \lceil \frac{d(G,w)}{4} \rceil$, $n_2 = \lceil \frac{d(G,w)}{12} \rceil$, $n_3 = \lceil \frac{d(G,w)}{9} \rceil$, $n_4 = \lceil \frac{d(G,w)}{16} \rceil$ and the number of colors used is $n_1 c(4,\{1,2,3,4\}) + n_2 c(4,\{2,3,4\}) + n_3 c(4,\{3,4\}) + n_4 c(4,\{4\}) = 6n_1 + 4n_2 + 4n_3 + 4n_4$ which is at most $\frac{91}{36} d(G,w) + 18$.
- If we set $d_1 = d_2 = 5$ and $d_3 = d_4 = 4$, then $n_1 = \lceil \frac{d(G,w)}{5} \rceil$, $n_2 = \lceil \frac{d(G,w)}{20} \rceil$, $n_3 = \lceil \frac{d(G,w)}{12} \rceil$, $n_4 = \lceil \frac{d(G,w)}{16} \rceil$ and the number of colors used is $n_1 c(5,\{1,2,3,4\}) + n_2 c(5,\{2,3,4\}) + n_3 c(4,\{3,4\}) + n_4 c(4,\{4\}) = 8n_1 + 5n_2 + 4n_3 + 4n_4$ which is at most $\frac{73}{30} d(G,w) + 21$.
- If we set $d_1 = d_2 = d_3 = 6$, $d_4 = 4$, then $n_1 = \lceil \frac{d(G,w)}{6} \rceil$, $n_2 = \lceil \frac{d(G,w)}{30} \rceil$, $n_3 = \lceil \frac{d(G,w)}{20} \rceil$, $n_4 = \lceil \frac{d(G,w)}{16} \rceil$ and the number of colors used is $n_1 c(6,\{1,2,3,4\}) + n_2 c(6,\{2,3,4\}) + n_3 c(6,\{3,4\}) + n_4 c(4,\{4\}) = 9n_1 + 8n_2 + 6n_3 + 4n_4$ which is at most $\frac{139}{60} d(G,w) + 27$.
- If we set $d_1 = d_2 = 6$ and $d_3 = d_4 = 4$, then $n_1 = \lceil \frac{d(G,w)}{6} \rceil$, $n_2 = \lceil \frac{d(G,w)}{30} \rceil$, $n_3 = \lceil \frac{d(G,w)}{15} \rceil$, $n_4 = \lceil \frac{d(G,w)}{20} \rceil$ and the number of colors used is $n_1 c(6,\{1,2,3,4\}) + n_2 c(6,\{2,3,4\}) + n_3 c(4,\{3,4\}) + n_4 c(4,\{4\}) = 9n_1 + 8n_2 + 4n_3 + 4n_4$ which is at most $\frac{67}{30} d(G,w) + 25$.

The best approximation obtained for maximum weight 4 is with a multiplicative ratio of $\frac{67}{30}$ and an additive constant of 25.

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

- $c(5,\{1,\ldots,5\}) = 8$. 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.
- $c(5,\{2,\ldots,5\}) = 5$. 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 $c(5,\{2,3\}) = 5$.
- $c(5,\{3,4,5\}) = 5$. All vertices are isolated and can be easily colored.
- $c(5,\{4,5\}) = 5$. All vertices are isolated and can be easily colored.
- $c(6,\{1,\ldots,5\}) = 10$.

- 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 each 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. No vertex of weight 1 can use color 5 because otherwise its neighbor of weight 5 cannot be colored. In any possible coloring of the vertices of weight 1 without using color 5, there are two vertices whose neighbor of weight 4 cannot be colored.
- Now, let us describe the algorithm that takes an instance of $\mathcal{C}(6, \{1, \ldots, 5\})$ and colors it with at most 10 colors. The algorithm uses first fit in the RPEO with the additional feature of forbidding colors $\{7, 8, 9, 10\}$ to vertices of weight 1.

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

- $c(6, \{3, 4, 5\}) = 6$. Vertices of weight 4 and 5 are isolated and vertices of weight 3 can be colored using first fit in the RPEO with at most 6 colors.

Now that we have solved some basic cases with maximum weight 5 and small densities, let us see some possible ways to partition a weighted chordal graph (G, w) with density $d(G, w)$ and maximum weight 5 into small groups with small densities in order to find good approximations.

- If we set $d_1 = d_2 = d_3 = d_4 = d_5 = 5$, then $n_1 = \lceil \frac{d(G,w)}{5} \rceil$, $n_2 = \lceil \frac{d(G,w)}{20} \rceil$, $n_3 = \lceil \frac{d(G,w)}{12} \rceil$, $n_4 = \lceil \frac{d(G,w)}{16} \rceil$, $n_5 = \lceil \frac{d(G,w)}{20} \rceil$, and the number of colors used is $n_1 c(5, \{1, \ldots, 5\}) + n_2 c(5, \{2, 3, 4, 5\}) + n_3 c(5, \{3, 4, 5\}) + n_4 c(5, \{4, 5\}) + n_5 c(5, \{5\}) = 8n_1 + 5n_2 + 5n_3 + 5n_4 + 5n_5$ which is at most $\frac{679}{240} d(G, w) + 28$.

- If we set $d_1 = d_2 = d_3 = 6$,$d_4 = d_5 = 5$, then $n_1 = \lceil \frac{d(G,w)}{6} \rceil$, $n_2 = \lceil \frac{d(G,w)}{30} \rceil$, $n_3 = \lceil \frac{d(G,w)}{20} \rceil$, $n_4 = \lceil \frac{d(G,w)}{16} \rceil$, $n_5 = \lceil \frac{d(G,w)}{25} \rceil$, and the number of colors used is $n_1 c(6, \{1, \ldots, 5\}) + n_2 c(6, \{2, 3, 4, 5\}) + n_3 c(6, \{3, 4, 5\}) + n_4 c(5, \{4, 5\}) + n_5 c(5, \{5\}) = 10n_1 + 8n_2 + 6n_3 + 5n_4 + 5n_5$ which is at most $\frac{659}{240} d(G, w) + 34$.

The best approximation obtained for maximum weight 5 is with a multiplicative ratio of $\frac{659}{240}$ and an additive constant of 34.

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.

- $c(6, \{1, \ldots, 6\}) = 10$.

- $c(6, \{2, \ldots, 6\}) = 8$.

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

If we set $d_1 = d_2 = d_3 = 6 = d_4 = d_5 = d_6 = 6$, then $n_1 = \lceil \frac{d(G,w)}{6} \rceil$, $n_2 = \lceil \frac{d(G,w)}{30} \rceil$, $n_3 = \lceil \frac{d(G,w)}{20} \rceil$, $n_4 = \lceil \frac{d(G,w)}{16} \rceil$, $n_5 = \lceil \frac{d(G,w)}{25} \rceil$, $n_6 = \lceil \frac{d(G,w)}{36} \rceil$,, and the number of colors used is $n_1 c(6, \{1, \ldots, 6\}) + n_2 c(6, \{2, \ldots, 6\}) + n_3 c(6, \{3, \ldots, 6\}) + n_4 c(5, \{4, 5, 6\}) + n_5 c(5, \{5, 6\}) + n_6 c(6, 6) = 10n_1 + 8n_2 + 6n_3 + 6n_4 + 6n_5 + 6n_6$ which is at most $\frac{603}{200} d(G, w) + 42$.

For maximum weight 6, we obtain an approximation is with a multiplicative ratio of $\frac{603}{200}$ and an additive constant of 42. □

6 Conclusion

To find approximation algorithms for interval coloring in chordal graphs in general and to SA in binary trees in particular 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 [21]. 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. This bound is at most $O(\log(n))\chi(G)$ [16].

- 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).

- 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.

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] 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 Berlin 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 Berlin 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 (February 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. Edge and vertex intersection of paths in a tree. *Discrete Mathematics 55*, 2 (1985), 151 – 159.

[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. Third European Conference on Combinatorics Graph Theory and Applications Third European Conference on Combinatorics.

[15] 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.

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

[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 Berlin 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. *J. Exp. Algorithmics 10* (Dec. 2005).

[22] 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.

[23] 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.

[24] 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.

[25] Wang, R., and Mukherjee, B. Spectrum management in heterogeneous bandwidth optical networks. *Optical Switching and Networking 11, Part A*, 0 (2014), 83 – 91.