Channel assignment and graph multicoloring — Source link

Lata Narayanan

Institutions: Concordia University

Published on: 01 Jan 2002

Topics: Graph (abstract data type) and Co-channel interference

1 Channel assignment and graph multicoloring

LATA NARAYANAN

Department of Computer Science

Concordia University

Montreal, Quebec, Canada

Cellular data and communication networks are usually modeled as graphs with each node representing a base station in a cell in the network, and edges representing geographical adjacency of cells. The problem of channel assignment in such networks can be seen as a graph multicoloring problem. We survey the models, algorithms, and lower bounds for this problem.

—

1.1 INTRODUCTION

In a cellular network, there are ongoing requests for communication links from mobiles in each cell to the base stations responsible for the cell. In FDMA or TDMA networks, the available spectrum is divided into narrow frequency channels, and each communication request is served by the assignment of a frequency channel. Spectrum is a scarce resource, and careful assignment of channels to calls is critical to being able to maximize the number of users in the network. Cellular networks employ a low power transmitter in every cell, which makes it possible to reuse the same frequency channel in different cells. Frequency reuse, however, is limited by two kinds of radio interference. *Co-channel interference* is caused by two simultaneous transmissions on the same channel. To avoid this, once a channel is assigned to a certain call, it should not be reused by another call in an area where it may cause significant interference. *Adjacent channel interference* is the result of signal energy from an adjacent channel spilling over into the current channel.

In this chapter, we model cellular data and communication networks as graphs with each node representing a base station in a cell in the network, and edges representing geographical adjacency of cells. Associated with each node v in the graph at any time is a set $C(v)$ which is the set of calls or active mobiles in the cell served by v. The size of the set $C(v)$, is denoted $w(v)$, and called the weight of the node v. Co-channel interference constraints are modeled in terms of reuse distance; it is assumed that the same channel can be assigned to two different nodes in the graph if and only if their graph distance is at least r. We do not deal with adjacent channel interference in this chapter; see [20] (in this book) for models and solutions for this problem. For our purposes, the objective of an algorithm for the *channel assignment problem* is, at time instant t, to assign $w_t(v)$ channels to each node v in the network, where $w_t(v)$ is the weight of the node v at time t, in such a way that co-channel interference constraints are respected, and the total number of channels used over all nodes in the network is minimized. The problem can thus be seen as a graph multicoloring problem.

The channel assignment problem has been studied extensively in the last two decades. A look at an online research index turns up hundreds of papers on the subject. The communities involved include radio and electrical engineers, operations researchers, graph theorists, and computer scientists of all stripes. In this chapter, we limit ourselves to papers taking a graph-theoretic approach to the problem, and furthermore, to results that prove some bounds on the performance of the proposed algorithms. A comprehensive survey of channel assignment strategies proposed in the engineering literature is given in [26] and techniques used in operations research can be found in [21]. The relationship to graph coloring and multicoloring was set out in detail in [13]. We use the competitive analysis framework to analyze algorithms. Basic definitions are given in Section 1.2; see [3] for more details on competitive analysis and [2] on graph theory. A large part of our focus will be on

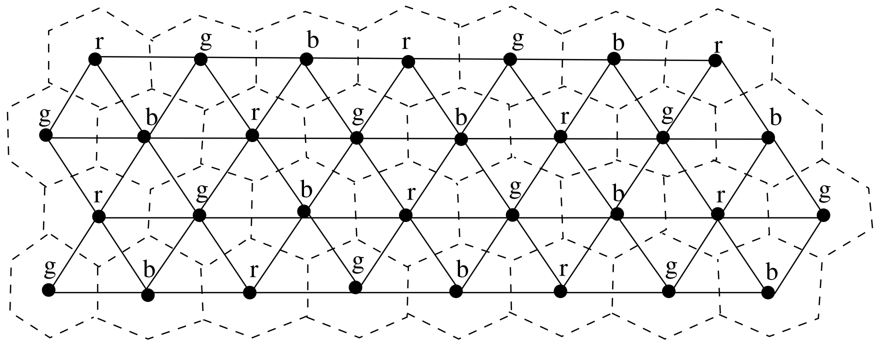

Fig. 1.1 A hexagon graph with a 3-coloring.

the so-called hexagon graphs (see Figure 1.1). It is well-known these are a very imperfect approximation of real-life cells; nevertheless they provide a convenient idealization that continues to be useful in practice. We will look at different values of the reuse distance parameter; this amounts to multicoloring powers of hexagon graphs. We will also look at the results on unit disk graphs. These are the intersection graphs of equal-sized disks in the plane, where a disk represents a transceiver and its transmission area. Finally we will mention a few results on graphs where adjacent cells have some overlapping areas.

Since the processing of calls and assignment of frequencies is an ongoing process, and present decisions can influence future ones, the channel assignment problem is best modeled in an online fashion. The graph to be multicolored changes over time. These changes can be modeled as an ordered sequence of graph-call vector pairs $\{(G, C_t) : t \geq 0\}$ where C_t is the set of new and ongoing calls to be serviced at time t. Clearly $C_t \cap C_{t+1}$ may not be empty. This brings us to an important issue, which concerns whether or not a node, when allocating channels for the next time step, can change the channels it has already assigned to its ongoing local calls on previous steps. An algorithm said to be *non-recoloring* if once having assigned a channel in response to a particular new call it never changes that assignment (*i.e.*, recolors the call).

The available technology, however, allows for a limited rearrangement of frequency channels. It is interesting therefore, to consider also algorithms that allow recoloring of calls. In this case, the actual set of calls at any given time step is no longer relevant; just the number of calls at any step suffices to specify the problem. Thus, the problem can now be modeled as multicoloring a sequence of weighted graphs given by $\{(G, w_t) : t \geq 0\}$. If arbitrary recoloring is allowed, it is not difficult to see that the competitive ratio of any algorithm for this online problem is no better than the performance ratio of an algorithm for the *static* version of the problem. Essentially, at every time step, run the algorithm for the static version of the problem, with no concern for how many calls get recolored. Another motivation for studying the static problem is that the weight of a node can be considered to represent the expected traffic in the corresponding cell, and an offline solution then gives a non-uniform precomputed fixed assignment to be used in the online setting.

It is intuitively clear that recoloring is a very powerful tool, and algorithms that are allowed to recolor are more powerful than those that are not. Upper and lower bounds that confirm this are reported in the later sections. An interesting challenge would be to develop algorithms that do allow recoloring but only a limited amount. Alternatively, there could be a cost associated with recoloring, and the algorithm would have to minimize this cost along with other objectives, or keep the cost below a specified allowable limit. As far as the author is aware, there has not been significant work in this area.

Another important issue concerns whether the algorithm uses centralized or distributed control. Some algorithms are inherently centralized: they require knowledge of all nodes and their current assignments before being able to make an assignment. The strategy in [35] for reuse distance 2, and the algorithm with competitive ratio 2 for reuse distance 3 given in [9] are centralized strategies. A commonly used strategy precomputes channels to be assigned to different cells; new calls are assigned channels from these precomputed sets. This strategy is completely distributed and requires no online communication with neighbors. There are yet other algorithms which can be implemented in a distributed manner, but require a limited amount of communication with nearby base stations to ensure that no conflict occurs. To limit the amount of information about other nodes that needs to be collected to aid in decision-making, the concept of the *locality* of an algorithm was introduced in [19].

An algorithm is said to be k-local, if the only information available to a node apart from its own history, is the current weights of nodes that are within graph distance k from it. A certain amount of precomputed information independent of the weight sequence, such as a base coloring of the graph, is also allowable. This model makes sense for recoloring algorithms; indeed knowing the weight of a neighbor also gives some knowledge about the decisions the neighbor would make in the current time step. However, for non-recoloring algorithms, it would make more sense to allow a k-local algorithm to also collect information about the current color assignments of nodes in its k-locality. The distributed algorithms described in this chapter are for the most part, synchronous. They proceed in rounds, and a mechanism to synchronize the rounds is assumed. In [41], however, the algorithm is asynchronous, and is proved to be free of deadlock and starvation. No bounds are proved on performance, as in the number of channels used in the average case or worst case.

There are very few lower bounds known for this problem, for either recoloring or non-recoloring algorithms. Janssen and Kilakos [17] show that for general k-colorable graphs, the competitive ratio of any 1-local recoloring algorithm is at least $k/2$ and that of any non-recoloring algorithm is at least k. These bounds are tight. However, for hexagon graphs or their powers, tight lower bounds are not known. A straightforward lower bound is the clique bound [10]: clearly all nodes in any clique in the graph need disjoint sets of channels. Thus the maximum over all cliques in the graph over the total weight of the clique is a lower bound for the number of channels needed. Another lower bound is given by odd cycles [37]: an odd cycle with n nodes needs at least $\frac{2W}{n-1}$ colors where W is the sum of weights of all nodes in the cycle. Most of the known algorithms are deterministic, and the lower bounds are for deterministic algorithms as well. Randomized algorithms are given in [27, 48], and the simulation results are promising, but no performance bounds are proved.

A related problem to the channel assignment problem is that of *online call control*. Suppose the size of the available spectrum is fixed to be C. Then given a sequence of weighted graphs, the call control problem is to assign channels to the nodes so as to maximize the total number of calls admitted. In this case, some calls may actually be rejected, so the number of channels assigned to a node in a time step may be less than its weight. An algorithm for channel assignment can generally be converted to one for online call control. Moreover, the performance bound for channel assignment also translates to a condition for blocking in the case of online control. Suppose an algorithm always produces a channel assignment with at most kD channels when given a weighted graph, where D is the clique bound mentioned above. Then it is easy to see that given a spectrum with C channels, the algorithm will not block unless there is a weighted clique in the graph with total weight more than C/k. A few recent papers study the call control problem [4, 28, 39] in both static and online versions. The static version is very similar to the maximum independent set problem. For the online version, Caragiannis *et al.* [4] give a randomized algorithm with competitive ratio 2.934 in hexagon graphs, and also give a lower bound of 1.857 on the competitive ratio of any randomized algorithm for the problem on such graphs. It is assumed that a single frequency channel is available.

Another related problem is the problem of *list coloring* [8]. In this problem, every node of the graph has associated with it a list of colors which in our case is the list of available frequency channels. The problem is to find a proper coloring of nodes such that each node is colored with a color from its list. A number of sequential solutions are known [1, 22, 44]. The relationship to channel assignment was noticed in [11, 31], and a distributed protocol was given in [11].

The rest of the chapter is organized as follows. Section 1.2 defines the terms we use. Section 1.3 outlines the basic types of algorithms proposed in the literature, and Section 1.4 summarizes the known lower bound results. Section 1.5 discusses the static version of the problem, and Section 1.6 the online version. We conclude with a discussion of open problems in Section 1.7.

1.2 PRELIMINARIES

1.2.1 Graph-theoretic preliminaries

Let $G = (V, E)$ denote an *interference* graph, where the node set V denotes cells or base stations that require call service, and the edge set E represents geographical proximity of cells and therefore the possibility of co-channel interference. A *weighted* *graph* is a pair (G, w) where G is an interference graph and w is a weight vector indexed by the nodes of G, and $w(v)$ represents the number of calls to be served at node v. A static snapshot of the network at a fixed instant of time is given by a weighted graph (G, w). The goal of an algorithm for the *static channel assignment* problem, at that instant in time, is to be able to allocate $w(v) \geq 0$ distinct channels to each node $v \in V$ such that no two adjacent nodes have channels in common. In graph-theoretic parlance, what is required is a *proper* multicoloring of G with distinct colors representing distinct channels.

Formally, a proper multicoloring of the weighted graph (G, w) where $G = (V, E)$ consists of a set of colors C and a function f that assigns to each $v \in V$ a subset $f(v)$ of C such that

- $\forall v,\ |f(v)| = w(v)$: each node gets $w(v)$ distinct colors, and
- $\forall (u, v) \in E,\ f(u) \cap f(v) = \phi$: two neighboring nodes get disjoint sets of colors.

Thus, a proper multicoloring is equivalent to a valid channel assignment and vice-versa. We use the terms *colors* and *channels* interchangeably in the sequel. Many of the algorithms use a base coloring of the underlying unweighted graph G; it should be clear from the context when the base color of a node is being referred to, rather than the channels it is assigned. It is convenient to treat the set of available channels to be a set of natural numbers. We further assume without loss of generality that any such set can be suitably reordered or partitioned. The *span* of a channel assignment is the cardinality of the set C.

For the *online channel assignment* problem, the set of calls to be served changes with time. Furthermore, calls cannot always be considered interchangeable, and therefore, the identities of individual calls may be relevant. We define a *call graph* to be a pair (G, C) where G is an interference graph and C is a call vector indexed by the nodes of the graph. $C(v)$ represents the set of ongoing and new calls requiring service at a node at a particular instant of time. A call graph has a one-to-one correspondence with a weighted graph, where $w(v) = |C(v)|$. We model the changes in the set of calls over time as an ordered sequence of call graphs $\{(G, C_t) : t \geq 0\}$, where

C_t represents the set of calls to be serviced at time t. At time instant t, an online algorithm must arrange to perform an assignment for the call graph (G, C_t) before moving on to the call graph (G, C_{t+1}) at the next time instant $t + 1$. It must perform this assignment with no knowledge of the later graphs in the sequence. As mentioned in the introduction, an algorithm for the online problem may be *recoloring* or *non-recoloring*. A recoloring algorithm can change the channels assigned to a call while the call is in progress, whereas in a non-recoloring algorithm, a call keeps the channel it is initially assigned for its entire duration.

If unlimited recoloring is allowed, the online problem becomes equivalent to the static problem, as an algorithm for the static problem can be used at every time step of the online algorithm. In this case, since the calls *can* be considered interchangeable, it is the *number* of calls at a node that is the relevant parameter. Thus it is enough to consider the sequence of weighted graphs $\{(G, w_t) : t \geq 0\}$ corresponding to the sequence of call graphs, and in fact, it suffices to consider each element of this sequence completely independently of the others. In practice, though, it is generally desirable to reassign channels to calls as little as possible.

We refer to finite induced subgraphs of the infinite triangular lattice as *hexagon graphs* (see Figure 1.1). A *unit disk graph* is the intersection graph of disks of equal diameter in the plane. They can also be described in terms of distance or proximity models, which consist of a value $d \geq 0$ and an embedding of nodes in the plane such that (u, v) is an edge if and only if the Euclidean distance between u and v in the specified embedding is at most *d*. For each fixed pair or real values $r, s > 0$ a graph G can be drawn in $\mathcal{R}^d$ in an (r, s)-civilized manner if its nodes can be mapped to points in $\mathcal{R}^d$ so that the length of each edge is at most r and the distance between any two points is at least s.

An *independent* (or, *stable*) set in G is a set $V' \subseteq V$ such that for any $u, v \in V'$, $(u, v) \notin E$. Note that a proper multicoloring of G is essentially a covering of G with stable sets; each stable set of nodes corresponds to a color and a node v appears in exactly $w(v)$ such sets. The *weighted chromatic number* of a weighted graph (G, w), denoted $\chi(G, w)$, is the smallest number m such that there exists a multicoloring of G of span m, i.e. $\chi(G, w)$ is the *optimal* number of colors required to properly multicolor G. Let the weight of a maximal clique in (G, w) be defined as the sum of the weights of the nodes belonging to the clique; note that when G is a hexagon graph, the only maximal cliques are isolated nodes, edges or triangles. The *weighted clique number*, denoted $\omega(G, w)$ is the maximum weight of any maximal clique in the graph. The weighted chromatic number and clique number of a call graph are defined in an analogous fashion. For integers $k \geq 1$, we define the k-locality of a node v to be the induced subgraph consisting of those nodes in G whose graph distance from v is less than or equal to k.

Given $G = (V, E)$, the graph $G^r = (V, E')$ is defined by $E' = E \cup E^2 \cup \ldots \cup E^{r-1}$. Thus any pair of nodes at distance $i < r$ in G is connected by an edge in G^r. The problem of channel assignment in a weighted graph (G, w) with reuse distance r is thus the same as multicoloring the graph G^r. The *unweighted clique number* of G^r is the maximum size of any clique in G^r, and is denoted $\omega(G^r)$. Similarly, the *chromatic number* of G^r, the minimum number of colors needed to color G^r, is denoted by $\chi(G^r)$. It is known that when G is a hexagon graph, $\chi(G^r) = \omega(G^r)$ (see Section 1.5.3), and an optimal coloring can be computed in polynomial time. We assume that such an optimal coloring of the graph G^r is available; thus every node in G^r is assigned a color from the set $\{1, 2, \ldots, \chi(G^r)\}$. If G is a hexagon graph, we say that G^2 is a *square graph.* Figure 1.1 shows a base coloring with red, blue, and green colors for a hexagon graph. It is easy to see that a square graph can be base colored with seven colors; Figure 1.8 shows such a coloring. We define $N_r(v)$ to be all neighbors of v in G^r, and $H_r(v)$ to be $w(v) + \Sigma_{u \in N_r(v)} w(u)$. Finally, given a assignment of channels to $N_r(v)$, $RC_r(v)$ is defined to be the number of channels assigned to more than one node in $N_r(v)$. Thus $RC_r(v)$ is a measure of the reuse of channels within $N_r(v)$.

1.2.2 Performance measures

An approximation algorithm for the static channel assignment problem is said to have performance ratio r if on any weighted graph (G, w), the algorithm uses at most $r\chi(G, w) + O(1)$ channels. We use a standard yardstick for measuring the efficacy of online algorithms: that of *competitive ratios* [25, 3]. Given an online algorithm

P that processes a sequence of N call graphs $(G, C_t),\ \ t = 0, \ldots, N$, let $\mathcal{S}(P_t)$ denote the span of the channel assignment (multicoloring) computed by P after step t, *i.e.* after graph (G, C_t) has been processed. Let $\mathcal{S}_N(P) = \max_t\{\mathcal{S}(P_t)\}$ and $\chi_N(G) = \max_t\{\chi(G, C_t)\}$. We say that P is a *c-competitive* algorithm if and only if there is a constant b independent of N such that for any input sequence,

$$\mathcal{S}_N(P) \quad \leq \quad c \cdot \chi_N(G) + b.$$

In other words, a c-competitive algorithm uses at most c times as many channels (colors) overall as the optimal offline algorithm would. We note that all of the algorithms discussed in this chapter in fact satisfy the stricter requirement

$$\mathcal{S}(P_t) \quad \leq \quad c \cdot \chi(G, C_t) + b$$

for all $t \geq 0$, *i.e.* they approximate the optimal span within a factor of c at *all* times while still processing the input sequence online. All of the lower bounds mentioned in this chapter hold for the above definition of c-competitive (and therefore imply lower bounds on algorithms satisfying the stricter requirement).

1.3 BASIC TYPES OF ALGORITHMS

In this section, we describe the basic types of algorithms developed for channel assignment seen as a graph multicoloring problem.

Fixed Assignment

Fixed assignment (FA) is a very simple, non-recoloring strategy for channel assignment. In this strategy, the nodes are partitioned into independent sets, and each such set is assigned a separate set of channels [30]. It is easy to see that this works very well when the traffic is described by a uniform distribution. However, it behaves quite poorly in the worst case. In particular, the algorithm is k-competitive where k is the chromatic number of the underlying graph. In [17], it is shown that this is the best possible for arbitrary k-colorable graphs.

Borrowing algorithms

To improve the performance of FA, one idea has been to assign nominal sets of channels as with FA, but allow borrowing of available channels [7, 18, 37, 35]. A simple method is to have a two phase algorithm. In the first phase, every node uses as many channels as it needs from its own nominal set of channels. In the second phase, a node borrows channels if necessary from its neighbors' sets of channels. Some mechanism to avoid conflicts caused by two neighbors trying to borrow the same channel from a mutual neighbor in the second phase is usually needed. One such mechanism might be to restrict the borrowing in some way. For example, red nodes can only borrow green channels, green nodes can only borrow blue channels, and blue nodes can only borrow red channels.

Hybrid Channel Assignment

Another variation of FA is to assign nominal sets of channels to nodes, but divide these sets into *reserved* and *borrowable* channels. The node may use all the channels from its own set, both reserved and borrowable ones, but may only use the borrowable channels from its neighbors, provided they are not being used by any of the neighbors. Many hybrid strategies have been studied in the literature [24, 47], but performance bounds are not generally given. Jordan and Schwabe [23] analyze a simple hybrid channel assignment strategy; the results for small values of reuse distance are not as good as the borrowing strategies.

Dynamic Channel Assignment

The main characteristic common to all dynamic channel assignment (DCA) schemes is that all channels are kept in a central pool and are assigned dynamically to radio cells as new calls arrive in the system. A channel is eligible for use in any cell, provided interference constraints are met. DCA strategies vary in the criterion used to select the channel to be assigned from the set of all eligible channels. They also vary in whether they are centralized or distributed, and the synchronization mechanism used in the latter case. A wide variety of selection criteria to choose the assigned channel can be found in the literature. For example, in [49], the algorithm tries to maximize the amount of reuse: roughly, the channel that has been used the most often at distance r but least often at distances $r+1$ and $r+2$ is used. A number of recently proposed DCA schemes are based on measurement of signal strength from various eligible channels, and aim to increase the reusability of channels (see for example, [43]). A commonly used strategy [5, 15, 41], is the purely *greedy* strategy of using the minimum numbered channel among those eligible. This is the only DCA strategy for which bounds on the competitive ratio are known.

1.4 LOWER BOUNDS

Graph multicoloring is well known to be NP-hard for arbitrary graphs. Mcdiarmid and Reed showed that it is also NP-hard to multicolor a hexagon graph with reuse distance 2 [35]; the proof can be adapted for other values of reuse distance as well. It is also known that multicoloring unit disk graphs is NP-hard; see for example, [12]. Janssen and Kilakos [17] showed that for a k-colorable graph, any non-recoloring algorithm has competitive ratio at least k, and any 1-local recoloring algorithm has competitive ratio at least $k/2$.

As mentioned earlier, the clique bound $\omega(G, w)$ is always a lower bound on the minimum number of channels needed to multicolor (G, w) [10]. Another lower bound is provided by odd cycles. Since the maximum size of an independent set in an n-node odd cycle is $(n-1)/2$, any color can be assigned to at most $(n-1)/2$ nodes. Therefore, the minimum number of channels needed for an odd cycle with n nodes and reuse distance 2 is at least $\frac{2W}{n-1}$ colors where W is the sum of weights of all nodes in the cycle. Thus, a 9-cycle with equal weight on all nodes requires at least $9\omega(G, w)/8$ channels, and in fact this number suffices. Since the smallest odd cycle that is a hexagon graph is of length 9, no algorithm for channel assignment with reuse distance 2 can guarantee using less than $9\omega(G, w)/8$ channels on all weighted hexagon graphs. For higher values of reuse distance, there are hexagon graphs G such that G^r is an induced 5-cycle (see Figure 1.2). Therefore for reuse

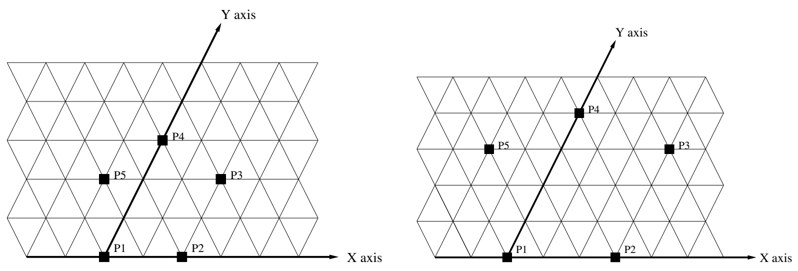

Fig. 1.2 (a) 5-cycle for reuse distance 3, and (b) 5-cycle for reuse distance 4.

distance 3 or higher, no algorithm for channel assignment can guarantee using less than $5\omega(G,w)/4$ channels on all weighted hexagon graphs.

In [19], a lower bound for the competitive ratio of any recoloring algorithm was shown. The constraint that recoloring can only occur in response to a change in demand within a node's immediate neighborhood is added. A recoloring algorithm is said to have *recoloring distance k* if a node recolors its calls during a time step only if a change of weight has occurred within its k-locality.

The following technical lemma aids the proof of the lower bound.

Lemma 1.4.1 *[19] Let P be a path of length ℓ, with weight n on each of its $\ell+1$ nodes. Then the minimal number of colors required to color P such that the end nodes have exactly α colors in common, is at least $2n+\frac{2\alpha}{\ell-1}$ when ℓ is odd, and at least $2n+\frac{2(n-\alpha)}{\ell}$ when ℓ is even.*

Let P, n and ℓ be as in the statement of the lemma. Let u and v be the end nodes of P, and let u' and v' be the neighbors of u and v, respectively (See Figure 1.3a). Then P' is constructed from P as follows. Node u is split into two connected nodes, u_1 and u_2, which are assigned weight α and $n-\alpha$, respectively. Similarly, node v is split into the connected nodes v_1 and v_2, with weight α and $n-\alpha$, respectively (Figure 1.3b). Obviously, coloring P such that u and v have exactly α colors in common is equivalent to coloring P' such that u_1 and v_1 receive the same colors, and u_2 and v_2 receive completely different colors. Next, the graph P'' is constructed from P' as follows. Nodes u_1 and v_1 are identified into one node uv_1, and nodes u_2 and v_2 are joined by an edge (Figure 1.3c). It is easy to see that any coloring of

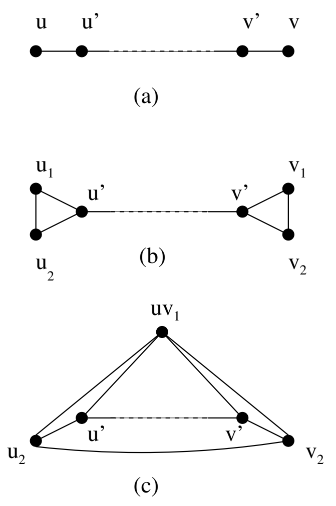

Fig. 1.3 Multicoloring a path where the endpoints have colors in common

P'' is equivalent to a coloring of P' in which u_1 and u_2 receive the same colors, and u_2 and v_2 receive different colors, which in turn is equivalent to a coloring of P as required by the lemma.

To determine a lower bound on the minimal number of colors needed to color P'', observe the following. If ℓ is odd, then the subgraph of P'' induced by all nodes except u_2 and v_2 is an odd cycle of length ℓ. The sum of the weights on this cycle is $(\ell-1)n+\alpha$, and the maximum size of an independent set in this cycle is $\frac{1}{2}(\ell-1)$. Hence the minimum number of colors needed to color P'' is at least

$$\frac{(\ell-1)n+\alpha}{\frac{1}{2}(\ell-1)} = 2n + \frac{2\alpha}{\ell-1}$$

If ℓ is even, then P'' consists of an odd cycle of length $\ell+1$, plus a node (uv_1), which is joined to four consecutive nodes of this cycle. So the maximal size of an independent set in P'' is $\frac{1}{2}\ell$. The sum of the weights on the nodes of P'' is $(\ell-1)n+\alpha+2(n-\alpha) = (\ell+1)n-\alpha$. Hence the minimum number of colors needed to color P'' is at least

$$\frac{(\ell+1)n-\alpha}{\frac{1}{2}\ell} = 2n + \frac{2(n-\alpha)}{\ell}$$

as claimed in the lemma.

The above lemma can be used to prove a lower bound on any online algorithm with recoloring distance k, using an adversary argument. The adversary chooses a hexagon graph, and at any step can *raise* or increase the weight at any node. The algorithm must respond by assigning colors corresponding to the increased weight.

The following argument is from [19]. Fix an online algorithm with recoloring distance $k \geq 0$. There is a strategy for the adversary which forces the algorithm to use at least $2n + \frac{n}{2(k+1)}$ colors, while the offline algorithm never needs more than $2n$ colors. The graph used by the adversary is shown in Fig 1.4. Let u and v be two nodes at distance 3 of each other along one of the axis of the grid. The adversary starts by raising the weight on u and v to n. If $k > 0$, the adversary continues to raise the weight to n on all nodes along two parallel axes of length $k - 1$ which make an angle of $\frac{\pi}{3}$ with the axis uv, and which start at u and v, respectively. The algorithm may color and recolor as desired.

Let u' and v' be the last nodes of the axis growing out of u and v, respectively, on which the weight has been raised. Nodes u' and v' have distance $k - 1$ from u and v, respectively. Next, the adversary raises the weight to n on two nodes, a and b, situated as follows. Node a is a neighbor of u', situated along the axis uu', at distance k from u. Node b is a neighbor of v' and lies at distance k from v, but is situated at an angle $\frac{\pi}{3}$ from the axis vv', and thus lies at distance 2 from node a (See Fig 1.4).

The next moves of the adversary will only involve nodes at distance greater than k from u and v, so the colors on u and v are now fixed. Let α be the number of colors that u and v have in common. The strategy of the adversary now depends on α. If $\alpha \geq n/2$, then the adversary raises the weight to n on the nodes c and d, which can be observed to have distance greater than k to both u and v. The nodes of positive weight now lie on a path of length $2k + 3$. By Lemma 1.4.1 the algorithm must now use at least $2n + \frac{n}{2k+2}$ colors. If $\alpha < n/2$, the adversary raises the weight of node e, the common neighbor of a and b to n. By Lemma 1.4.1, the algorithm will have to use at least $2n + \frac{n}{2(k+1)}$ colors. The number of colors needed by the off-line algorithm is $2n$. The above construction can be repeated as many times as

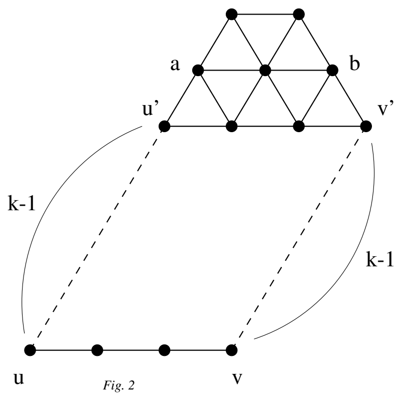

Fig. 1.4 Graph used by adversary to show a lower bound on any online algorithm with recoloring distance k.

desired, proving that there are infinitely long sequences on which the ratio bound of $1 + \frac{1}{4(k+1)}$ is achieved.

For the special case $k = 0$, a better bound can be proved [19]. Janssen *et al* also show a lower bound on the competitive ratio for any non-recoloring online algorithm. For such algorithms, the adversary can specify which colors the algorithm should drop, by which it can force the remaining colors to stay. Thus stronger lower bounds can be obtained for non-recoloring algorithms. A graph and a sequence of requests can be created such that the offline algorithm could always color the graph using n colors, but any non-recoloring online algorithm is forced to use $2n$ colors. The following theorem summarizes the lower bound results in [19]:

Theorem 1.4.2

1. *Any online algorithm with recoloring distance $k \geq 0$ has competitive ratio at least $1 + \frac{1}{4(k+1)}$.*

2. *Any online algorithm with recoloring distance 0, acting on a hexagon graph of diameter at least 3, has competitive ratio at least $1 + 2/7$.*

3. *Any non-recoloring online algorithm has competitive ratio at least 2.*

1.5 THE STATIC CASE

In this section, we describe the results on static channel assignment. The algorithms described here can be used as recoloring algorithms in the online setting as well. Hexagon graphs, unit disk graphs, odd cycles and outerplanar graphs are The best known upper bounds for the various classes of graphs are summarized in Table 1.1.

Graph	Competitive ratio of best known algorithm
Odd cycles	1 [37]
Outerplanar graphs	1 [37]
Hexagon graphs, $r = 2$	$\frac{4}{3}$ [37, 35]
Hexagon graphs, $r = 3$	$\frac{7}{3}$ [9]
Hexagon graphs, $r > 3$	4 [23]
Unit disk graphs	3 [12, 32, 40]
k-colorable graphs	$\frac{k}{2}$ [17]

Table 1.1 Best known upper bounds for recoloring algorithms

1.5.1 Reuse distance 2 in hexagon graphs

As stated in the introduction, for this case, FA has a competitive ratio of 3. To see that this is a tight bound, consider a network with a red node, blue node, and green node, no two of which are adjacent to each other, and each of which has a weight of w. It is easy to see that $\chi(G, w) = \omega(G, w)$, while FA must use different colors for each, and therefore uses 3 times the number of colors required.

Borrowing strategies. There is a simple borrowing strategy called *Fixed Preference Allocation (FPA)* that cuts the number of channels used by FA down by a factor of two. The key idea is as follows: Divide the channels into three sets of $\lceil \omega(G, w)/2 \rceil$ channels each. A red node takes as many red channels as it needs, starting from the first, and if it still needs more channels, takes green channels starting from the end.

Similarly, blue nodes borrow from red if necessary, and green nodes borrow from blue. Suppose this assignment of channels causes a conflict between a green node and a red node that are neighbors. Then their combined weight must be greater than $\omega(G, w)$, a contradiction. Thus, the assignment is conflict-free, and the algorithm is $\frac{3}{2}$-competitive.

However, the best known algorithms have a competitive ratio of $\frac{4}{3}$ [37, 35, 42], and all make use of the structure of the hexagon graph. The algorithms in [35, 42] have slightly different descriptions, but are identical, and centralized. Narayanan and Shende [37] give a different algorithm that can be implemented in a distributed fashion. Furthermore, it was modified and implemented in a local manner, as shown in [19]. The general idea in all is as follows. The algorithm consists of three phases. At the end of the first phase, the resulting graph is triangle-free. At the end of the second phase, the resulting graph is bipartite. The third phase finishes up by assigning channels to the bipartite graph.

We go on to describe the algorithm in detail now. Let $D = 3\lceil \omega(G, w)/3 \rceil$; we divide the channels into 4 sets of $D/3$ channels each. The four sets will be called the red, blue, green and purple sets of channels. As stated earlier, the algorithm can be divided into three phases. In the first phase, each node takes as many channels as needed from its nominal set of channels. For example, red nodes take as many red channels as needed and decrease their weight accordingly. Consider the resulting graph induced by nodes of positive weight. Suppose there is an triangle in this graph. This means all three nodes had weight greater than $D/3$, a contradiction. Thus the remaining graph is *triangle-free*. This implies, furthermore, that any node in the graph with degree 3 has all neighbors of the same color, and the only way it has two neighbors of different colors is if they are all in a *straight line* (see Figure 1.5 for different types of nodes of degree at least 2). We call a node a *corner node* if it is of degree 2 or 3, and all its neighbors are of the same color.

The second phase is a borrowing phase, where nodes borrow from a specified color set, and only borrow channels that are unused at any neighbors. In particular, red corner nodes with green neighbors borrow available blue channels, blue corner nodes with red neighbors borrow available green channels, and green corner nodes

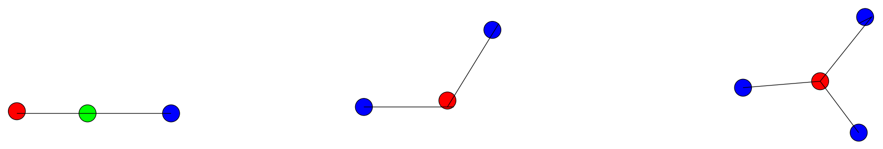

Fig. 1.5 A non-corner node of degree 2, a corner node of degree 2, and a corner node of degree 3

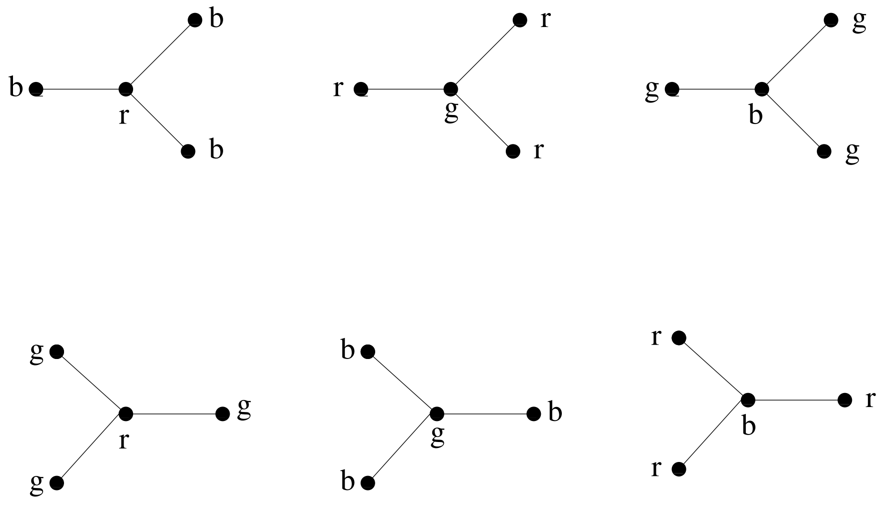

Fig. 1.6 The orientations of corner nodes. After the second phase, corner nodes of the type in the second row drop out.

with blue neighbors borrow available red channels. It is not difficult to see that all such corner nodes receive enough channels to complete their assignment, and drop out of the next phase.

From Figure 1.6, it is obvious that the resulting graph induced by nodes of positive weight at this stage cannot have a cycle, and is thus bipartite. Furthermore, any edge in this bipartite graph has weight at most $D/3$, and all nodes that are endpoints of these edges can be assigned channels from the purple set in the final phase. An isolated node with remaining weight at most $D/3$ can use purple channels, and one with remaining weight at least $D/3$ can be shown to have neighbors that are light enough so that it can finish its assignment using borrowed channels from neighbors and purple channels.

The greedy strategy. The greedy strategy is generally understood to be a non-recoloring strategy in the online setting. However, it makes sense also for the static problem, and a distributed recoloring version of the greedy strategy was formulated and analyzed in [38]. Recall that in the greedy strategy, in every step, every node is assigned the minimum numbered channels not being used by its neighbors. Some ordering of the nodes must be used in order to perform the assignment. In a distributed implementation, the ordering must take care to avoid two neighbors performing the assignment simultaneously, thereby deciding that a particular channel is available and claiming it. Prakash *et al.* [41] give a distributed implementation which focuses on how this mutual exclusion type of problem is solved, but do not give any bounds on the number of channels needed. In [38], the authors suggest a simple synchronization strategy based on rounds: first red nodes assign channels, followed by blue nodes, which are followed by green nodes. In this recoloring version of the greedy strategy, the red nodes do not have to consider any other nodes while performing their assignment, the blue nodes have to consider only the red nodes, and the green nodes have to consider all neighbors.

The key lemma used in [38] to analyze the maximum number of channels used by the greedy algorithm for an arbitrary graph and reuse distance is the following:

Lemma 1.5.1 *Let $mc(v)$ be the highest channel used by the node v. For the greedy algorithm, and for any node v, $mc(v) \leq min\{H_r(v) - RC_r(v), w(v) + max_{u \in N_r(v)} mc(u)\}$.*

The number of distinct channels used by nodes in $N_r(v)$ is at most $H_r(v) - w(v) - RC_r(v)$. Therefore, v will never use a channel higher than $H_r(v) - RC_r(v)$. Also, if u is a node in $N_r(v)$ that uses the highest channel in v's neighborhood, v will never use more than the next $w(v)$ channels.

For the case of reuse distance 2 in hexagon graphs, the above lemma can be used to show that the greedy algorithm can never use a channel higher than $5\omega(G, w)/3$. To see this, observe first that red and blue nodes can never use a channel higher than

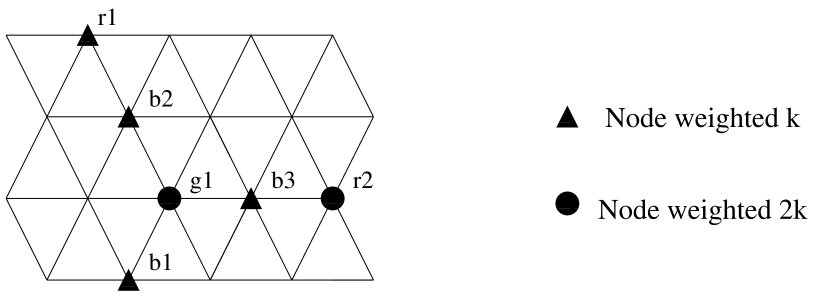

Fig. 1.7 An example where the greedy algorithm uses $5\omega(G, w)/3$ channels.

$\omega(G, w)$. For a green node v, if $w(v) \leq 2\omega(G, w)/3$, then since no neighbor of v uses a channel higher than $\omega(G, w)$, we have $mc(v) \leq \omega(G, w) + w(v)$. If instead $w(v) > 2\omega(G, w)/3$, then $mc(v) \leq H_2(v) \leq 5\omega(G, w)/3$. Finally, this is a tight bound, as shown by Figure 1.7. The reader can verify that $\omega(G, w) = 3k$, and that there is an optimal assignment using $3k$ channels, but the greedy algorithm will use $5k$ channels. Thus, the greedy algorithm uses $5/3$ times the optimal number of channels required.

Distributed algorithms. Janssen *et al.* [19] give a framework to analyze distributed algorithms. In particular, an algorithm is said to be k-local, if the only information available to a node apart from its own history, is the current weights of nodes that are within graph distance k from it. However, since the nodes themselves communicate via a wireline network, it is reasonable to allow nodes to exchange other information with nodes in their k-locality. In particular, a node may send the list of channels it is using to the nodes in its k-locality. By this revised definition, the greedy algorithm and FPA can be seen to be 1-local. While the $4/3$-approximation algorithm described earlier is not distributed, two distributed implementations of the algorithm in [37] are given in [19]. Algorithms with better competitive ratios are shown for increasing values of locality. The best known results for k local algorithms for small values of k are summarized in Table 1.2. The corresponding lower bounds that follow from Theorem 1.4.2 are also given for comparison.

Recoloring distance	Lower bound on competitive ratio	Best known
0-local	9/7	3
1-local	9/8	3/2
2-local	13/12	17/12
4-local	21/20	4/3

Table 1.2 Best known k-local algorithms

1.5.2 Reuse distance 3 in hexagon graphs

For reuse distance 3, since there is a 7-coloring for any square graph, FA has a competitive ratio of 7. The borrowing strategy given by [17] has a competitive ratio of 3.5. The best known algorithm was given by Feder and Shende [9]. We describe it briefly here. The graph is divided into seven independent sets, according to the base coloring, and each set is assigned a nominal set of ℓ channels. Call a node *heavy* if its weight is greater than ℓ and light otherwise. The algorithm proceeds in two phases. In the first phase, each node takes as many channels as needed from its own nominal set. All light nodes are done in this phase and drop out. In the second phase, any remaining (heavy) node borrows as many unused channels as needed from its neighbors' nominal sets.

It remains to show that the unused channels in a heavy node v's neighborhood suffice to finish v's assignment. Note that the neighborhood of any node v can be covered by four cliques (see Figure 1.8). Thus the total weight of the neighborhood of v is at most $4\omega(G, w) - 3w(v)$, and this forms an upper bound on the number of channels needed for the algorithm to work. For $\ell = 2\omega(G, w)/5$, for example, it is clear that $4\omega(G, w) - 3w(v) \leq 7\ell$ for any heavy node v. However, this would give an algorithm with performance ratio 2.8.

The authors perform a more careful analysis of the neighborhood of a heavy node v to show that $\ell = \omega(G, w)/3$ suffices. For instance, they observe that a node v of base color i has three neighbors of each base color $\neq i$, but many of the channels used

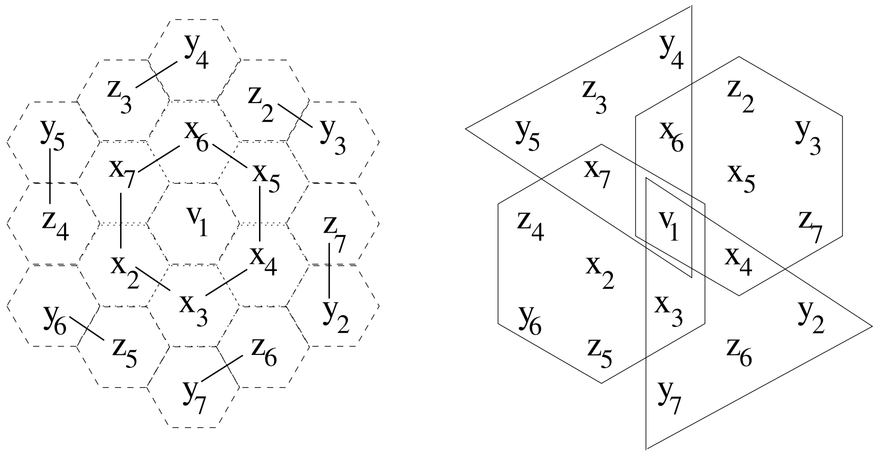

Fig. 1.8 The neighborhood of a node v contains exactly three nodes of each base color different from v's, and can be covered with three cliques.

are the same. Thus some of the weight of the two lighter neighbors can be charged to the heaviest neighbor of that color. This means that for a heavy node v, if all three of its neighbors of a particular base color are light, then only one of them counts, and the other two can be considered to be of zero weight, since all the channels assigned to them would also be assigned to the heaviest neighbor. Another observation is that there cannot be too many heavy nodes in the neighborhood of a heavy node v. In particular, there can either be at most four heavy nodes if all are *outer* neighbors of v (see Figure 1.8), or at most three if one of them is an *inner* neighbor of v.

Thus there are only four possibilities for v's neighborhood: It contains (a) at most one node of each distinct base color, except for two pairs of heavy nodes, each corresponding to a base color different from v's and from each other (b) at most one node of each distinct base color except for a triple of heavy nodes, all of the same color (c) at most one node of each distinct color and (d) at most one node of each distinct color, except for a pair of heavy nodes of a color different from v's. In the first two cases, there are two nodes whose demand can effectively be reduced by $\omega(G, w)/3$ since this corresponds to reused channels. Thus, the total weight of v's neighborhood is at most $4\omega(G, w) - 3w(v) - 2\omega(G, w)/3 \leq 7\omega(G, w)/3$, which implies that there are sufficient channels for v to borrow to complete its assignment.

The other cases are more complicated, but lead to the same conclusion. See [9] for details.

It is possible to construct a square graph G such that $\chi(G, 2) = \omega(G, w)$, but the algorithm needs $7\omega(G, w)/3$ channels, thus the competitive ratio of the algorithm is exactly $7/3$.

We note here that Feder and Shende mention a centralized strategy for this case that has competitive ratio at most 2. This uses a simple idea involving a convex hull technique. The algorithm follows from the following simple observation. Consider an embedding of the square graph on the triangular lattice. Extract a node on the convex hull of the graph. It is easy to see that its neighborhood can be covered with two cliques. Hence if every node in v's neighborhood is already assigned channels from the set $[1, \ldots, 2\omega(G, w)]$, then v can also find channels from this set without conflicting with any neighbors. Therefore, the algorithm first constructs an ordering of nodes by computing the convex hull of the nodes, removing a hull node, and repeating this process on the remaining nodes. Finally, nodes are assigned channels in the reverse of this ordering. It is clear that $2\omega(G, w)$ channels suffice.

In [38], the authors show that the recoloring greedy strategy has a competitive ratio between $7/3$ and $23/8$ for reuse distance 3 in hexagon graphs, and thus for the recoloring case, the borrowing strategy is at least as good as the greedy strategy.

1.5.3 Arbitrary graphs

Janssen and Kilakos [17] show that for general k-colorable graphs, a generalization of the FPA strategy discussed earlier has a performance ratio of $k/2$. In what follows, we discuss some specific classes of graphs that have been studied in the context of radio channel assignment. These include odd cycles and outerplanar graphs, graphs derived from hexagon graphs, either by considering arbitrary reuse distance in such graphs, or by considering networks that have a certain amount of overlap between adjacent cells. Another interesting class of graphs is the unit disk graphs and graphs that can be drawn in an (r, s)-civilized manner.

Odd cycles and outerplanar graphs Odd cycles and outerplanar graphs can be multicolored optimally using simple linear time sequential algorithms. The chromatic number of an odd cycle with $2m+1$ nodes is $max\{\omega(G,w), \lceil W/m \rceil\}$ where W is the sum of the weights of all nodes in the cycle. The centralized algorithm given in [37] first finds the minimum index k such that $\Sigma_{i=1}^{2k+1} w_i \leq k\chi(G,w)$. Such an index must exist, because m satisfies the above property. Then the algorithm uses contiguous colors in a cyclic manner from the colors $[1, \chi(G,w)]$ to color nodes 1 to $2k$. The remaining nodes are colored based on their parity as in a bipartite graph. The optimal algorithm for odd cycles is extended in [37] to derive an optimal algorithm to multicolor outerplanar graphs.

Arbitrary reuse distance in hexagon graphs. We consider the case of arbitrary reuse distance for hexagon graphs. The problem of channel assignment in a hexagon graph (G,w) with reuse distance r is equivalent to multicoloring the graph (G^r,w). As stated in Section 1.4, there exist graphs (G_r,w) which require $5\omega(G^r,w)/4$ channels. An upper bound of 6 for the performance ratio of the greedy strategy follows from the observation that the neighborhood of any node in a graph G^r can always be covered with 6 cliques. However, a borrowing strategy similar to the one in [9] for reuse distance 3 has a better upper bound. In particular, as shown in [35] and [45], for a hexagon graph G with reuse distance r, $\omega(G^r) = \chi(G^r) = \frac{3r^2}{4}$ if r is even and $\frac{3r^2+1}{4}$ otherwise. This was used in [38] to derive an algorithm for channel assignment with performance ratio $\frac{18r^2}{3r^2+20}$ if r is even, and $\frac{18r^2+6}{3r^2+21}$ when r is odd.

The best known performance ratio for channel assignment for reuse distance $r > 3$ is achieved by an algorithm called *cluster partitioning* in [23]. The key idea is to partition the graph into clusters which are maximal cliques, as with FA. However, unlike FA, where identical sets of channels are assigned to corresponding cells in different clusters, here, sets of D channels are assigned to entire clusters, in such a way that any pair of clusters containing cells that are within distance $r-1$ are assigned different sets. Calls arising in any cluster are assigned channels from its nominal set of channels. Furthermore, it turns out that it is possible to color the clusters with 4 colors such that any two clusters that have nodes within distance $r-1$

of each other get different colors. Thus four sets of channels suffice, which implies a performance ratio of 4 for the algorithm.

Overlapping cells. Matula *et al.* [34] define a model for channel assignment in cellular networks that include overlapping cell segments. In particular, instead of assuming the network to consist of disjoint hexagonal cells, each served by a single base station, they assume that adjacent cells include an overlap region that can be covered by both transceivers. Cell segments thus fall into two classes: those that can be covered by a single transceiver and those that can covered by two [1]. The channel assignment problem is then modeled by a regular tripartite graph with three sets of nodes: (a) transceiver nodes with degree 7, (b) cell segment nodes with degree 1, and (c) cell segment nodes with degree 2. There is an edge between transceiver node i and cell segment j if transceiver node i covers the segment j. A channel assignment is an assignment of integers to edges such that the total weight assigned to all edges outgoing from a transceiver node is at most the number of channels it holds, and the total weight on all edges incoming to a cell segment is equal to the number of mobiles in the cell segment. Using network flow techniques, they show a *Capacity-Demand* theorem, which states that a channel assignment is always possible unless there is a connected sub-region of cell segments containing more channel requests than the total capacity of all transceivers within or on the boundary of the subregion and covering any part of the subregion with an overlapping segment.

A subsequent paper [6] uses the same model as described above. The authors propose a new load balancing algorithm called the *cluster algorithm* and show that it has a competitive ratio of 4 when applied to the channel assignment problem.

Unit disk graphs. Unit disk graphs are the intersection graphs of equal sized disks in the plane. It is easy to see that, given the distance model, multicoloring is equivalent to coloring for unit disk graphs: a node with weight w can be replaced with w nodes of weight 1 in close proximity of each other. Such graphs can be seen as a more

[1] If the overlap segments are larger, then there is a third type of segment which can be covered by three transceivers.

general model of cellular networks. Also, a hexagon graph for any reuse distance r is a unit disk graph.

Unit disk graphs can be used to model broadcast networks where the nodes are transceivers of equal power and two nodes that are within transmission range of each other are within each other's disks and are therefore neighbors. Neighboring nodes may not broadcast simultaneously; coloring a unit disk graph thus gives a broadcast schedule for the corresponding adhoc network. The number of colors required is the number of rounds required in the broadcast schedule.

In [12], it is shown that the coloring problem for unit disk graphs is NP-complete, for any fixed number of colors $k \geq 3$, even if the distance model is given. 3-approximation algorithms for the problem are given in [12, 32, 40]. The algorithm in [12] works as follows. Divide the part of the plane containing the unit disk graph into strips of width $\sqrt{3}/2$. The induced subgraph in each strip is a co-comparability graph, and can therefore be colored optimally using $\omega(G, w)$ colors. Furthermore, a node in a given strip may be a neighbor of a node in its own strip, an adjacent strip or a node that is 2 strips away, but no other nodes in the graph. Therefore, three disjoint sets of $\omega(G, w)$ colors can be used to color the nodes in three adjacent strips, and then the sets of colors can be reused. Marathe *et al.* [32] use an algorithm by Hochbaum [14] for coloring arbitrary graphs in $\delta(G) + 1$ colors (where $\delta(G)$ is the maximum δ such that G contains a subgraph in which every node has degree at least δ) and prove that it has performance ratio 3 for unit disk graphs. Peeters [40] has shown that coloring the nodes greedily and sequentially using a lexicographic ordering of the nodes also achieves a performance ratio of 3.

McDiarmid and Reed [36] study the case when the number of nodes is infinite. For V a countable set of points in the plane, the *upper density* of V is defined as $inf_{x>0} f(x)$ where $f(x)$ is the supremum of the ratio $|V \cap S|/x^2$ over all open $x \times x$ squares S with sides aligned with the axes. They show that for any countable set of points V in the plane with a finite positive upper density, the ratio of chromatic number by clique number tends to to $2\sqrt{3}/\pi$ which is about 1.103 as the radius of the disks tends to infinity.

Marathe *et al.* [33] give approximation algorithms for the coloring and distance 2-coloring problem in graphs that can be drawn in an (r, s)-civilized manner.

1.6 THE ONLINE CASE

In this section, we describe algorithms for online channel assignment that do not perform reassignment of channels. Once a call is assigned a channel, it "keeps" the channel for its entire duration. While there has been a lot of work on online graph coloring (see, for example, [16, 29, 46]), there has been very little work on online graph multicoloring. The best known upper bounds for the various classes of graphs are summarized in Table 1.3.

Graph	Competitive ratio of best known algorithm
Hexagon graphs, $r = 2$	3 [30]
Hexagon graphs, $r = 3$	4 [9]
Hexagon graphs, $r > 3$	4 [23]
Unit disk graphs	6 [32]
k-colorable graphs	k [17]

Table 1.3 Best known upper bounds for non-recoloring algorithms

1.6.1 Reuse distance 2 in hexagon graphs

FA is a non-recoloring strategy that has a competitive ratio of 3. A lower bound of 2 for this case was derived in [19] for the competitive ratio of any non-recoloring strategy. The only other non-recoloring strategy to have been investigated in terms of its competitive ratio is the greedy strategy. In this algorithm, every node simply uses the lowest numbered channels that are not being used currently by any of its neighbors. In [4], a careful case-by-case analysis of the neighborhood of a node was used to show an upper bound of $2.5\omega(G, w)$ for the number of channels used by this algorithm, under the restriction that calls are of infinite duration. In contrast, in [38], a graph and sequence of weight vectors was given to show that the greedy algorithm would use $2.5\omega(G, w)$ channels while the optimal offline algorithm would use only $\omega(G, w)$ channels. Thus the competitive ratio of the greedy algorithm was shown to be at least 2.5.

1.6.2 Reuse distance 3 in hexagon graphs

FA has a competitive ratio of 7 for this case. The only other non-recoloring strategy to have been studied for this case is the greedy strategy. Since the neighborhood of any node can be covered with 4 cliques (see Figure 1.8), it is easy to see that $4\omega(G, w)$ channels will suffice for the greedy strategy. In [38], a lower bound of 3 is shown on the competitive ratio of the greedy strategy: a graph and a sequence of weight vectors are given on which the greedy strategy uses at least three times the number of channels needed by an offline algorithm.

1.6.3 Arbitrary graphs

For a k-colorable graph, FA has a competitive ratio of k. For a hexagon graph with arbitrary reuse distance, as already mentioned, the neighborhood of a node can be covered with 6 cliques. Thus, the greedy strategy has a competitive ratio of at most 6 for this case. The greedy strategy is also shown to have a competitive ratio of 6 for unit disk graphs in [32]. The cluster partitioning algorithm has a competitive ratio of 4 for hexagon graphs with arbitrary reuse distance [23].

1.7 DISCUSSION AND OPEN PROBLEMS

The problem of channel assignment has been extensively studied over the last decade. However, many key questions remain unanswered as yet. For reuse distance 2, in the static case, what is a tight bound on the number of channels required in the worst case? It is known that $4\omega(G, w)/3$ channels suffice, and that there is a graph requiring $9\omega(G, w)/8$ channels. In the online case, where once assigned a channel, a call retains it for its duration, what is a tight bound on the competitive ratio of an online algorithm? It is known that 2 is a lower bound, while the greedy algorithm achieves a competitive ratio of 2.5. Table 1.2 demonstrates that for increasing values of k, distributed algorithms in which nodes have access to information about their k-localities can achieve better competitive ratios. At the same time, the limitations posed by restricting the locality of an algorithm are not completely understood yet.

For reuse distance 3, very little is known, and the gap between the known lower bound and best known upper bound is quite wide. While a few authors have proposed randomized algorithms, there are no known bounds on the competitive ratios of these algorithms.

Prakash *et al.* [41] propose several desirable features of channel assignment algorithms. Some of these are features such as minimizing connection set-up time, and energy conservation at mobile hosts, which would be true of the distributed algorithms studied here. Another quality that is considered important is minimizing the number of hand-offs. While inter-cell handoffs were not taken into account by any of the algorithms described here, clearly the non-recoloring algorithms do not create any intra-cell handoffs. On the other hand, the recoloring algorithms can potentially force many intra-cell handoffs to occur. Designing algorithms that would limit the number of such handoffs would be an interesting avenue for future research. Finally, adaptability to load distributions is a desirable feature of channel assignment algorithms. While it is clear that dynamic channel allocation does better than fixed allocation in this regard, it would be useful to know where the borrowing strategies stand in the picture.

REFERENCES

1. N. Alon and M. Tarsi. Colorings and orientations of graphs. *Combinatorica*, 12(2):125–134, 1992.

2. J. Bondy and U. Murty. *Graph Theory with Applications*. Macmillan Press Ltd, 1976.

3. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

4. I. Caragiannis, C. Kaklamanis, and E. Papaionnou. Efficient online communication in cellular networks. In *Symposium on Parallel Algorithms and Architecture*, 2000.

5. D. C. Cox and D. O. Reudink. Dynamic channel assignment in two dimension large-scale mobile radio systems. *Bell Sys. Tech. J.*, 51:1611–28, 1972.

6. P. Cresenzi, G. Gambosi, and P. Penna. Online algorithms for the channel assignment problem in cellular networks. In *Proceedings of Dial M for Mobility*, 2000.

7. S. Engel and M. M. Peritsky. Statistically-optimum dynamic server assignment in systems with interfering servers. *IEEE Transactions on Vehicular Technology*, 22:203–209, 1973.

8. P. Erdos, A. L. Rubin, and H. Taylor. Choosability in graphs. In *Proceedings of the West Coast Conference on Combinatorics, Graph Theory, and Computing*, pages 125–157, 1979.

9. T. Feder and S. M. Shende. Online channel allocation in FDMA networks with reuse constraints. *Inform. Process. Lett.*, 67(6):295–302, 1998.

10. A. Gamst. Some lower bounds for a class of frequency assignment problems. *IEEE Trans. Veh. Technol.*, 35(1):8–14, 1986.

11. N. Garg, M. Papatriantafilou, and T. Tsigas. Distributed list coloring: how to dynamically allocate frequencies to mobile base stations. In *Symposium on Parallel and Distributed Processing*, pages 18–25, 1996.

12. A. Graf, M. Stumpf, and G. Weibenfels. On coloring unit disk graphs. *Algorithmica*, 20:277–293, 1998.

13. W. K. Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

14. D. S. Hochbaum. Efficient bounds for the stable set, vertex cover, and set packing problems. *Discrete Applied Mathematics*, 6:243–254, 1983.

15. C. L. I and P. H. Chao. Local packing-distributed dynamic channel allocation at cellular base station. *Proceedings of GLOBECOM*, 1993.

16. S. Irani. Coloring inductive graphs online. In *Symposium on the Foundations of Computer Science*, pages 470–479, 1990.

17. J. Janssen and K. Kilakos. Adaptive multicolourings. *Combinatorica*, 20(1):87–102, 2000.

18. J. Janssen, K. Kilakos, and O. Marcotte. Fixed preference frequency allocation for cellular telephone systems. *IEEE Transactions on Vehicular Technology*, 48(2):533–541, March 1999.

19. J. Janssen, D. Krizanc, L. Narayanan, and S. Shende. Distributed online frequency assignment in cellular networks. *J. of Algorithms*, 36:119–151, 2000.

20. J. M. Janssen. *Channel assignment and graph labeling*. In Ivan Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*. John Wiley and Sons, 2001.

21. B. Jaumard, O. Marcotte, and C. Meyer. Mathematical models and exact methods for channel assignment in cellular networks. In B. Sansó and P. Soriano, editors, *Telecommunications Network Planning*. Kluwer, 1999.

22. T. R. Jensen and B. Toft. *Graph Coloring Problems*. Wiley Interscience Series in Discrete Mathematics and Optimization, 1995.

23. S. Jordan and E. J. Schwabe. Worst-case performance of cellular channel assignment policies. *Wireless Networks*, 2:265–275, 1996.

24. T. Kahwa and N. Georganas. A hybrid channel assignment scheme in large-scale cellular-structured mobile communication systems. *IEEE Transactions on Communications*, 4:432–438, 1978.

25. A. Karlin, M. Manasse, L. Rudolph, and D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):70–119, 1988.

26. I. Katzela and S. Naghshineh. Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey. *IEEE Personal Communications*, pages 10–31, 1996.

27. L. Le Bris. A simple randomized algorithm for the fixed frequency assignment problem. 1997.

28. S. Leonardi, A. Marchetti-Spaccamela, A. Prescuitti, and A. Rosen. Online randomized call control revisited. In *Symposium on Discrete Algorithms*, pages 323–332, 1998.

29. L. Lovasz, M. Saks, and W. Trotter. An online graph coloring algorithm with sub-linear performance ratio. *Discrete Math*, 75:319–325, 1989.

30. V. H. MacDonald. Advanced mobile phone service: The cellular concept. *Bell Systems Technical Journal*, 58(1), 1979.

31. E. Malesinsca. An optimization method for the channel assignment in mixed environments. In *Proceedings of MOBICOM*, 1995.

32. M. V. Marathe, H. Breu, H. B. Hunt, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25:59–68, 1995.

33. M. V. Marathe, S. O. Krumke, and S. S. Ravi. Approximation algorithms for broadcast scheduling in radio networks. In *Proceedings of Dial M for Mobility*, 1998.

34. D. Matula, M. Iridon, C. Yang, and H. C. Cankaya. A graph-theoretic approach for channel assignment in cellular networks. In *Proceedings of Dial M for Mobility*, 1998.

35. C. McDiarmid and B. Reed. Channel assignment and weighted colouring. Submitted for publication, 1997.

36. C. McDiarmid and B. Reed. Colouring proximity graphs in the plane. *Discrete Mathematics*, 199:123–137, 1999.

37. L. Narayanan and S. Shende. Static frequency assignment in cellular networks. *Algorithmica*, 29:396–409, 2001.

38. L. Narayanan and Y. Tang. Worst-case analysis of a dynamic channel assignment strategy. In *Proceedings of Dial M for Mobility 2000*, pages 215–227, 2000.

39. G. Pantizou, G. Penatris, and P. Spirakis. Competitive call control in mobile networks. In *Proceedings of ISAAC '97*, Springer-Verlag Lecture Notes in Computer Science, pages 404–413, 1997.

40. R. Peeters. On coloring j-unit sphere graphs. Technical Report FEW 512, Department of Economics, Tilburg University, 1991.

41. R. Prakash, N. Shivaratri, and M. Singhal. Distributed dynamic channel allocation for mobile computing. In *Principles of Distributed Computing*, pages 47–56, 1995.

42. N. Schabanel, S. Ubeda, and Zerovnik. A note on upper bounds for the span of frequency planning in cellular networks. *Submitted for publication*, 1997.

43. M. Serizawa and D. Goodman. Instability and deadlock of distributed dynamic channel allocation. *Proc. 43rd IEEE VTC*, pages 528–31, 1993.

44. C. Thomassen. Every planar graph is 5-choosable. *Journal of combinatorial theory, Series B*, 62:180–181, 1994.

45. J. van den Heuvel, R. A. Leese, and M. A. Shepherd. Graph labeling and radio channel assignment. *Journal of Graph Theory*, 29:263–283, 1999.

46. S. Vishwanathan. Randomized online graph coloring. *Journal of Algorithms*, 13:657–669, 1992.

47. W. Yue. Analytical methods to calculate the performance of a cellular mobile radio communication system with hybrid channel assignment. *IEEE Transactions on Vehicular Technology*, pages 453–459, 1991.

48. J. Zerovnik. On a distributed randomized algorithm for frequency assignment. In *Proceedings of the High Performance Computing Symposium*, pages 399–404, 1999.

49. M. Zhang and T.-S. P. Yum. Comparisons of channel assignment strategies in cellular mobile telephone systems. *IEEE Transactions in Vehicular Technology*, 38:211–215, 1989.