Chapter 10

Channel Assignment in Wireless Local Area Networks

Alan A. Bertossi and M. Cristina Pinotti

10.1 Introduction

The tremendous growth of wireless networks requires an efficient use of the scarce radio spectrum allocated to wireless communications. However, the main difficulty against an efficient use of radio spectrum is given by *interferences*, caused by unconstrained simultaneous transmissions, which result in damaged communications that need to be retransmitted leading to a higher cost of the service. Interferences can be eliminated (or at least reduced) by means of suitable *channel assignment* techniques. Indeed, co-channel interferences caused by frequency reuse is one of the most critical factors on the overall system capacity in the wireless networks. The purpose of channel assignment algorithms is to make use of radio propagation loss characteristics to increase the radio spectrum reuse efficiency and thus to reduce the overall cost of the service.

The channel assignment algorithms partition the given radio spectrum into a set of disjoint channels that can be used simultaneously by the stations while maintaining acceptable radio signals. By taking advantage of physical characteristics of the radio environment, the same channel can be reused by two stations at the same time without interferences (*co-channel stations*), provided that the two stations are spaced sufficiently apart.

The minimum distance at which co-channels can be reused with no interferences is called *co-channel reuse distance*.

The interference phenomena may be so strong that even different channels used at near stations may interfere if the channels are too close. Since perfect filters are not available, interference between close frequencies is a serious problem, which can be handled either by adding guard frequencies between adjacent channels or by imposing channel separation. In this latter approach, followed in the present chapter, channels assigned to near stations must be separated by a gap on the radio spectrum—counted in a certain number of channels—which is inversely proportional to the distance between the two stations. In other words, the channels $f(u)$ and $f(v)$ assigned to the stations u and v at distance i must verify $|f(u)-f(v)| \geq \delta_i$ when a minimum *channel separation* δ_i is required between stations at distance i. The purpose of channel assignment algorithms is to assign channels to transmitters in such a way that the co-channel reuse distance and channel separation constraints are verified and the difference between the highest and lowest channels assigned is kept as small as possible.

Formally, the channel assignment problem can be modeled as an appropriate coloring problem on an undirected graph $G=(V,E)$ representing the wireless network topology, whose vertices in V correspond to stations, and edges in E correspond to pairs of stations whose transmission areas intersect. Specifically, given a vector $(\delta_1, \delta_2, \ldots, \delta_t)$ of nonincreasing positive integers, and an undirected graph $G=(V,E)$, an $L(\delta_1, \delta_2, \ldots, \delta_t)$ coloring of G is a function f from the vertex set V to the set of nonnegative integers $\{0, \ldots, \lambda\}$ such that $|f(u)-f(v)| \geq \delta_i$ whenever $d(u,v)=i$, $1 \leq i \leq t$, where $d(u,v)$ is the distance (i.e., the minimum number of edges) between the vertices u and v. An *optimal* $L(\delta_1, \delta_2, \ldots, \delta_t)$ coloring for G is one minimizing λ over all such colorings. Note that the co-channel reuse distance is actually $t+1$ but one need not explicitly state the co-channel reuse constraint because it is implied by the channel separation constraints. Note also that, since the set of colors includes 0, the overall number of colors involved by an optimal coloring f is in fact $\lambda+1$ (although, due to the channel separation constraint, some colors in $\{1, \ldots, \lambda-1\}$ might not be actually assigned to any vertex). Thus, the channel assignment problem consists of finding an optimal $L(\delta_1, \delta_2, \ldots, \delta_t)$ coloring for G.

When the separation vector $(\delta_1, \delta_2, \ldots, \delta_t)$ is equal to $(1, 1, \ldots, 1)$, the channel assignment problem has been widely studied in the past.[1–5] In particular, the intractability of optimal $L(1, \ldots, 1)$ coloring, for any positive integer t, has been proved by McCormick.[4] Optimal $L(1, \ldots, 1)$ colorings, for any positive integer t, have been proposed for rings, bidimensional grids, and honeycomb grids,[2,6] and for trees and interval graphs.[7] Moreover, when the separation vector is equal to $(\delta_1, 1, \ldots, 1)$, optimal $L(\delta_1, 1, \ldots, 1)$ colorings have been proposed for rings, bidimensional grids,

and cellular grids.[8,9] Optimal solutions have been proposed for the $L(\delta_1, \delta_2)$ coloring problem on rings[10] and on bidimensional and cellular grids.[11] This latter paper provided also an optimal $L(2, 1, 1)$ coloring for bidimensional grids. The $L(2, 1, 1)$ coloring problem has been also optimally solved for cellular grids, honeycomb grids, and rings.[8] The $L(2, 1)$ coloring has been investigated.[12–15] Bodlaender et al.[12] proved that the $L(2, 1)$ coloring problem is *NP*-hard for planar, bipartite, and chordal graphs, and presented approximate solutions for outerplanar, permutation and split graphs. Moreover, $L(2,1)$ colorings for unit interval graphs and trees have been found, respectively, by Sakai[15] and by Chang and Kuo.[13] A recent annotated bibliography on the $L(\delta_1, \delta_2)$ coloring problem for several special classes of graphs can be found in the literature.[16]

As a related case, when $(\delta_1, \delta_2) = (0, 1)$, the $L(0, 1)$ coloring problem models that of avoiding only the so-called *hidden interferences*, due to stations which are outside the hearing range of each other and transmit to the same receiving station. Optimal $L(0, 1)$ colorings have been provided for bidimensional grids,[17] whereas the intractability of optimal $L(0, 1)$ coloring has been proved,[18] where also optimal solutions for rings and complete binary trees were given.[18] As another related case, observe that the classical minimum vertex coloring problem on undirected graphs arises when $t = 1$ and $\delta_1 = 1$. Thus, the minimum vertex coloring problem consists in finding an optimal $L(1)$ coloring.

For arbitrary graphs and general separation vectors, the $L(\delta_1, \delta_2, \ldots, \delta_t)$ coloring problem is usually addressed by means of heuristic approaches, like genetic algorithms, taboo search, saturation degree, simulated annealing, and ants heuristics, just to name a few.[19] However, approximation algorithms have been proposed for trees and interval graphs.[20]

This chapter deals with the channel assignment problem for general separation vectors on some relevant classes of graphs—rings, grids, trees, and interval graphs—which occur in modeling realistic wireless network topologies. Indeed, rings are perhaps the most used topologies for local area networks, grids represent tessellations of the plane with regular polygons (like squares or triangles), trees model hierarchical topologies, and interval graphs model wireless networks serving narrow surfaces, like highways or valleys confined by natural barriers (e.g., mountains or lakes). It is still unknown whether finding optimal $L(\delta_1, \ldots, \delta_t)$ colorings on such classes of graphs is polynomially time-solvable or not. While the $L(\delta_1, \delta_2)$ coloring problem can be solved in polynomial time grids[11] and rings,[10] some authors conjecture that it is *NP*-hard for trees and unit interval graphs when $\delta_2 > 1$.[16]

In the rest of this chapter, several algorithms are reviewed.[8,11,14,20,21] First, some preliminary results useful to derive upper and lower bounds on the largest color needed are summarized in Section 10.2, where the notions of t-simplicial and strongly simplicial vertices are also recalled. Then, Section 10.3 presents optimal $O(n)$ time algorithms for solving the $L(2, 1)$,

$L(2,1,1)$, and $L(\delta_1,1,\ldots,1)$ coloring problems of rings with n vertices. Section 10.4 revises optimal algorithms for the $L(\delta_1,\delta_2)$ coloring problem on both bidimensional and cellular grids, and for the $L(\delta_1,1,\ldots,1)$ coloring problem on bidimensional grids. All the algorithms are based on arithmetic progressions and take $O(rc)$ time to color the entire grid of r rows and c columns. Sections 10.5 and 10.6 present two polynomial time algorithms to approximate the $L(\delta_1,\ldots,\delta_t)$ coloring problem on interval graphs and trees, respectively. The algorithms give the same constant approximation, which depends on t and $\delta_1,\ldots,\delta_t$, but they run in $O(n(t+\delta_1))$ and $O(nt^2\delta_1)$ time, respectively, where n is the number of vertices. A better approximation for the $L(\delta_1,\delta_2)$ coloring of unit interval graphs is also given in Section 10.5.2. Finally, conclusions are offered in Section 10.7.

10.2 Preliminaries

Throughout this chapter, it is assumed that G is a connected undirected graph with at least two vertices and that the separations verify $\delta_1 \geq \delta_2 \geq \cdots \geq \delta_t$. When $\delta_1 = \delta_2 = \cdots = \delta_t = 1$, the $L(1,\ldots,1)$ coloring problem reduces to the classical vertex coloring problem on the tth *power* G^t of the graph G. The vertex set of G^t is the same as the vertex set of G, while the edge uv belongs to the edge set of G^t if and only if the distance $d(u,v)$ between the vertices u and v in G is at most t. Now, colors must be assigned to the vertices of G^t so that every pair of vertices connected by an edge are assigned different colors and the minimum number of colors is used. Hence, the role of *maximum cliques* in G^t is apparent for deriving lower bounds on the minimum number of channels for the $L(1,\ldots,1)$ coloring problem on G. A *clique* K for G^t is a subset of vertices of G^t such that for each pair of vertices in K there is an edge. Clearly, a clique of size k in the power graph G^t implies that at least k different colors are needed to color G^t. In other words, the size of the largest clique in G^t is a lower bound for the $L(1,\ldots,1)$ coloring problem on G. Of course, a lower bound for the $L(1,\ldots,1)$ coloring problem is also a lower bound for the $L(\delta_1,1,\ldots,1)$ coloring problem, with $\delta_1 \geq 1$.

A simple lower bound for the $L(2,1)$ coloring problem, which can be applied to the vertex with maximum degree of any graph, can be derived as follows. Consider the *star* graph S_ρ which consists of a *center* vertex v with degree ρ, and ρ *ray* vertices of degree 1.

Lemma 10.1[14] *Let the center v of S_ρ be already colored. Then, the largest color λ required for an $L(2,1)$ coloring of S_ρ is at least*

$$\lambda = \begin{cases} \rho+1 & \text{if } f(v)=0 \text{ or } f(v)=\rho+1 \\ \rho+2 & \text{if } 0<f(v)<\rho+1 \end{cases}$$

For any value of $t \leq |V|$, let $\lambda^*_{G,t}$ denote the minimum value of λ over all the $L(1,\ldots,1)$ colorings $f : V \to \{0,\ldots,\lambda\}$ of $G=(V,E)$. Note that: (i) $\lambda^*_{G,1} \geq 1$ since G is assumed to be connected and to have at least 2 vertices; (ii) $\lambda^*_{G,t} = \lambda^*_{G^t,1}$; and (iii) $\lambda^*_{G,t}+1$ is at least as large as the size ω_{G^t} of the largest clique of the power graph G^t.

Lemma 10.2[20] *Any $L(\delta_1,\delta_2,\ldots,\delta_t)$ coloring requires at least $\max_{1\leq j\leq t}\{\delta_j\lambda^*_{G,j}\}$ as the largest color.*

Proof Since $\delta_1 \geq \delta_2 \geq \cdots \geq \delta_t$, any $L(\delta_1,\delta_2,\ldots,\delta_j)$ coloring, for any value of $j \leq t$, requires at least as many colors as any $L(\delta_j,\delta_j,\ldots,\delta_j)$ coloring, which, in its turn, requires at least $\delta_j\lambda^*_{G,j}$ as the largest color. □

Given $G=(V,E)$, let S be a subset of V, and let $G[S]$ denote the subgraph of G induced by S, i.e. $G[S]=(S,\{uv \in E : u,v \in S\})$. A vertex x of G is called *t-simplicial* when, for every pair of vertices u and v such that $d(x,u) \leq t$ and $d(x,v) \leq t$, it holds also that $d(u,v) \leq t$. A vertex x is called *strongly simplicial* when x is t-simplicial for any value of t.

Lemma 10.3[21] *Given $G=(V,E)$ and an integer t, let v be a t-simplicial vertex of G. Consider $G'=G[V-\{v\}]$ and let f' be an optimal $L(1,\ldots,1)$ coloring of G' using $\lambda^*_{G',t}$ as the largest color. Define an $L(1,\ldots,1)$ coloring f of V extending f' to v so that*

$$f(x)=\begin{cases}\min\{i : i \neq f'(u) \text{ for each } u \in G' \text{ with } d(u,v) \leq t\} & \text{if } x = v,\\ f'(x) & \text{if } x \in V-\{v\}\end{cases}$$

Then f is an optimal $L(1,\ldots,1)$ coloring for G.

Note that verifying whether a vertex is t-simplicial or not can be done in polynomial time.[7] Therefore, Lemma 10.3 implies the existence of an algorithm that optimally solves in polynomial time the $L(1,\ldots,1)$ coloring problem using exactly ω_{G^t} colors for any class of graphs closed under taking induced subgraphs and with the property that every graph of that class has a t-simplicial vertex. The next lemma shows that there is always an $L(\delta_1,\delta_2,\ldots,\delta_t)$ coloring for such a class of graphs where the largest used color is bounded from above by a function of the clique sizes ω_{G^j} and of the separations δ_j, with $1 \leq j \leq t$.

Lemma 10.4[20] *Given $G=(V,E)$ and t, let v be a t-simplicial vertex of G, and consider $G'=G[V-\{v\}]$. Then, there is an $L(\delta_1,\delta_2,\ldots,\delta_t)$ coloring such that $f(v)=c$, where $c \in \{0,1,\ldots,\lambda^*_{G,t}+2(\delta_t-1)\lambda^*_{G,t}+\sum_{j=1}^{t-1} 2(\delta_j-\delta_{j+1})\lambda^*_{G,j}\}$.*

Proof Let $N_j(v) = \{u \in V : d(u,v) \leq j\}$. Since v is t-simplicial, any two vertices in $N_j(v)$ are at distance at most j, for every $1 \leq j \leq t$. Hence $N_t(v)$ is a clique of G^t, and thus $|N_t(v)| \leq \omega_{G^t}$. Therefore at most $|N_t(v)| - 1 \leq \lambda^*_{G,t}$ colors have been used for $N_t(v) - \{v\}$. Each of them forbids $2(\delta_t - 1)$ colors due to the δ_t-separation constraint, and overall $2(\delta_t - 1)\lambda^*_{G,t}$ colors are forbidden. Moreover, for any $1 \leq j \leq t-1$, v is j-simplicial, and hence $N_j(v)$ is a clique of G^j, and $|N_j(v)| - 1 \leq \lambda^*_{G,j}$. Since each color assigned to a vertex of $N_j(v) - \{v\}$ forbids $2(\delta_j - \delta_{j+1})$ colors, $2(\delta_j - \delta_{j+1})\lambda^*_{G,j}$ colors are forbidden due to the δ_j-separation constraint. Before coloring v, the total number of used and forbidden colors is $\lambda^*_{G,t} + 2(\delta_t - 1)\lambda^*_{G,t} + \sum_{j=1}^{t-1} 2(\delta_j - \delta_{j+1})\lambda^*_{G,j}$. Therefore, there is at least an available color c in $\{0, 1, \ldots, \lambda^*_{G,t} + 2(\delta_t - 1)\lambda^*_{G,t} + \sum_{j=1}^{t-1} 2(\delta_j - \delta_{j+1})\lambda^*_{G,j}\}$ that can be assigned to v. □

The results of Lemmas 10.1 and 10.2 will be used in Sections 10.3 and 10.4 to obtain lower bounds on the number of colors needed by any $L(2,1)$ coloring of rings and $L(\delta_1, 1, \ldots, 1)$ coloring of grids. Moreover, the properties stated in Lemmas 10.2–10.4 will be exploited in Sections 10.5 and 10.6 to derive lower and upper bounds for the $L(\delta_1, \delta_2, \ldots, \delta_t)$ coloring problem on interval graphs and trees.

10.3 Rings

A *ring* R of size $n \geq 3$ is a sequence of n vertices, indexed consecutively from 0 to $n-1$, such that vertex i is connected to both vertices $(i-1)$ mod n and $(i+1)$ mod n.

In this section, three algorithms are presented to optimally solve the $L(2,1)$-, $L(2,1,1)$- and $L(\delta_1, 1, \ldots, 1)$ coloring problems on rings. Each algorithm colors a single vertex in $O(1)$ time and thus colors the entire ring in $O(n)$ time.

By Lemma 10.1, the largest color used by any $L(2,1)$ coloring of a ring is at least 4, as one can easily check by observing that in a ring $\rho = 2$ and there is a vertex v that must be colored $0 < f(v) < 3$. An optimal solution has been provided by Griggs and Yeh,[14] who color each vertex i as follows:

$$f(i) = \begin{cases} 0 & \text{if } i \equiv 0 \mod 3 \\ 2 & \text{if } i \equiv 1 \mod 3 \\ 4 & \text{if } i \equiv 2 \mod 3 \end{cases}$$

However the above coloring is redefined on the ring tail depending on whether $n \equiv 1 \mod 3$ or $n \equiv 2 \mod 3$. In the first case, $f(n-4), \ldots, f(n-1)$

become

$$f(i) = \begin{cases} 0 & \text{if } i = n-4 \\ 3 & \text{if } i = n-3 \\ 1 & \text{if } i = n-2 \\ 4 & \text{if } i = n-1 \end{cases}$$

In the second case, $f(n-2)$ and $f(n-1)$ are modified as

$$f(i) = \begin{cases} 1 & \text{if } i = n-2 \\ 3 & \text{if } i = n-1 \end{cases}$$

For the $L(2,1,1)$ coloring problem on rings, the same lower bound previously discussed for the $L(2,1)$ coloring holds and an optimal coloring can be derived as follows. Let $n \geq 12$ and $\theta = 4\,(\lfloor \frac{n}{4} \rfloor - (n \bmod 4))$. Then assign to each vertex i the color:

$$f(i) = \begin{cases} 0 & \text{if } i \equiv 0 \bmod 4 \text{ and } i < \theta, \text{ or } (i-\theta) \equiv 0 \bmod 5 \text{ and } i \geq \theta \\ 1 & \text{if } i \equiv 2 \bmod 4 \text{ and } i < \theta, \text{ or } (i-\theta) \equiv 3 \bmod 5 \text{ and } i \geq \theta \\ 2 & \text{if } (i-\theta) \equiv 1 \bmod 5 \text{ and } i \geq \theta \\ 3 & \text{if } i \equiv 3 \bmod 4 \text{ and } i < \theta, \text{ or } (i-\theta) \equiv 4 \bmod 5 \text{ and } i \geq \theta \\ 4 & \text{if } i \equiv 1 \bmod 4 \text{ and } i < \theta, \text{ or } (i-\theta) \equiv 2 \bmod 5 \text{ and } i \geq \theta \end{cases}$$

The above algorithm colors the first θ vertices repeating $\lfloor \frac{n}{4} \rfloor - (n \bmod 4)$ times the color sequence 0,4,1,3 of length 4, while the remaining vertices are colored repeating $n \bmod 4$ times the sequence 0,2,4,1,3 of length 5. It is worth noting that, for $n < 12$, an optimal coloring requires a larger number of colors.[8]

10.3.1 Optimal $L(\delta_1,1,\ldots,1)$ coloring of Rings

In this subsection, an optimal $L(\delta_1, 1, \ldots, 1)$ coloring of rings is discussed. Assuming a sufficiently large ring, the following lower bound holds:

Lemma 10.5[2] *Given $n \geq t+2$, $t \geq 2$, and $\delta_1 \geq 1$, the largest color used by any $L(\delta_1, 1, \ldots, 1)$ coloring is at least* $t + \left\lceil \frac{n \bmod (t+1)}{\lfloor \frac{n}{t+1} \rfloor} \right\rceil$.

Proof Observe that any $L(\delta_1, \ldots, 1)$ coloring needs at least as many colors as any $L(1, \ldots, 1)$ coloring. One notes that in an $L(1, \ldots, 1)$ coloring each color may appear at most $\tau = \left\lfloor \frac{n}{t+1} \right\rfloor$ times, and thus at least $\lceil \frac{n}{\tau} \rceil$ colors are needed. Since $n = \left\lfloor \frac{n}{t+1} \right\rfloor (t+1) + (n \bmod (t+1))$, it follows

that at least $\lceil \frac{n}{\tau} \rceil = (t+1) + \lceil \frac{n \bmod (t+1)}{\lfloor \frac{n}{t+1} \rfloor} \rceil$ colors are required. Therefore, the largest color is at least $t + \lceil \frac{n \bmod (t+1)}{\lfloor \frac{n}{t+1} \rfloor} \rceil$. □

The optimal $L(\delta_1, 1, \ldots, 1)$ coloring algorithm for rings acts as follows. Let $\zeta = t + \lceil \frac{n \bmod (t+1)}{\lfloor \frac{n}{t+1} \rfloor} \rceil$. If $n \equiv 0 \bmod (t+1)$ then $\zeta = t$ results, and one assigns to each vertex i the color:

$$f(i) = \begin{cases} i\lfloor \frac{\zeta}{2} \rfloor \mod (t+1) & \text{if (} t \text{ is even) or } (t \text{ is odd and } \lfloor \frac{\zeta}{2} \rfloor \text{ is odd)} \\ i\lfloor \frac{\zeta}{2} \rfloor \mod (t+1) & \text{if } t \text{ is odd and } \lfloor \frac{\zeta}{2} \rfloor \text{ is even and } i \equiv h \bmod (t+1), \text{ for } 0 \le h \le \lfloor \frac{t}{2} \rfloor \\ (i\lfloor \frac{\zeta}{2} \rfloor + 1) \bmod (t+1) & \text{if } t \text{ is odd and } \lfloor \frac{\zeta}{2} \rfloor \text{ is even and } i \equiv h \bmod (t+1), \text{ for } \lceil \frac{t}{2} \rceil \le h \le t \end{cases}$$

If $n \not\equiv 0 \bmod (t+1)$, let $\theta = \lfloor \frac{n}{\lambda} \rfloor - n \bmod \zeta$. If $i \ge \theta\zeta$, then

$$f(i) = \begin{cases} (i-\theta\zeta)\lfloor \frac{\zeta}{2} \rfloor \bmod (\zeta+1) & \text{if } (\zeta \text{ is even) or } (\zeta \text{ is odd and } \lfloor \frac{\zeta}{2} \rfloor \text{ is odd)} \\ (i-\theta\zeta)\lfloor \frac{\zeta}{2} \rfloor \bmod (\zeta+1) & \text{if } \zeta \text{ is odd and } \lfloor \frac{\zeta}{2} \rfloor \text{ is even and } i \equiv h \bmod (\zeta+1), \text{ for } 0 \le h \le \frac{\zeta+1}{2} - 1 \\ \left((i-\theta\zeta)\lfloor \frac{\zeta}{2} \rfloor + 1\right) \bmod (\zeta+1) & \text{if } \zeta \text{ is odd and } \lfloor \frac{\zeta}{2} \rfloor \text{ is even and } i \equiv h \bmod (\zeta+1), \text{ for } \frac{\zeta+1}{2} \le h \le \zeta \end{cases}$$

otherwise (that is, when $i < \theta\zeta$)

$$f(i) = \begin{cases} 0 & \text{if } i \equiv 0 \bmod \zeta \\ (i+1)\lfloor \frac{\zeta}{2} \rfloor \bmod (\zeta+1) & \text{if } (\zeta \text{ is even) or } (\zeta \text{ is odd and } \lfloor \frac{\zeta}{2} \rfloor \text{ is odd) and } i \not\equiv 0 \bmod \zeta \\ (i+1)\lfloor \frac{\zeta}{2} \rfloor \bmod (\zeta+1) & \text{if } \zeta \text{ is odd and } \lfloor \frac{\zeta}{2} \rfloor \text{ is even and } i \equiv h \bmod \zeta, \text{ for } 1 \le h \le \frac{\zeta+1}{2} - 1 \\ \left((i+1)\lfloor \frac{\zeta}{2} \rfloor + 1\right) \bmod (\zeta+1) & \text{if } \zeta \text{ is odd and } \lfloor \frac{\zeta}{2} \rfloor \text{ is even and } i \equiv h \bmod \zeta, \text{ for } \frac{\zeta+1}{2} \le h \le \zeta \end{cases}$$

The above $L(\delta_1, 1, \ldots, 1)$ coloring algorithm assumes $t \geq 2$ and when $n \geq t+2$ works for any value of $\delta_1 \leq \lfloor \frac{\zeta}{2} \rfloor$. Note that when $t=2$ or when $t=3$ and $n \equiv 0 \bmod 4$, such an algorithm solves the $L(1,1)$- and $L(1,1,1)$-coloring problems, but not the $L(2,1)$- and $L(2,1,1)$ colorings because in such a case $2 \leq \zeta \leq 3$ and thus $\delta_1 = 1$. However, such cases are optimally solved by the previous algorithms.

The correctness and optimality of the algorithm is based on the following lemma:

Lemma 10.6[8] *Given $n \geq t+2$, and $t \geq 2$, let $\zeta = \sigma - 1 + \left\lceil \frac{n \bmod \sigma}{\lfloor \frac{n}{\sigma} \rfloor} \right\rceil$.*

- *If (ζ is even) or (ζ is odd and $\lfloor \frac{\zeta}{2} \rfloor$ is odd), then $i \lfloor \frac{\zeta}{2} \rfloor \bmod (\zeta+1)$ assumes all the values in the range $[0, \zeta]$ while i varies within the range $[0, \zeta]$.*
- *If (ζ is odd and $\lfloor \frac{\zeta}{2} \rfloor$ is even), then $i \lfloor \frac{\zeta}{2} \rfloor \bmod (\zeta+1)$ assumes all the even values in the range $[0, \zeta-1]$ while i varies within the range $[0, \frac{\zeta+1}{2} - 1]$.*

It is worth noting that the three algorithms seen in this section solve the $L(2, 1, \ldots, 1)$ coloring problem for any $t \geq 2$. In general, given any $\delta_1 \geq 1$, the $L(\delta_1, 1, \ldots, 1)$ coloring problem can be optimally solved for every $t \geq 2\delta_1$, independently on the size n of the ring and using as few colors as the $L(1, 1, \ldots, 1)$ coloring problem.[8] Nevertheless, the $L(\delta_1, 1, \ldots, 1)$ coloring problem may be also solved for values of t smaller than $2\delta_1$ for suitable values of $n < (2\delta_1 + 1)^2$.

10.4 Grids

A *bidimensional grid* B of size $r \times c$ has r rows and c columns, indexed respectively from 0 to $r-1$ (from top to bottom) and from 0 to $c-1$ (from left to right), with $r \geq 2$ and $c \geq 2$. A generic vertex u of B will be denoted by $u = (i, j)$, where i is its row index and j is its column index. All internal vertices, i.e., those not on the borders, have degree 4. In particular, an internal vertex $u = (i, j)$ is adjacent to the vertices $(i-1, j), (i, j+1), (i+1, j)$, and $(i, j-1)$.

A *cellular grid* C of size $r \times c$, with $r \geq 2$ and $c \geq 2$, is obtained from a bidimensional grid B of the same size augmenting the set of edges with left-to-right *diagonal* connections. Specifically, each vertex $u = (i, j)$ of C is also connected to the vertices $v = (i-1, j-1)$ and $z = (i+1, j+1)$. Hence, each vertex has degree 6, except for the vertices on the borders.

10.4.1 Optimal $L(\delta_1,\delta_2)$ coloring of Grids

Optimal solutions for the $L(\delta_1, \delta_2)$ coloring problems on both bidimensional and cellular grids have been provided by Van Den Heuvel et al.[11] They have shown that optimal $L(\delta_1, \delta_2)$ colorings can be obtained by *arithmetic progression*, namely, determining three nonnegative integers a, b, and m such that the color assigned to any vertex $u=(i, j)$ is calculated as $f(u)=f(i, j)=(ai+bj) \bmod m$, where $m-1$ is the largest used color.

Precisely, such parameters for bidimensional grids are $a=\delta_1$, $b=\delta_1+\delta_2$, and $m=2\delta_1+3\delta_2$, and thus

$$f(i, j) = (\delta_1 i + (\delta_1 + \delta_2)j) \bmod (2\delta_1 + 3\delta_2)$$

In contrast, for cellular grids, an optimal $L(\delta_1, \delta_2)$ colorings can be obtained by a different arithmetic progression

$$f(i, j) = \begin{cases} ((2\delta_1 + \delta_2)i - \delta_1 j) \bmod (4\delta_1 + 3\delta_2) & \text{if } \delta_1 \leq \frac{3}{2}\delta_2 \\ (5\delta_2 i - 2\delta_2 j) \bmod 9\delta_2 & \text{if } \frac{3}{2}\delta_2 \leq \delta_1 \leq 2\delta_2 \\ ((2\delta_1 + \delta_2)i - \delta_1 j) \bmod (3\delta_1 + 3\delta_2) & \text{if } \delta_1 \geq 2\delta_2 \end{cases}$$

Clearly, $O(1)$ time is spent to color a single vertex, and thus the overall time complexity of the algorithm is $O(n)$, where $n=rc$.

10.4.2 Optimal $L(\delta_1,1,\ldots,1)$ coloring of Bidimensional Grids

In this subsection, the $L(\delta_1, 1, \ldots, 1)$ coloring problem is dealt with. Although this problem can be optimally solved for both bidimensional grids[8] and cellular grids,[9] only the algorithm for bidimensional grids is shown here since it is considerably simpler to be described than its counterpart for cellular grids.

A lower bound is given by the following lemma.

Lemma 10.7[8] *If* $\delta_1 \leq \left\lceil \frac{(t+1)^2}{2} \right\rceil$, *any* $L(\delta_1, 1, \ldots, 1)$ *coloring of a bidimensional grid* B *of size* $r \times c$, *with* $r \geq t+1$ *and* $c \geq t+1$, *requires at least* $\left\lceil \frac{(t+1)^2}{2} \right\rceil - 1$ *as the largest color.*

Indeed, consider a generic vertex $x=(i, j)$ of B, and its opposite vertex at distance t on the same column, i.e., $y=(i-t, j)$. All the vertices of B at distance t or less from both x and y are mutually at distance t or less. Therefore, in the power graph B^t, they form a

clique of size $\left\lceil \frac{(t+1)^2}{2} \right\rceil$, and thus $\lambda^*_{B^t,1} \geq \left\lceil \frac{(t+1)^2}{2} \right\rceil - 1$. By Lemma 10.2, any $L(\delta_1, 1, \ldots, 1)$ coloring on B requires at least $\max\{\delta_1 \lambda^*_{B,1}, \lambda^*_{B^t,1}\} = \max\left\{\delta_1, \left\lceil \frac{(t+1)^2}{2} \right\rceil - 1\right\} = \left\lceil \frac{(t+1)^2}{2} \right\rceil - 1$ as the largest color.

The following optimal $L(\delta_1, 1, \ldots, 1)$ coloring is based on arithmetic progression and works for $t \geq 2$ and $\delta_1 \leq \lfloor \frac{t}{2} \rfloor$.

If t is even, then assign to each vertex $u = (i, j)$ the color:

$$f(i,j) = \left(\left(\frac{t}{2} + 1\right) i + \frac{t}{2} j\right) \bmod \left\lceil \frac{(t+1)^2}{2} \right\rceil$$

If t is odd, let $i' = i \bmod (t+1)$ and $j' = j \bmod (t+1)$.
If t is odd and $\lfloor \frac{t}{2} \rfloor$ is even, then

$$f(i,j) = \begin{cases} \lfloor \frac{t}{2} \rfloor (\lceil \frac{t}{2} \rceil i' + j') \bmod \frac{(t+1)^2}{2} & \text{if } 0 \leq i' \leq \lfloor \frac{t}{2} \rfloor \text{ and } 0 \leq j' \leq \lfloor \frac{t}{2} \rfloor \\ & \text{or } \lceil \frac{t}{2} \rceil \leq i' \leq t \text{ and } \lceil \frac{t}{2} \rceil \leq j' \leq t \\ \left(\lfloor \frac{t}{2} \rfloor (\lceil \frac{t}{2} \rceil i' + j') + 1\right) & \text{if } 0 \leq i' \leq \lfloor \frac{t}{2} \rfloor \text{ and } \lceil \frac{t}{2} \rceil \leq j' \leq t \\ \quad \bmod \frac{(t+1)^2}{2} & \text{or } \lceil \frac{t}{2} \rceil \leq i' \leq t \text{ and } 0 \leq j' \leq \lfloor \frac{t}{2} \rfloor \end{cases}$$

If both t and $\lfloor \frac{t}{2} \rfloor$ are odd, then

$$f(i,j) = \begin{cases} \lfloor \frac{t}{2} \rfloor (\lceil \frac{t}{2} \rceil t' + j') \bmod \frac{(t+1)^2}{2} & \text{if } 0 \leq i' \leq \lfloor \frac{t}{2} \rfloor \text{ and } 0 \leq j' \leq \lfloor \frac{t}{2} \rfloor \\ & \text{or } \lceil \frac{t}{2} \rceil \leq i' \leq t \text{ and } \lceil \frac{t}{2} \rceil \leq j' \leq t \\ \lfloor \frac{t}{2} \rfloor \left(\frac{(t+1)^2}{4} - 1 + \lceil \frac{t}{2} \rceil i' + j'\right) & \text{if } 0 \leq i' \leq \lfloor \frac{t}{2} \rfloor \text{ and } \lceil \frac{t}{2} \rceil \leq j' \leq t \\ \quad \bmod \frac{(t+1)^2}{2} & \text{or } \lceil \frac{t}{2} \rceil \leq i' \leq t \text{ and } 0 \leq j' \leq \lfloor \frac{t}{2} \rfloor \end{cases}$$

The correctness and optimality of the above coloring algorithm has been proved.[8] It is worth noting that arithmetic progression is followed on the whole grid when σ is odd. In contrast, when σ is even, the coloring covers the bidimensional grid B with a tessellation of basic tiles T of size $\sigma \times \sigma$, each consisting of four subtiles of size $\frac{\sigma}{2} \times \frac{\sigma}{2}$, and each following arithmetic progression.

Given any $\delta_1 \geq 1$, the above algorithm optimally solves the $L(\delta_1, 1, \ldots, 1)$ coloring problem for every $t \geq 2\delta_1$. As a consequence, given $\delta_1 = 2$, the algorithm solves the $L(2, 1, \ldots, 1)$ coloring problem for every $t \geq 4$. Hence, when $\delta_1 = 2$, the only values of t not covered by the algorithm are 2 and 3. However, in such cases, the $L(2, 1)$- and $L(2, 1, 1)$ coloring problems had been solved by Van Den Heuvel et al.[11] Therefore, the $L(2, 1, \ldots, 1)$ coloring problem can be optimally solved for any value of t.

10.5 Interval Graphs

A graph $G=(V,E)$ is termed an *interval graph* if it has an *interval representation*, namely, if each vertex of V can be represented by an interval of the real line such that there is an edge $uv \in E$ if and only if the intervals corresponding to u and v intersect. More formally, let the graph $G=(V,E)$ have n vertices. Two integers l_v and r_v, with $l_v < r_v$, (the *interval endpoints*) are associated to every vertex v of G, and there is an edge $uv \in E$ if and only if $l_u < l_v < r_u$ or $l_u < r_v < r_u$. Without loss of generality, one can assume that all the $2n$ interval endpoints are distinct and are indexed from 1 to $2n$.

Several alternative characterizations of interval graphs have been proposed so far in the literature.[20] Polynomial-time algorithms that recognize interval graphs and compute their interval representations are known.[22,23] Polynomial-time algorithms are also known for the classical vertex coloring problem on interval graphs.[24] Since it is known that a power of an interval graph is also an interval graph, the $L(1,\ldots,1)$ coloring problem of an interval graph G can be solved in polynomial time[7] by coloring the interval graph G^t.

The following lemma shows how to locate the strongly simplicial vertex of an interval graph, which will be used to find an approximate $L(\delta_1,\delta_2,\ldots,\delta_t)$ coloring.

Lemma 10.8[21] *In an interval graph, the vertex with maximum left endpoint is strongly simplicial.*

Lemma 10.8 suggests one scan the vertices of an interval graph by increasing left endpoints since, in this way, the t-simplicial vertex v of the induced subgraph $G[\{1,\ldots,v\}]$ is processed at each step, for $1 \le v \le n$.

10.5.1 Approximate $L(\delta_1,\ldots,\delta_t)$ coloring of Interval Graphs

Consider the interval representation of G, and assume that the intervals (vertices) are indexed by increasing left endpoints, namely $l_1 < l_2 < \cdots < l_n$. For each endpoint k, an interval v is called *open* if $l_v \le k < r_v$ and *deepest* if it is open and its right endpoint is maximum.

The algorithm to be presented maintains a family of $t+1$ sets of colors, called *palettes*, denoted by $P_0, P_1, \ldots, P_t$. The palette P_0 is initialized to the set of colors $\{0,1,\ldots,U\}$, where $U = \lambda^*_{G,t} + 2(\delta_t - 1)\lambda^*_{G,t} + \sum_{j=1}^{t-1} 2(\delta_j - \delta_{j+1})\lambda^*_{G,j}$. As shown by Lemma 10.4, such a color set is sufficiently large to obtain a legal $L(\delta_1,\ldots,\delta_t)$ coloring for G. The palette P_0 contains the readily usable colors. A color can leave P_0 because it is assigned to an interval. In such a case, the color is inserted into P_t and it will go downward through all the previous palettes before being reusable.

Algorithm Interval-Coloring $(G = (V, E), t, \delta_1, \ldots, \delta_t)$;

$U := \lambda^*_{G,t} + 2(\delta_t - 1)\lambda^*_{G,t} + \sum_{j=1}^{t-1} 2(\delta_j - \delta_{j+1})\lambda^*_{G,j}$;
$L_v := \emptyset$ for every vertex $v = 1, \ldots, n$;
$P_0 := \{0, 1, \ldots, U\}$ and $P_j := \emptyset$ for $j = 1, \ldots, t$;
TABOO$[\gamma] := 0$ for $\gamma = 0, \ldots, U$;
MAX $:= 0$;
$\delta_{t+1} := 0$;
for $k := 1$ to $2n$ do
  if $k = l_v$ for some v, then
    extract a color c from P_0;
    $f(v) := c$;
    for each color $\max\{0, c - \delta_1 + 1\} \leq \gamma \leq \min\{c + \delta_1 - 1, U\}$ do
      if $\gamma \in P_0$ then extract γ from P_0;
      TABOO$[\gamma]$:= TABOO$[\gamma]$+1;
    insert color c into both L_v and P_t;
    if $r_v >$ MAX then set MAX $:= r_v$ and DEEP $:= v$;
  otherwise, if $k = r_v$ for some v, then
    for each color c in L_v do
      let j be such that $c \in P_j$;
      extract c from P_j;
      for each color $\max\{0, c - \delta_{t-j+1} + 1\} \leq \gamma \leq \max\{0, c - \delta_{t-j+2}\}$
          or $\min\{c + \delta_{t-j+2}, U\} \leq \gamma \leq \min\{0, c + \delta_{t-j+1} - 1, U\}$ do
          if $\gamma \neq c$ then TABOO$[\gamma]$:= TABOO$[\gamma]$−1;
          if TABOO$[\gamma] = 0$ then insert γ into P_0;
      if $j > 1$ then
        insert c into both P_{j-1} and L_{DEEP}
      else
        TABOO$[c]$:= TABOO$[c]$−1;
        if TABOO$[c]$=0 then insert c into P_0;

Figure 10.1 The approximate algorithm for $L(\delta_1, \ldots, \delta_t)$ coloring of interval graphs.

Precisely, the palette P_t includes the colors used for the currently open intervals, while the generic palette P_i, with $1 \leq i \leq t-1$, contains the colors that could be reused as soon as all the next i consecutive deepest intervals will be ended. A color can leave P_0 without being assigned to any interval just because another used color forbids it. A counter is used to keep track of how many used colors currently forbid it.

Figure 10.1 illustrates the algorithm for $L(\delta_1, \ldots, \delta_t)$ coloring of interval graphs, called *Interval-Coloring*. The algorithm scans the $2n$ interval endpoints from left to right. Whenever a new interval v begins, that is a left endpoint l_v is encountered, v is colored by a color c extracted from the palette P_0 and, if needed, the deepest interval is updated. The used color c is put both in the palette P_t and in the set L_v of colors which depend on vertex v. Moreover, all the colors γ with $|\gamma - c| < \delta_1$ are forbidden by c, and thus their counters are incremented. Whenever an interval v ends, that is a right endpoint r_v is encountered, every color c belonging to L_v is deleted from its current palette, say P_j. Since the δ_{t-j+1}-separation constraint due to c does not hold anymore, all the colors γ with $\delta_{t-j+2} \leq |\gamma - c| < \delta_{t-j+1}$

are no more forbidden by c, and their counters are decremented. A color γ becomes available whenever its counter reaches 0, and in such a case it is reinserted in P_0. Whenever j is larger than 1, the color c, previously extracted from P_j, is moved to palette P_{j-1}, and inserted in the set L_{DEEP} of the colors which depend on the current deepest interval DEEP. If j is equal to 1, the color c becomes reusable, and thus it is inserted into P_0 provided that its counter becomes 0.

Lemma 10.9[20] *Consider the Interval-Coloring algorithm at the beginning of iteration k with $k = l_v$ for some interval v to be colored. Consider any color $c \in P_j$, and let w be the rightmost interval colored by c. Then v is at distance $t - j + 1$ from w, that is $d(w, v) = t - j + 1$.*

Lemma 10.10[20] *Consider the Interval-Coloring algorithm at the beginning of iteration k, and let c be any color.*

- *If $c \in P_0$, then c is readily usable, and it does not forbid any other color.*
- *If $c \in P_j$ with $j > 0$, then c cannot be used, and it forbids all the colors γ such that $c - \delta_{t-j+1} + 1 \leq \gamma \leq c + \delta_{t-j+1} - 1$.*
- *If $c \notin P_0 \cup \cdots \cup P_t$, then it is forbidden, but it does not forbid any other color.*

In practice, the above two lemmas guarantee that the Interval-Coloring algorithm finds a legal $L(\delta_1, \ldots, \delta_t)$ coloring, that is one verifying all the separation constraints. Instead, the next theorem provides a bound on the ratio U/L between the largest used color $U = \lambda^*_{G,t} + 2(\delta_t - 1)\lambda^*_{G,t} + \sum_{j=1}^{t-1} 2(\delta_j - \delta_{j+1})\lambda^*_{G,j}$, given by Lemma 10.4, and the lower bound $L = \max_{1 \leq j \leq t}\{\delta_j \lambda^*_{G,j}\}$, provided by Lemma 10.2.

Theorem 10.1[20] *Let $\delta_m \lambda^*_{G,m} = \max_{1 \leq j \leq t}\{\delta_j \lambda^*_{G,j}\}$, and recall that $\delta_{t+1} = 0$. The Interval-Coloring algorithm gives an α-approximation with $\alpha = \min\{2t, \frac{2\delta_{m+1} - 1}{\delta_t} + \frac{2(\delta_1 - \delta_{m+1})}{\delta_m}\}$.*

It is worth noting that the Interval-Coloring algorithm provides a 4-approximation for the $L(\delta_1, \delta_2)$ coloring problem, as one can easily check by setting $t = 2$ in the formula of α given by Theorem 10.1. However, a better 3-approximation has been found,[21] even for arbitrary t, when $\delta_1 \geq 1$ and $\delta_2 = \cdots = \delta_t = 1$. As regard to the time complexity, one can prove that the Interval-Coloring algorithm runs in $O(n(t + \delta_1))$ time. Such a result is based on the fact that U can be computed in $O(nt)$ time[7] and that, between two consecutive assignments of the same color c to two intervals, $O(t)$ time is spent for at most $t + 1$ insertions and extractions of c through the palettes

and $O(\delta_1)$ time is paid for updating the TABOO counters. All the details can be found in the literature.[20]

10.5.2 Approximate $L(\delta_1,\delta_2)$ coloring of Unit Interval Graphs

Consider now the $L(\delta_1, \delta_2)$ coloring problem on the class of *unit interval graphs*. This is a subclass of the interval graphs for which all the intervals have the same length, or equivalently, for which no interval is properly contained within another. Recalling that vertices are assumed to be indexed by increasing left endpoints, the main property of unit interval graphs is that whenever $v < u$ and $vu \in E$, then the vertex set $\{v, v+1, \ldots, u-1, u\}$ forms a clique and $u \leq v + \lambda^*_{G,1}$ (as a consequence, the maximum vertex w at distance 2 from v verifies $w \leq v + 2\lambda^*_{G,1}$). Assume that the unit interval graph to be colored is not a path since otherwise the optimal $L(\delta_1, \delta_2)$ coloring algorithm for paths can be applied.[11]

A linear time algorithm has been given,[20] which colors each vertex v in $O(1)$ time as follows. If $\delta_1 > 2\delta_2$, then assign to vertex v the color

$$f(v) = \begin{cases} \delta_1(\lambda^*_{G,1} - p) & \text{if } 0 \leq p \leq \lambda^*_{G,1} \\ \delta_1(\lambda^*_{G,1} - p) + \delta_2 & \text{if } \lambda^*_{G,1} + 1 \leq p \leq 2\lambda^*_{G,1} + 1 \end{cases}$$

where $p = (v-1) \bmod (2\lambda^*_{G,1} + 2)$. Otherwise, namely when $\delta_1 \leq 2\delta_2$, then color v as

$$f(v) = (2\delta_2(v-1)) \bmod (2\delta_2\lambda^*_{G,1} + 3\delta_2).$$

The algorithm colors the vertices in a cyclic way. When $\delta_1 > 2\delta_2$, the vertices are colored by repeating the following sequence of length $2\lambda^*_{G,1} + 2$:

$$0, \delta_1, 2\delta_1, \ldots, \lambda^*_{G,1}\delta_1, \delta_2, \delta_1 + \delta_2, \delta_1 + 2\delta_2, \ldots, \lambda^*_{G,1}\delta_1 + \delta_2.$$

Instead, when $\delta_1 \leq 2\delta_2$, the vertices are colored by repeating the sequence of length $2\lambda^*_{G,1} + 3$:

$$0, 2\delta_2, 4\delta_2, \ldots, 2(\lambda^*_{G,1} + 1)\delta_2, \delta_2, 3\delta_2, 5\delta_2, \ldots, 2\lambda^*_{G,1}\delta_2 + \delta_2.$$

The algorithm runs in $O(n)$ time and provides a 3-approximation.[21] In particular, it uses either at most δ_2 additional colors with respect to the optimum, when $\delta_1 > 2\delta_2$, or at most $2\delta_2$ additional colors when $\delta_1 \leq 2\delta_2$. When $\delta_1 = 2$ and $\delta_2 = 1$, the algorithm uses at most 2 extra colors with respect to the optimum; as found by Sakai.[15]

10.6 Trees

An undirected graph $T = (V, E)$ is a *free tree* when it is connected, and it has exactly $|V| - 1$ edges. Given a vertex v of a free tree T, $Adj(v)$ denotes the set of vertices adjacent to v. Given also an integer t, $N_t(v)$ denotes the set of vertices at distance at most t from v. Clearly, $Adj(v) = N_1(v) - \{v\}$. A *rooted tree* is a free tree in which a vertex r is identified as a *root*, and all the other vertices are ordered by levels, where the level $\ell(v)$ of a vertex v is equal to the distance $d(r, v)$. Thus, all the vertices adjacent to v are partitioned into its *father*, denoted by $father(v)$, which is at level $\ell(v) - 1$, and into its children, which are at level $\ell(v) + 1$. The *height* h of a tree T is the maximum level of its vertices. For each vertex v of T, let $anc_i(v)$ denote the ancestor of v at distance i from v (which clearly is at level $\ell(v) - i$). Of course, $anc_1(v) = father(v)$, and $anc_0(v) = v$. Moreover, $lca(u, v)$ denotes the *lowest common ancestor* of u and v, that is the vertex with maximum level among all the common ancestors of both u and v. Finally, given a vertex v of T, T_v denotes the induced subtree rooted at v consisting of all the vertices having v as an ancestor.

To derive an approximate $L(\delta_1, \ldots, \delta_t)$ coloring of a rooted tree, the following lemma is useful since it shows how to locate a strongly simplicial vertex:

Lemma 10.11[21] *In a rooted tree of height h, any vertex at level h is strongly simplicial.*

Lemma 10.11 suggests visiting the tree in breadth-first-search order, namely scanning the vertices by increasing levels. Hereafter, it is assumed that the vertices are numbered according to the breadth-first-search order, obtained by starting the visit from the root. Precisely, the vertices are numbered level by level, and those at the same level from left to right. In this way, when a vertex v is considered, v is a t-simplicial vertex of the subtree $T[\{1, 2, \ldots, v\}]$ induced by the first v vertices of T, for $1 \leq v \leq n$.

10.6.1 Approximate $L(\delta_1, \ldots, \delta_t)$ coloring of Trees

Consider a double implementation of T, where T is viewed both as a free tree and as the rooted tree T_1. Specifically, $Adj(v)$, $father(v)$, and $\ell(v)$ are maintained for each vertex v. As before, the palette P_0 of readily usable colors is maintained, which is initialized to the set $\{0, 1, \ldots, U\}$, where U is the upper bound given by Lemma 10.4. Again, a counter TABOO[c] keeps track of how many used colors forbid color c.

The *Tree-Coloring* algorithm, illustrated in Figure 10.2, uses three procedures: *Ancestor*, *Up-Neighborhood-BFS*, and *Clear-BFS*, depicted in Figures 10.3, 10.4, and 10.5, respectively. Tree-Coloring first performs a

Algorithm Tree-Coloring $(T = (V, E), t, \delta_1, \ldots, \delta_t)$;

$U := \lambda^*_{G,t} + 2(\delta_t - 1)\lambda^*_{G,t} + \sum_{j=1}^{t-1} 2(\delta_j - \delta_{j+1})\lambda^*_{G,j}$;
$P_0 := \{0, 1, \ldots, U\}$;
TABOO$[\gamma] := 0$ for $\gamma = 0, \ldots, U$;
$u := 1$; $\delta_0 := 0$;
for $v := 1$ to n do
  $x :=$ Ancestor(u, v);
  if new then Clear-BFS(u);
  Up-Neighborhood-BFS(v, x);
  extract a color c from P_0;
  $f(v) := c$;
  TABOO$[c]$:= TABOO$[c] + 1$;
  $u := v$;

Figure 10.2 The approximate Tree-Coloring algorithm for $L(\delta_1, \ldots, \delta_t)$ coloring of trees.

Function Ancestor (u, v);

$w := u$; $y := u$; $up := \min\{t, \ell(u)\}$;
if $\ell(v) = \ell(u) + 1$
  then $i := 1$; $x := father(v)$
  else $i := 0$; $x := v$;
while $x \neq w$ and $i < up$ do
  $y := w$; $w := father(y)$;
  $x := father(x)$; $i := i + 1$;
if $x \neq w$ or $\ell(v) = \ell(u) + 1$
  then $new :=$ true
  else $new :=$ false;
return(x)

Figure 10.3 The Ancestor procedure to find the vertex *x* with maximum level between *lca(u,v)* and $anc_t(v)$. If *x* = *lca(u,v)* then *y* is the child of *x* on the path between *u* and *x*.

preprocessing to compute the upper bound U, which depends on $\lambda_{T,j}$, for each i, $1 \leq j \leq t$.

The algorithm scans the n vertices according to the BFS numbering. At each iteration, v represents the vertex to be colored next, while u is the last colored vertex. Hence, $u = v - 1$, for $2 \leq v \leq n$. To color v, one needs to determine the set of colors already used and forbidden in the neighborhood $N'_t(v) = N_t(v) \cap T[\{1, \ldots, v\}]$. Such a set of colors depends only on the distance $d(z, v)$ for each vertex $z \in N'_t(v)$, and it is computed incrementally with respect to the neighborhood $N'_t(u)$ of the last colored vertex u.

The behavior of the Tree-Coloring algorithm depends on whether $N'_t(v)$ and $N'_t(u)$ do intersect or not, and on whether u and v are at the same level or not. When $\ell(v) = \ell(u) + 1$ or when $\ell(v) = \ell(u)$ and $N'_t(v) \cap N'_t(u) = \emptyset$,

Procedure Up-Neighborhood-BFS (v, x);

```
dist(v) := 0; M := ∅;
Enqueue(Q, v);
while Q ≠ ∅ do
  w := Dequeue(Q);
  if w = x and not new then S := {y} else S := Adj(w);
(1) for each z ∈ S do
    if ℓ(x) ≤ ℓ(z) ≤ ℓ(v) and z < v and z is not marked then
      if w = v or 0 < dist(w) < t then
(2)     for each f(z) − δ_dist(z) + 1 ≤ γ ≤ f(z) + δ_dist(z) − 1 do
          if γ ≠ f(z) then
            TABOO[γ] := TABOO[γ] − 1;
            if TABOO[γ] = 0 then insert γ into P_0;
        dist(z) := dist(w) + 1; mark z;
(3)     for each f(z) − δ_dist(z) + 1 ≤ γ ≤ f(z) + δ_dist(z) − 1 do
          if γ ≠ f(z) then
            if TABOO[γ] = 0 then extract γ from P_0;
            TABOO[γ] := TABOO[γ] + 1;
      if not new and w ≠ v and (dist(w) = 0 or dist(w) = t) and dist(z) > 0 then
(4)     for each f(z) − δ_dist(z) + 1 ≤ γ ≤ f(z) + δ_dist(z) − 1 do
          TABOO[γ] := TABOO[γ] − 1;
          if TABOO[γ] = 0 then insert γ into P_0;
        dist(z) := 0; mark z;
    Enqueue(Q, z);
  insert w into M;
for each z ∈ M do unmark z;
```

Figure 10.4 The modified BFS procedure to compute distances and forbid colors for vertices in $N'_t(v)$ and clear distances and colors for vertices in $N'_t(u) - N'_t(v)$.

Procedure Clear-BFS (u);

```
Enqueue(Q, u);
  while Q ≠ ∅ do
    w := Dequeue(Q);
    for each z ∈ Adj(w) do
      if dist(z) > 0 then
        for each f(z) − δ_dist(z) + 1 ≤ γ ≤ f(z) + δ_dist(z) − 1 do
          TABOO[γ] := TABOO[γ] − 1;
          if TABOO[γ] = 0 then insert γ into P_0;
        dist(z) := 0;
        Enqueue (Q, z);
```

Figure 10.5 The procedure to clear distances and colors for vertices in $N'_t(u)$ when $\ell(v) = \ell(u) + 1$ or $N'_t(v) \cap N'_t(u) = \emptyset$.

the variable *new* is set true by the Ancestor function, the counters of all the used and forbidden colors in the old neighborhood $N'_t(u)$ are decremented by the Clear-BFS procedure, while the distances and the forbidden colors in $N'_t(v)$ are computed from scratch by the Up-Neighborhood-BFS

procedure. Indeed, although the neighborhoods $N'_t(v)$ and $N'_t(u)$ may intersect when $\ell(v) = \ell(u) + 1$, each node z belonging to the intersection has $d(z, v) \neq d(z, u)$, and thus its distance and its forbidden colors have to be recomputed. In particular, the empty intersection between $N'_t(v)$ and $N'_t(u)$ is recognized by the Ancestor function when $x = anc_t(v)$ is deeper than $lca(u, v)$, the least common ancestor between u and v. When $\ell(v) = \ell(u)$ and $N'_t(v) \cap N'_t(u) \neq \emptyset$, $x = lca(u, v)$ is deeper than $anc_t(v)$, and x belongs to the shortest path $sp(u, v)$ between u and v. Consider the vertices y and y' which are the children of x on the paths $sp(u, x)$ and $sp(v, x)$, respectively. In this case, the Ancestor function sets *new* to false and returns $x = lca(u, v)$ along with the vertex y. Indeed, y and y' are the roots of the subtrees $T_y \cap N'_t(v)$ and $T_{y'} \cap N'_t(u)$ containing each vertex $z \in N'_t(v) \cap N'_t(u)$ such that $d(z, u) \neq d(z, v)$. The Up-Neighborhood-BFS procedure updates only the colors already used and forbidden by vertices in $T_y \cap N'_t(v)$ and in $T_{y'} \cap N'_t(u)$, leaving unchanged those colors used and forbidden by vertices in $(N'_t(v) \cap N'_t(u)) - (T_y \cup T_{y'})$ because for each vertex z in such a subset $d(z, v) = d(z, u)$ holds. Moreover, the procedure introduces new forbidden colors due to the vertices in $N'_t(v) - N'_t(u)$, and finally frees the colors no longer used or forbidden by vertices in $N'_t(u) - N'_t(v)$.

Precisely, the Up-Neighborhood-BFS procedure is invoked with the aim of computing the distances from v of each vertex z in $N'_t(v)$, and accordingly changing the TABOO counters. This is done in Loop (1) during a Breadth-First-Search starting from vertex v in which the label $dist(z)$ is set to $d(z, v)$, and each separation constraint is updated by first decrementing in Loop (2) the counters of the colors γ with $0 < |f(z) - \gamma| < \delta_{d(z,u)}$, and then incrementing in Loop (3) those of the colors with $0 < |f(z) - \gamma| < \delta_{d(z,v)}$. Summarizing, when *new* is true, the procedure computes from scratch all the distances and forbidden colors for all vertices in $N'_t(v)$. In contrast, when *new* is false, distances and forbidden colors are computed from scratch only for vertices in $N'_t(v) - N'_t(u)$, they are updated only for a subset of vertices in $N'_t(v) \cap N'_t(u)$, and they are cleared in Loop (4) for those in $N'_t(u) - N'_t(v)$.

Lemma 10.12[20] *Let the Tree-Coloring algorithm be at iteration v just before coloring vertex v, and consider any vertex z in $T[\{1, \ldots, v\}]$. Then*

$$dist(z) = \begin{cases} d(z, v) & \text{if } z \in N'_t(v) \\ 0 & otherwise \end{cases}$$

Lemma 10.13[20] *Let the Tree-Coloring algorithm be at iteration v just before coloring vertex v, and consider any color c.*

- *If c is assigned to a vertex $z \in N'_t(v)$, then $c \notin P_0$ and it forbids only the colors γ such that $c - \delta_{d(z,v)} + 1 \leq \gamma \leq c + \delta_{d(z,v)} - 1$.*
- *If c is not assigned to any vertex in $N'_t(v)$ and $c \notin P_0$, then c is forbidden by at least a color assigned to a vertex in $N'_t(v)$, but c does not forbid any color.*
- *If $c \in P_0$, then c is readily usable and it does not forbid any color.*

In practice, the above lemma guarantees that a legal $L(\delta_1, \delta_2, \ldots, \delta_t)$-coloring is found, namely, a color c is assigned to a vertex v only when it satisfies all the separation constraints due to colors assigned to vertices at distance at most t from v. In particular, any vertex already colored c is at distance greater than t from v. As regard to the approximation ratio, the Tree-Coloring algorithm provides exactly the same α-approximation as the Interval-Coloring algorithm (see Theorem 10.1). In particular, the same 4- and 3-approximations hold for $L(\delta_1, \delta_2)$- and $L(\delta_1, 1, \ldots, 1)$-colorings, respectively. In addition, when $t = 2$ and T is a binary tree then a 10/3-approximation is achieved for the $L(\delta_1, \delta_2)$ coloring problem.[20]

The Tree-Coloring algorithm runs in $O(nt^2\delta_1)$ time. This result is based on the fact that computing U takes $O(nt^2)$ time,[7] while all the updates of $dist(x)$ for a given vertex x require $O(t^2\delta_1)$ time. Indeed, while coloring any vertex v at a given level ℓ, $dist(x)$ can assume $O(t)$ different values, one for each possible $lca(x, v) = anc_k(x)$ with $0 \leq k \leq \lfloor \frac{t}{2} \rfloor$. Moreover, all the vertices at level ℓ for which $dist(x)$ has a given value are split into at most two sequences of consecutive vertices and thus $O(\delta_1)$ time is paid only at the beginning of each subsequence. Since x can be involved in coloring vertices in at most levels $\ell(x), \ell(x) + 1, \ldots, \ell(x) + t$ and there are n vertices, the overall time taken by the algorithm is $O(nt^2\delta_1)$.

10.7 Conclusion

This chapter has considered the channel assignment problem for various separation vectors and several particular network topologies—rings, grids, trees, and interval graphs. Specifically, $O(n)$ time algorithms have been presented for optimally solving the $L(2, 1)$-, $L(2, 1, 1)$-, and $L(\delta_1, 1, \ldots, 1)$ coloring problems on rings. Moreover, optimal solutions for the $L(\delta_1, \delta_2)$ coloring problem on both bidimensional and cellular grids, and for the $L(\delta_1, 1, \ldots, 1)$ coloring problem on bidimensional grids have been described, which are based on arithmetic progressions and take $O(rc)$ time to color the entire grid of r rows and c columns. Then, based on the notion of strongly-simplicial vertices, $O(n(t + \delta_1))$ and $O(nt^2\delta_1)$ time algorithms have been proposed to find α-approximate $L(\delta_1, \ldots, \delta_t)$ colorings on trees and interval graphs,

respectively, where α is a constant depending on t and $\delta_1, \ldots, \delta_t$. In particular, when $t = 2$, such algorithms provide 4-approximate $L(\delta_1, \delta_2)$ colorings, while they yield 3-approximate $L(\delta_1, 1, \ldots, 1)$ colorings when δ_1 is the only separation greater than 1. For $t = 2$ and binary trees, a 10/3-approximation is achieved. Moreover, an $O(n)$ time 3-approximate algorithm giving an $L(\delta_1, \delta_2)$ coloring of unit interval graphs has also been presented.

The main results reviewed in this chapter are summarized in Table 10.1. Several questions remain open. For instance, at the best of our knowledge, no results have been presented so far on the $L(\delta_1, \ldots, \delta_t)$ coloring problem for rings and grids. In addition, it is still unknown whether finding optimal $L(\delta_1, \delta_2)$ colorings of unit interval graphs or trees is NP-hard or not. Finally, as a matter of further research, one could devise better approximate algorithms for finding $L(\delta_1, \ldots, \delta_t)$ colorings of interval graphs and trees.

Table 10.1 Main Results on the Channel Assignment Problem Reviewed in the Present Chapter, where $\zeta = t + \left\lceil \frac{n \bmod (t+1)}{\lfloor \frac{n}{t+1} \rfloor} \right\rceil$ and $\alpha = \min\left\{2t, \frac{2\delta_{m+1} - 1}{\delta_t} + \frac{2(\delta_1 - \delta_{m+1})}{\delta_m}\right\}$

Networks	$L(\delta_1,\delta_2)$		$L(\delta_1,1,\ldots,1)$		$L(\delta_1,\delta_2,\ldots,\delta_t)$	
Rings	Optimum	[10]	Optimum if $\delta_1 \leq \lfloor \frac{\zeta}{2} \rfloor$	[8]	Open	
Bidimensional grids	Optimum	[11]	Optimum if $\delta_1 \leq \lfloor \frac{t}{2} \rfloor$	[8]	Open	
Cellular grids	Optimum	[11]	Optimum	[9]	Open	
Trees	4-approximation	[20]	3-approximation	[20]	α-approximation	[20]
Interval graphs	4-approximation	[20]	3-approximation	[20]	α-approximation	[20]

References

1. R. Battiti, A.A. Bertossi, and M.A. Bonuccelli, "Assigning Codes in Wireless Networks: Bounds and Scaling Properties", *Wireless Networks*, Vol. 5, 1999, pp. 195–209.
2. A.A. Bertossi and M.C. Pinotti, "Mappings for Conflict-Free Access of Paths in Bidimensional Arrays, Circular Lists, and Complete Trees", *Journal of Parallel and Distributed Computing*, Vol. 62, 2002, pp. 1314–1333.
3. I. Chlamtac and S.S. Pinter, "Distributed Nodes Organizations Algorithm for Channel Access in a Multihop Dynamic Radio Network", *IEEE Transactions on Computers*, Vol. 36, 1987, pp. 728–737.
4. S.T. McCormick, "Optimal Approximation of Sparse Hessians and its Equivalence to a Graph Coloring Problem", *Mathematical Programming,* Vol. 26, 1983, pp. 153–171.

5. A. Sen, T. Roxborough, and S. Medidi, "Upper and Lower Bounds of a Class of Channel Assignment Problems in Cellular Networks", *Proceedings of IEEE INFOCOM'98*, 1998.
6. A.A. Bertossi, M.C. Pinotti, R. Rizzi, and A.M. Shende "Channel Assignment for Interference Avoidance in Honeycomb Wireless Networks", *Journal of Parallel and Distributed Computing*, Vol. 64, No. 12, 2004, pp. 1329–1344.
7. G. Agnarsson, R. Greenlaw, and M.M. Halldorson, "On Powers of Chordal Graphs and Their Colorings", *Congressus Numerantium*, Vol. 144, 2000, pp. 41–65.
8. A.A. Bertossi, M.C. Pinotti and R.B. Tan, "Channel Assignment with Separation for Interference Avoidance in Wireless Networks", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, 2003, pp. 222–235.
9. M.V.S. Shashanka, A. Pati, and A.M. Shende, "A Characterisation of Optimal Channel Assignments for Cellular and Square Grid Wireless Networks", *Mobile Networks and Applications*, Vol. 10, 2005, pp. 89–98.
10. J.P. Georges and D.W. Mauro, "Generalized Vertex Labelling with a Condition at Distance Two", *Congressus Numerantium*, Vol. 109, 1995, pp. 141–159.
11. J. Van den Heuvel, R.A. Leese, and M.A. Shepherd, "Graph Labelling and Radio Channel Assignment", *Journal of Graph Theory*, Vol. 29, 1998, pp. 263–283.
12. H.L. Bodlaender, T. Kloks, R.B. Tan, and J. van Leeuwen, "Approximations for λ-Coloring of Graphs", *The Computer Journal*, Vol. 47, 2004, pp. 193–204.
13. G.J. Chang and D. Kuo, "The $L(2,1)$-Labeling Problem on Graphs", *SIAM Journal on Discrete Mathematics*, Vol. 9, 1996, pp. 309–316.
14. J.R. Griggs and R.K. Yeh, "Labelling Graphs with a Condition at Distance 2", *SIAM Journal on Discrete Mathematics*, Vol. 5, 1992, pp. 586–595.
15. D. Sakai, "Labeling Chordal Graphs: Distance Two Condition", *SIAM Journal on Discrete Mathematics*, Vol. 7, 1994, pp. 133–140.
16. T. Calamoneri, "The $L(h,k)$-Labelling Problem: An Annotated Bibliography", *The Computer Journal*, Vol. 49, 1996, pp. 585–608.
17. T. Makansi, "Transmitted Oriented Code Assignment for Multihop Packet Radio", *IEEE Transactions on Communications*, Vol. 35, 1987, pp. 1379–1382.
18. A.A. Bertossi and M.A. Bonuccelli, "Code Assignment for Hidden Terminal Interference Avoidance in Multihop Packet Radio Networks", *IEEE/ACM Transactions on Networking*, Vol. 3, 1995, pp. 441–449.
19. K.I. Aardal, S.P.M. van Hoesel, A.M.C.A. Koster, C. Mannino, and A. Sassano, "Models and Solution Techniques for Frequency Assignment Problems", 4OR, Vol. 1, 2003, pp. 261–317.
20. A.A. Bertossi and M.C. Pinotti, "Approximate $L(\delta_1, \delta_2, \ldots, \delta_t)$ coloring of Trees and Interval Graphs", Networks, to appear.
21. A.A. Bertossi, M.C. Pinotti, and R. Rizzi, "Channel Assignment with Separation on Trees and Interval Graphs", *3rd International Workshop on Wireless, Mobile and Ad Hoc Networks (WMAN)*, IEEE IPDPS, Nice, April 2003.

22. K.S. Booth and G.S. Lueker, "Linear Algorithms to Recognize Interval Graphs and Test for the Consecutive Ones Property", *Seventh Annual ACM Symposium on Theory of Computing*, Albuquerque, New Mexico, 1975, pp. 255–265.
23. D.G. Corneil, S. Olariu, and L. Stewart, "The Ultimate Interval Graph Recognition Algorithm?", *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, 1998, pp. 175–180.
24. A. Brandstådt, V.B. Le, and J.P. Spinrad, *Graph Classes: A Survey*, SIAM, Philadelphia, PA, 1999.