Algorithmica (2001) 29: 396–409
DOI: 10.1007/s004530010067

Algorithmica


# Static Frequency Assignment in Cellular Networks[1]

L. Narayanan[2] and S. M. Shende[3]

**Abstract.** A cellular network is generally modeled as a subgraph of the triangular lattice. In the static frequency assignment problem, each vertex of the graph is a base station in the network, and has associated with it an integer weight that represents the number of calls that must be served at the vertex by assigning distinct frequencies per call. The edges of the graph model interference constraints for frequencies assigned to neighboring stations. The *static frequency assignment* problem can be abstracted as a graph multicoloring problem. We describe an efficient algorithm to multicolor optimally any weighted even or odd length cycle representing a cellular network. This result is further extended to any outerplanar graph. For the problem of multicoloring an arbitrary connected subgraph of the triangular lattice, we demonstrate an approximation algorithm which guarantees that no more than $\frac{4}{3}$ times the minimum number of required colors are used. Further, we show that this algorithm can be implemented in a distributed manner, where each station needs to have knowledge only of the weights at a small neighborhood.

**Key Words.** Frequency assignment, Cellular networks, Approximation algorithms, Graph multicoloring, Distributed algorithms.

**1. Introduction.** Cellular data and communication networks can be modeled as graphs with each vertex representing a base station (sometimes called a *cell*) in the network. Cells can communicate with their neighbors in the graph via directional radio transceivers. At any given time, a certain number of active connections (or *calls* in cellular network terminology) are serviced by their nearest base station. This service consists mainly of assigning a frequency to each client call in a manner that minimizes or avoids radio interference between two distinct calls in the network. However, cellular networks use a fixed spectrum of radio frequencies and the efficient shared utilization of the limited available bandwidth is critical to the viability and efficiency of the network. The *static frequency assignment* problem, therefore, consists of designing an interference-free frequency allocation protocol for a network where the number of calls per cell is known a priori. This forms the motivation for the problems studied in this paper.

In particular,cellular networks are usually modeled as finite portions of the infinite triangular grid embedded in the plane. Vertices representing cells are placed at the apexes of similar triangles, and each vertex has at most six other neighbors surrounding it in the grid. The reason for adopting this particular geometry stems from the fact that cells are uniformly distributed in the geographic area of the network, and an individual cell

[1] The research of the first author was supported by NSERC, Canada. The research of the second author was conducted at Concordia University during the summer of 1996, with partial support from NSERC, Canada.
[2] Department of Computer Science, Concordia University, Montreal, Quebec, Canada H3G 1M8. lata@cs.concordia.ca.
[3] Department of Computer Science, Rutgers University, Camden, NJ 08102, USA. shende@crab.rutgers.edu.

Received May 13, 1997; revised August 24, 1998. Communicated by D. S. Johnson.
Online publication October 25, 2000.

generally has six directional transceivers. Hence, the Voronoi region around a cell (or, equivalently, that cell's calling area) can be idealized as a regular hexagon. The triangular tiling representing the network is simply the planar dual of the resulting Voronoi diagram. We refer to the resulting graphs as *hexagon* graphs.[4]

The frequency assignment problem incorporating interference constraints can be abstracted as follows. Let $G = (V, E)$ denote a hexagon graph. Each vertex $v \in V$ has an associated integer weight, $w(v) \geq 0$. A $w$-coloring (or multicoloring) of $G$ is an assignment of sets of colors to the vertices such that each vertex $v$ is assigned $w(v)$ *distinct* colors whereby for every edge $(u, v) \in E$, the set of colors assigned to the endpoints $u$ and $v$ are *disjoint*. In particular, we are interested in a *minimum multicoloring* or a $w$-coloring of $G$ that uses the least number of colors.

In the context of frequency assignment, a multicoloring as defined above, provides a useful abstraction of the essential interference constraints: each color represents a distinct frequency and it is assumed that two calls may use the same frequency if and only if they originate in distinct cells that are not neighbors. It should be noted that in practice, the available cellular frequency spectrum is a contiguous linear subinterval of the radio spectrum, and frequency reuse is controlled by a sequence of nonnegative integers, $c_0 \geq c_1 \ldots$, with $c_0 \geq 1$, called distance reuse constraints. Two distinct calls in cells that are a distance $i$ apart in the underlying graph must be assigned frequencies that differ by $c_i$ in the frequency spectrum. Hale [3] discusses many generalizations and versions of the frequency assignment problem. We formulate our problem under the simplest constraints, namely when $c_0 = c_1 = 1$ and $c_i = 0$, $i \geq 2$. Under this formulation, the problem reduces to being able to compute a minimum multicoloring to a given hexagon graph.

In what follows we assume that $G = (V, E, w)$ denotes a hexagon graph, i.e. it is a (vertex) weighted graph that is a finite, induced subgraph of the infinite triangular grid. Thus, the graph is planar, and every vertex $v \in V$ has degree at most 6 and an associated integer weight, $w(v) \geq 0$. The weighted chromatic number of $G$, denoted $\chi(G)$, is the minimum number of colors required in a $w$-coloring of $G$. Even for graphs with a regular structure such as those considered in the paper, the problem of determining $\chi(G)$ is nontrivial. In fact, it has been established only recently that the corresponding decision problem is NP-complete [7], and hence it is unlikely that a polynomial time algorithm for computing $\chi(G)$ can be devised. Naturally, it is of interest to study approximation algorithms for the problem.

It is easy to see that $\chi(G)$ must be greater than the total number of colors required at any set of mutually adjacent vertices. Thus the maximum over the sum of weights at vertices in any maximal clique in the graph is a trivial lower bound on $\chi(G)$. Note that for hexagon graphs, edges and triangles are maximal cliques. In the direction of upper bounds, while there is a vast literature on algorithms for frequency assignment on graphs (especially hexagon graphs) that claim to use few colors, generally there are no *proven*

[4] We note here that this is the most commonly used model for frequency assignment problems in the cellular network literature. However, in practice, cellular systems tend to be more complicated and there have been recent studies that attempt to model more general interference patterns and cost functions; see for example [1]. Nevertheless, the hexagon graph model continues to be of significance from the historical standpoint and as an abstraction that is sufficiently close to reality to provide useful insights.

*bounds* on the performance of the proposed algorithms, in terms of the number of colors used in relation to the weighted chromatic number [2], [5], [6], [8], [9]. We note here two exceptions. A well-known algorithm, sometimes referred to as *Fixed Allocation*, uses the fact that the underlying graph can be 3-colored. The algorithm uses three fixed sets of colors, one for each base color. A vertex that has base color 1 uses colors from the first set, and a vertex that has base color 2 or 3 uses colors from the second or third sets, respectively. It is easy to show that this algorithm could use as many as three times the number of required colors. Janssen et al. [4] propose a different algorithm called *Fixed Preference Allocation* that is guaranteed to use no more than $\frac{3}{2}$ times the minimum number of colors required.

In the next section we formally define some basic terminology and problems. In Section 3 we present optimal algorithms for multicoloring cycles and outerplanar graphs. In Section 4 we address the question of multicoloring an arbitrary hexagon graph. Our main result is an efficient approximation algorithm with a performance guarantee of within $\frac{4}{3}$ of the optimal. Finally, in Section 5, we show how to implement the above algorithm in a distributed manner.[5] We conclude with a discussion of future directions in Section 6.

**2. Preliminaries.** Let $G = (V, E, w)$ be a hexagon graph with a nonnegative integer weight vector $w$ defined on the vertices of the graph, where $w(v)$ represents the number of calls to be served at vertex $v$. We assume hereafter that $G$ has a *fixed* planar embedding with vertices and edges contained in the infinite triangular lattice (tessellation) of the plane. Thus any vertex $v$ can be connected to at most six neighbors, and for a fixed edge incident on $v$, any other edge incident on $v$ is at an angle of $\pi/3$, $2\pi/3$, $\pi$, $4\pi/3$ or $5\pi/3$ from that edge. Since the triangular lattice is 3-colorable in the ordinary sense (i.e. when each vertex has unit weight), the underlying graph corresponding to unit weights at vertices of $G$ is also 3-colorable.

A $w$-coloring or multicoloring of the graph $G = (V, E, w)$ consists of a set of colors $C$ (the color palette) and a function $f$ that assigns to each $v \in V$ a subset $f(v)$ of the palette $C$ such that

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

The *span* of a multicoloring is the cardinality of the set $C$. The *weighted chromatic number* of $G$, denoted $\chi(G)$, is the smallest number $m$ such that there exists a multicoloring of $G$ of span $m$. Thus given a hexagon graph $G$, our objective is to find a multicoloring for $G$ whose span is as close to $\chi(G)$ as possible.

The only maximal cliques in $G$ being edges and triangles, we define the weight of an edge (triangle) in $G$ to be the sum of the weights of its endpoints (apexes). Note that the weight of any maximal clique of $G$ is a lower bound on $\chi(G)$. Let $D_G^{[2]}$ and $D_G^{[3]}$

[5] We note that in addition to showing the NP-hardness of this problem, McDiarmid and Reed [7] have also described a different $\frac{4}{3}$-approximate algorithm for the problem. Our result was independently derived, and, unlike the McDiarmid–Reed algorithm, has the advantage of being implementable in a distributed setting.

denote the respective maxima over the weights of edges and triangles in $G$, and define $D_G = \max\{D_G^{[2]}, D_G^{[3]}\}$. Then, if there exists a multicoloring of $G$ with span $\Delta$, it follows that

$$\Delta \geq \chi(G) \geq D_G.$$

We assume without loss of generality that any palette of available colors can be suitably ordered or partitioned; in particular, we often assume that vertices are assigned colors from the circularly ordered interval $[1, M] = \{1, 2, \ldots, M\}$, where $M \geq 1$ is a positive integer that depends on the particular graph under consideration. For instance, when a vertex is assigned the subinterval of colors $[i, j]$ from the palette, it means that the vertex is colored with the set $\{i, i+1, \ldots, j\}$ in a cyclic manner where color 1 is assumed to follow the color $M$.

**3. Optimal Multicoloring of Cycles.** Consider a hexagon graph $G = (V, E, w)$ with $n$ vertices in the form of a simple cycle, labeled $u_1, u_2, \ldots, u_n$ in clockwise order. For simplicity, let $w_i$, $1 \leq i \leq n$, denote the weight, $w(u_i)$, of vertex $u_i$. We show that any such hexagon graph can be *optimally* colored with exactly $\chi(G)$ colors. There are two cases to consider depending on whether $n$, the number of vertices on the cycle, is even or odd.

Suppose that $n = 2m$, i.e. the graph consists of an even length cycle. Then all maximal cliques of $G$ being edges, $D_G = D_G^{[2]}$ is the maximum weight of an edge in the cycle. We observe that a very simple greedy strategy suffices to multicolor $G$ with the color palette $[1, D_G]$. The idea is to assign for $1 \leq i \leq m$, the colors $[1, w_{2i-1}]$ to the odd-numbered vertex $u_{2i-1}$ and the colors $[D_G - w_{2i} + 1, D_G]$ to the even-numbered vertex $u_{2i}$. Noting that, for $1 \leq i \leq m$,

$$\begin{aligned} D_G &\geq w_{2i-1} + w_{2i} \quad \text{and} \\ D_G &\geq w_{2i} + w_{2i+1} \end{aligned}$$

with subscripts interpreted cyclically, it follows that the given multicoloring is proper. By construction, $\chi(G) = D_G$ and the simple parity-based algorithm thus provides an optimal multicoloring of $G$.

We note that a very similar idea has already been used in the cellular network literature [8], [4], but it was only applied to networks consisting of simple paths (it is easy to see that this strategy works in general for any bipartite graph). Unfortunately, the parity argument fails to multicolor odd-length cycles, precisely because the underlying unweighted odd-length cycle needs at least three colors in any ordinary coloring. For instance, if $G$ is a 9-cycle with weight 2 on each vertex, it is easy to see that $G$ cannot be multicolored with $D_G = 4$ colors, but needs 5 colors instead.

DEFINITION 3.1. Let $G = (V, E, w)$ be an odd-length simple cycle, with vertices labeled $u_1, u_2, \ldots, u_{2m+1}$, $m \geq 1$, in clockwise order. We define

$$D'_G = \max\left\{ D_G^{[2]}, \left\lceil \frac{\sum_{i=1}^{2m+1} w_i}{m} \right\rceil \right\}.$$

THEOREM 3.2. *Let $G = (V, E, w)$ be a cycle of odd length $n = 2m + 1 \geq 3$. Then $\chi(G) = D'_G$ and $G$ can be optimally multicolored with exactly $D'_G$ colors. Further, the multicoloring can be obtained in time $O(n)$.*

PROOF. It is clear that $D^{[2]}_G$ is a lower bound on $\chi(G)$; we establish that $\lceil \sum_{i=1}^{2m+1} w_i/m \rceil$, and hence $D'_G$, is a lower bound on $\chi(G)$. Since the size of an independent set in $G$ is at most $m$, any single color can be used only at $m$ vertices or fewer in the cycle. The total number of colors needed at all vertices being $\sum_{i=1}^{2m+1} w_i$, we conclude that $\chi(G) \geq \lceil \sum_{i=1}^{2m+1} w_i/m \rceil$. Thus, $D'_G$ is indeed a lower bound on $\chi(G)$.

Next we show that $G$ *can* be colored with $D'_G$ colors using a linear time algorithm; this completes the proof of the theorem. We first observe that there must be a *smallest* index $k$, $1 \leq k \leq m$, which satisfies the inequality

$$\sum_{i=1}^{2k+1} w_i \leq k D'_G.$$

Note that this property holds true for the index $m$ from Definition 3.1, and hence $k$ is well-defined and can be found easily in linear time.

The vertices of the cycle are now colored as follows:

1. Vertices $u_1$ through $u_{2k}$ are assigned *contiguous* colors in a cyclic manner from the palette $[1, D'_G]$. Specifically, for $1 \leq j \leq 2k$, vertex $u_j$ is assigned the colors
$$\left[ \left( 1 + \sum_{i=1}^{j-1} w_i \right), \sum_{i=1}^{j} w_i \right],$$
cyclically. By construction, this ensures that the path $u_1, u_2, \ldots, u_{2k}$ is properly multicolored since $D'_G \geq D^{[2]}_G$.
2. Vertices $u_{2k+1}$ through $u_{2m+1}$ are colored based on their parity (as in the even-cycle algorithm). In particular, for $2k+1 \leq i \leq 2m+1$, the vertex $u_i$ is assigned the colors $[1, w_i]$ if $i$ is even, or the colors $[D'_G - w_i + 1, D'_G]$ if $i$ is odd. Again, this ensures that the path $u_{2k+1}, u_{2k+2}, \ldots, u_{2m+1}$ is properly multicolored.

Since vertex $u_1$ has the colors $[1, w_1]$ and vertex $u_{2m+1}$ the colors $[D'_G - w_{2m+1} + 1, D'_G]$, the edge $(u_1, u_{2m+1})$ is also properly multicolored. All that remains is to verify that the edge $(u_{2k}, u_{2k+1})$ is properly multicolored: this is a consequence of the minimality of $k$, for we know that $\sum_{i=1}^{2k-1} w_i > (k-1) D'_G$. Hence, no color assigned to $u_{2k}$ can be among the colors $[D'_G - w_{2k+1} + 1, D'_G]$ assigned to vertex $u_{2k+1}$. □

We illustrate the labeling scheme using a 9-cycle with weight 2 on each vertex as an example. As $D'_G = \max\{4, 5\} = 5$, we use the palette $[1 \cdots 5]$. Since $\sum_{i=1}^{5} w_i = 10 \leq 2D'_G$ but $\Sigma_{i=1}^{3} w_i = 6 > D'_G$, we color the first four vertices in a cyclic manner, always taking the next four available colors in the palette. For the last five vertices, we assign colors as in a bipartite graph, from the two ends of the interval $[1, 5]$. Finally, we note that our algorithm can actually multicolor *any* cycle, and not just cycles that are hexagon graphs (i.e. embedded in the triangular lattice).

Theorem 3.2 can be used to derive an optimal multicoloring of any *outerplanar graph*. A graph is said to be outerplanar if it can be embedded in the plane so that every vertex of $G$ lies on the boundary of the exterior face. It is straightforward to see that the weighted chromatic number of any graph is the maximum taken over the weighted chromatic numbers of its biconnected components. Thus it suffices to consider a biconnected outerplanar graph: any such graph is a cycle with nonintersecting chords. A biconnected outerplanar graph $G$ without chords is a simple cycle and can be multicolored optimally using the construction in the proof of Theorem 3.2. Otherwise, let $(u, v)$ be a chord and let $G_1$ and $G_2$ be the two parts of $G$ on the sides of this chord, each one including the edge $(u, v)$. Recursively color $G_1$ and $G_2$. Relabel the color assignment of $G_2$ so that the colors assigned to $u$ and $v$ in $G_2$ agree with those assigned in $G_1$. The following corollary is immediate:

COROLLARY 3.3. *Let $G = (V, E, w)$ be an arbitrary outerplanar graph. Then $G$ can be colored optimally using $\chi(G)$ colors.*

A careful implementation of the algorithm sketched above, results in a linear time multicoloring. We remark that Corollary 3.3 applies to *any* outerplanar graph, and not just outerplanar hexagon graphs.

**4. Approximate Multicoloring of Hexagon Graphs.** In this section we consider the problem of computing an approximate multicoloring of an arbitrary hexagon graph. Since a hexagon graph may contain an odd cycle as an induced subgraph, it follows as a consequence of Theorem 3.2 that $D_G^{[3]}$, the maximum weight taken over all triangles, is not always a tight bound. For example, consider a 9-cycle where every vertex is given the weight $k$, for some integer $k \geq 2$. While $D_G^{[3]} = D_G^{[2]} = 2k$ for the graph, we know from Theorem 3.2 that $\chi(G) = \lceil 9k/4 \rceil$. We choose a 9-cycle because it is the smallest odd cycle that can be an induced subgraph of the triangular lattice. This shows that any algorithm to color hexagon graphs must use at least $\lceil 9D_G^{[3]}/8 \rceil$ colors on some graphs $G$ with triangle bound $D_G^{[3]}$. In fact, we demonstrate an efficient approximation algorithm that can multicolor *any* hexagon graph using at most $4\lceil D_G^{[3]}/3 \rceil$ colors (and hence, at most $4\lceil \chi(G)/3 \rceil$ colors).

Without loss of generality, we assume that $G$ is connected, since disconnected components of $G$ can be independently colored without any color conflicts. For simplicity, we let $M = \lceil D_G^{[3]}/3 \rceil$ and we choose the following color palette in our algorithm. Start with a *base coloring* of $G$ so that every vertex gets base color *red*, *blue* or *green*. With each base color, we associate a class of $M$ *hues* identified with the interval $[1, M]$. In addition, we have at our disposal a class of auxiliary *purple* hues, again identified with the interval $[1, M]$. The entire collection of $4M$ distinct hues forms our color palette.

The idea is to let each vertex $v$ use as many hues from its base color class as possible before trying to use hues either from the remaining two base classes or from the auxiliary purple class. We describe the algorithm as proceeding in five phases; we maintain the invariant that at the end of each phase, the graph is partially but correctly colored. To facilitate reasoning about the correctness of the algorithm, we let $G_i = (V_i, E_i, w_i)$

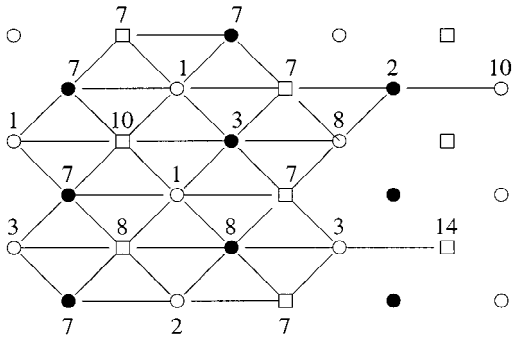


**Fig. 1.** A hexagon graph with initial weights. □, Red; ●, blue; ○, green.

denote the remaining graph after phase $i$ ($1 \leq i \leq 4$) has been completed. We also assign an artificial *priority* to vertices: red vertices dominate over blue ones which in turn dominate over green ones. This priority scheme is used in phases 2 and 3 to select, in each case, a suitable subset of vertices for partial coloring.

We illustrate our algorithm with a running example shown in Figure 1, a graph $G$ for which $3M = D_G^{[3]}$ can easily be verified to be 18. Hence, the color palette consists of 24 colors equally divided among the red, blue, green and purple hues.

A vertex $v \in G$ is defined to be *light* if $w(v) \leq M$ and to be *heavy* otherwise. This distinction is critical to each of the the five phases below:

*Phase* 1. Every vertex $v$ is assigned the first $w(v)$ hues from its base color class, in particular, the hues $[1, \min\{w(v), M\}]$. All the light vertices thus get completely colored and are deleted from the graph. The weight of every remaining heavy vertex $v$ is decreased by $M$, resulting in the graph $G_1$.

It is easy to see that $G_1$ has no maximal cliques of size greater than 2, because every triangle in $G$ must contain at least one light vertex that is eliminated in the first phase. Let $H$ denote the subgraph of $G_1$ induced by the degree 3 vertices in $G_1$. Note that if a vertex $v \in G_1$ has three neighbors (say, in clockwise order in the fixed embedding) in $G_1$, then the incident edges to the neighbors form successive angles of $2\pi/3$ radians in order; furthermore, the geometry implies that all three neighbors have the same base color. It follows that each connected component in $H$ contains vertices that belong to *at most two* base color classes. Thus, every connected component of $H$ consists of either an isolated vertex, or contains only red and blue, or red and green, or blue and green vertices.

Call a vertex $v \in H$ a *priority vertex* if and only if it has the highest priority among its neighbors (if any) in $H$ (recall that red dominates blue which dominates green). Clearly, the priority vertices form an independent set (in fact, a dominating set) in $H$.

*Phase* 2. Without loss of generality, let $v$ be a red priority vertex in $H$ with three blue neighbors in $H$. Let $g(v)$ be the *maximum* among the weights of the three green neighbors of $v$; these vertices must have been light vertices in $G$. Then $v$ can borrow from among the last $M - g(v)$ green hues; these suffice to color the remaining weight on

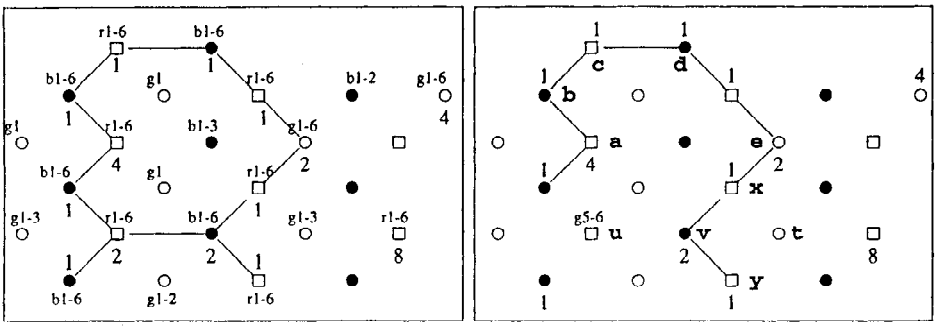


**Fig. 2.** Color assignment during (a) phase 1 and (b) phase 2.

$v$ since all three blue neighbors of $v$ are heavy vertices and, hence, $w_1(v) \leq M - g(v)$. Accordingly, $v$ is assigned the green hues, $[M - g(v) + 1, M]$, and eliminated from further consideration. Note that the partial color assignment at the end of phase 2 has no color conflicts among neighbors, and the remaining graph is designated $G_2$.

Figure 2 details the partial color assignment at the end of phases 1 and 2, respectively. Note that the six red hues are denoted as r1-6 and so forth in the figure. Since the subset of priority vertices eliminated in phase 2 is a dominating set of $H$ (the degree 3 vertices of $G_1$), every remaining vertex in $G_2$ now has degree at most 2. Equivalently, the connected components of $G_2$ consist of isolated vertices, cycles and paths in the triangular grid. Note also that any edge of $G_2$ has a residual weight of at most $M$, a consequence of the definition of heavy vertices. If the graph $G_2$ contains only even cycles or paths, then we can color all the vertices using the $M$ purple colors. However $G_2$ may contain isolated vertices and odd cycles. In the next phase we essentially eliminate *all* potential cycles in $G_2$.

Call a vertex $v \in G_2$ a *corner vertex* if and only if it has two neighbors $x, y$ of the same base color class in $G_2$ such that the angle subtended at $v$ by the incident edges $(v, x)$ and $(v, y)$ is exactly $2\pi/3$ radians. Further, a corner vertex $v$ is a *priority vertex* in $G_2$ if and only if $v$ has the highest priority among all its neighbors, if any, that are also corner vertices. It is not difficult to see that the subset of priority vertices in $G_2$ forms an independent set in $G_2$. Also, every corner vertex is either itself a priority vertex or is adjacent to a priority vertex; hence, the subset of priority vertices is a dominating set of the subgraph induced by the corner vertices in $G_2$. Finally, by definition *every* cycle in $G_2$ contains at least one corner vertex. Thus, coloring priority vertices and eliminating them also breaks all cycles. For example, in Figure 2(b), the vertices labeled $v$, $v'$ and $x$ are corner vertices of $G_2$; among them, $v$ and $v'$ are priority vertices.

*Phase* 3. Without loss of generality, let $v$ be a blue priority vertex in $G_2$ with red neighbors $x$ and $y$ in $G_2$ as shown in Figure 3. Note that $u$ denotes the third neighbor of $v$ of the same base color class as $x$ and $y$. It is also easy to see from priority considerations that $x$ and $y$ must be noncorner vertices and hence, the blue vertices $p$ and $q$ must be

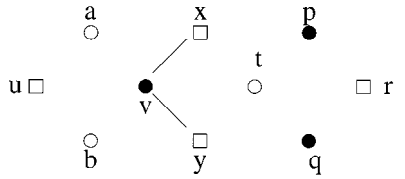


**Fig. 3.** Local geometry around a blue priority vertex $v$ in phase 3. □, Red; ●, blue; ○, green.

light vertices. While $u$ is absent in $G_2$, there are two possibilities:

(i) $u$ was eliminated in phase 1 (i.e. $w_1(u) = 0$): let $g(v)$ be the maximum over the weights of $v$'s green neighbors (i.e. vertices $a$, $b$ and $t$ in Figure 3) in $G$. Since $u$ did not participate in phase 2, $v$ can borrow from among the last $M - g(v)$ green hues; these suffice since two of $v$'s red neighbors are heavy vertices and hence $w_2(v) \leq M - g(v)$. Accordingly, $v$ is assigned the green hues, $[M - w_2(v) + 1, M]$, and eliminated.

(ii) $u$ was a priority vertex in phase 2: consider $a$ and $b$, the common (light) green neighbors of $u$ and $v$ in $G$. Let $w_{ab}$ be the maximum among $w(a)$ and $w(b)$ and recall that $u$ was assigned the *last* $w_1(u)$ green hues, $[M - w_1(u) + 1, M]$, during phase 2. Since $w_2(v) + w_1(u) + w_{ab} \leq M$, it appears that $v$ could borrow the green hues $[w_{ab} + 1, w_{ab} + w_2(v)]$ from the *middle* of the green spectrum. However, we must ensure that this assignment will not conflict with the green hues assigned to $t$ (see Figure 3).

If $w(t) \leq w_{ab}$, then clearly the new assignment does not conflict with prior color assignments. However, if $w(t) > w_{ab}$, then we can *recolor* $t$ as follows: its original assignment of the first $w(t)$ green hues (in phase 1) is changed so that $t$ now uses the first $w_{ab}$ and the last $w(t) - w_{ab}$ green hues. The new assignment to $t$ cannot conflict with the green hues, $[w_{ab} + 1, w_{ab} + w_2(v)]$, borrowed by $v$ since $w(t) + w_2(v) \leq M$.

It remains to verify that recoloring $t$ does not cause a cascading conflict with any other neighbor of $t$. Clearly such a conflict could occur only if some other neighbor of $t$ also borrows green hues in phases 2 or 3. From the observations above, we conclude that this would be impossible for vertices $x, y$ (they are not priority vertices in either phase) and $p, q$ (they are light in $G$). The only remaining possibility, namely vertex $r$, is not a problem either: $r$ could not have been a priority vertex in phase 2 (since it has three consecutive light neighbors $p, t$ and $q$) and if it were a corner vertex in phase 3, it would have green neighbors in $G_2$ and would borrow blue hues. Figure 4(a) depicts the color assignment during phase 3 for the running example; note that the light vertex labeled $t$ in Figure 2(b) is recolored as described above.

Thus, at the end of phase 3, we have a correct partial assignment of colors, and, furthermore, the remaining graph $G_3$ consists only of isolated vertices or straight-line paths. It is easy to see that any remaining isolated vertex may have a residual weight between 1 and $2M$, whereas the weight on every remaining edge is at most $M$.

*Phase* 4. To any isolated vertex $v \in G_3$, we first assign $\min\{w_3(v), M\}$ purple hues. If $w_3(v) > M$, then we still need to find $\delta = w_3(v) - M \leq M$ additional colors to finish

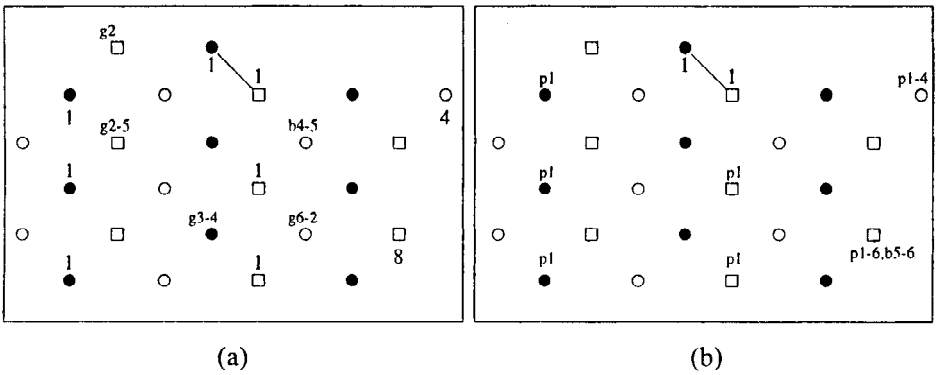


**Fig. 4.** Color assignment during (a) phase 3 and (b) phase 4.

coloring $v$. Without loss of generality, we assume that $v$ is a red vertex, and observe that all the neighbors of $v$ must have been light vertices in $G$. Hence, the blue (green) neighbors of $v$ must have had colors either assigned to them in phase 1 or in phase 3 (as a result of recoloring). We claim that $v$ can still borrow $\delta$ colors from either the blue or the green palettes without conflicting with any neighboring assignment.

From the description of phases 1 and 3, observe that the light neighbors of $v$ may be using colors from either end of their base color palettes. Let $b_3 \leq b_2 \leq b_1 \leq M$ and $g_3 \leq g_2 \leq g_1 \leq M$ be the weights of $v$'s blue and green neighbors in $G$ respectively. Clearly, regardless of the manner in which the blue neighbors are assigned blue hues, $v$ has at least $M - (b_1 + b_2)$ hues available for its use from the blue palette. Likewise, there are at least $M - (g_1 + g_2)$ green hues available to $g$. Since the green vertex with weight $g_1$ forms a triangle in $G$ with $v$ and either one of the blue vertices with weight $b_1$ or with weight $b_2$. In any event, $\delta \leq M - (b_2 + g_1)$. A similar argument shows that $\delta \leq M - (g_2 + b_1)$. It follows that

$$\delta \leq \max\{M - (b_1 + b_2), M - (g_1 + g_2)\};$$

or in other words, that $v$ can obtain the remaining $\delta$ colors by borrowing either only blue hues or only green hues without any color conflicts with assignments prior to the phase.

Figure 4(b) demonstrates the colors assigned in phase 4 to the remaining isolated vertices in our running example. At the end of phase 4, the remaining graph consists only of *straight-line* paths, i.e. paths in which any two consecutive edges subtend an angle of $\pi$ degrees at the common vertex. Further, as noted above, every remaining edge has a residual weight of at most $M$.

*Phase* 5. Since every remaining connected component is a straight-line path with a weighted chromatic number of at most $M$, it suffices to use the greedy parity-based strategy described in Section 3 to finish coloring the graph using the $M$ purple hues. This cannot cause any conflict with previously colored vertices since the purple hues were used only in phase 4 to color isolated vertices in $P$; the latter are disconnected from any remaining vertex in the current phase.

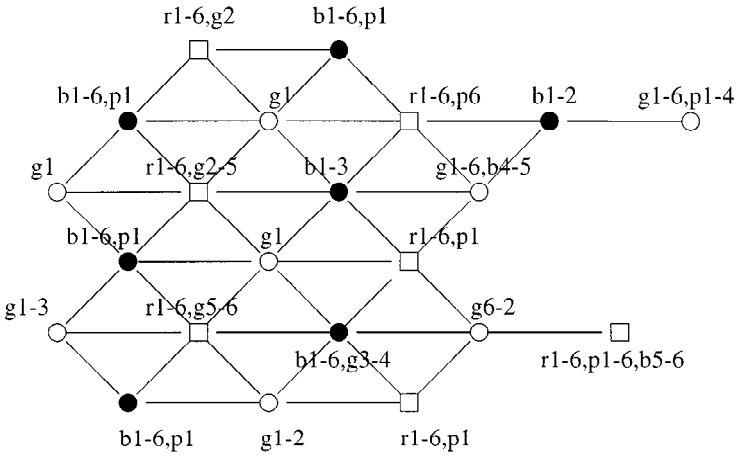


**Fig. 5.** Complete color assignment in the example graph.

Figure 5 shows the entire color assignment constructed by our algorithm for the running example. The following result is immediate:

THEOREM 4.1. *An approximate multicoloring that uses no more than* $4\lceil D_G^{[3]}/3\rceil$ *colors for any hexagon graph* $G = (V, E, w)$ *can be efficiently computed in linear time.*

**5. A Distributed Implementation.** The algorithm given in the previous section has the additional property that it can be implemented in a completely distributed manner. We consider the hexagon graph as modeling a network of processors (base stations), with each processor responsible for a single vertex in the hexagon graph. The network has the same spatial embedding as the graph, and processors at neighboring base stations in the network can exchange local information efficiently. For ease of description, we sometimes identify the vertices of the hexagon graph with their processors.

In the fixed planar embedding of the infinite triangular grid, we can select an arbitrary vertex to be the origin, and three *directional axes* that intersect the origin: one designated the horizontal axis, and the remaining two at angles $\pi/3$ and $2\pi/3$ from the horizontal axis. It is easy to see that any path in the graph, where the angles subtended by all intermediate edges in the path are exactly $\pi$, is *oriented along* one of the three directional axes. For every vertex, we wish to assign a *parity with respect to each directional axis*. This can be easily done as follows. The parity of a vertex $v$ along the horizontal axis is defined to be the parity of the length of the path oriented along the horizontal axis from $v$ to a vertex on the $\pi/3$ axis that intersects the origin. Similarly, the parity of $v$ along the $\pi/3$ axis ($2\pi/3$ axis) is the parity of the length of the path from $v$ oriented along the $\pi/3$ axis ($2\pi/3$ axis) to a vertex on the horizontal axis intersecting the origin. Thus given an arbitrary finite hexagon graph, any path in the graph that is oriented along one of the directional axes has a 2-coloring that can be precomputed according to the parities of the vertices along the path.

Our algorithm assumes that each processor initially has access to the following information:

- A base coloring for the graph is known: each processor knows whether its vertex is red, green or blue, and also the color of each of its neighbors.
- A 2-coloring along each path oriented along a directional axis is known. This means each processor knows three bits corresponding to whether it has even or odd parity along each of the three directional axes.
- The value of $D_G^{[3]}$ is known; this also implies that the division of base color classes among processors is known. Additionally, this implies that every vertex has access to a known set of $M$ purple hues if needed.

The distributed algorithm consists of each processor determining whether it should participate in one or more of the five phases of the approximation algorithm described in Section 4. The algorithm starts with three rounds of information gathering, after which no more communication other than informing neighbors of the current color assignment is required; essentially, processors can continue independently to compute the hues to assign to themselves. We describe the communication rounds from the perspective of a fixed processor $p$.

*Round* 1. $p$ sends its weight to each of its six neighbors.

*Round* 2. Having received the weights of all its neighbors, $p$ decides if it would be a degree 3 vertex after phase 1, and sends this information (a single bit) to each of its neighbors. This would be the case if $p$ is itself a heavy vertex and has three neighbors that are heavy vertices.

*Round* 3. The information received in the previous two rounds suffices for $p$ to decide if it will be a priority vertex in phase 2 as well as if any of its neighbors will be priority vertices in phase 2. For instance, $p$ will be a priority vertex if it is a blue vertex with degree 3 after phase 1 either with no neighbors that will also be degree 3 vertices after phase 1, or with green neighbors of degree 3 after phase 1 (see Section 4).

Next, $p$ determines if it will be a corner vertex in phase 4 and sends this information (a single bit) to each of its neighbors. $p$ is a corner vertex if all the following conditions are met:

- $p$ is a heavy vertex.
- $p$ will not be a priority vertex in phase 2.
- $p$ has exactly two neighbors of the *same* color class that are heavy vertices but not priority vertices in phase 2.

*Round* 4. The information derived from round 3 enables $p$ to determine if it will be a priority vertex in phase 3, as described in Section 4. If $p$ would be a priority vertex in phase 3, and would fall into case (ii) of phase 3, then it has a light neighbor, say $q$ that would need to be recolored in that phase. In this case, $p$ sends a message to $q$ with the maximum weight of its remaining two neighbors of the same color as $q$, so that $p$ can color itself appropriately.

*Round* 5. A light vertex that got a message to recolor itself in round 4 informs all its neighbors of how many colors it will use from the end of its base color

spectrum. This enables an isolated vertex in its neighborhood to color itself appropriately in phase 4.

From the weights of its neighboring processors and its limited global knowledge, and the information collected in the communication rounds described above, a processor can easily compute the colors it will use in each of the five phases. Without loss of generality, consider a processor that corresponds to a blue vertex $v \in G$. The processor emulates the five phases of the sequential algorithm as follows:

*Phase* 1. If $w(v) \leq M$, the processor assigns (to itself) the appropriate blue hues. Otherwise, it assigns to itself all the blue hues, reduces its weight by $M$ and continues.

*Phase* 2. If the processor is a priority vertex in phase 2, it simulates phase 2 and stops, or else continues to phase 3. Recall that the colors that a priority vertex would borrow from one of its neighboring color class are the last colors from that class; thus no consultation is required with neighbors to compute the colors at a priority vertex.

*Phase* 3. If the processor is a priority vertex in phase 3, it can determine the colors it needs to borrow from the appropriate neighboring color class.

If, however, the processor is a light vertex that would have undergone recoloring in phase 3 of the sequential algorithm, then it can emulate this behavior in the distributed algorithm as well. To do so, it uses the information it received in its neighbor in round 4, and recolors itself as described in Section 4 so that no conflict appears among the blue colors.

*Phase* 4. If a processor is an isolated vertex in this phase, it assigns itself any purple hues that it needs. Additional colors that it may need are borrowed from one of the neighboring color classes. Since any of its light neighbors that may have recolored itself in the last phase sent information in round 5 about the number of colors it would use from the end of its base color spectrum, the processor can determine all colors used by its light neighbors and can borrow colors without any possibility of conflict.

*Phase* 5. Any vertex with unassigned colors at this stage lies along some straight-line path in the grid. In particular, it can detect its one or two incompletely assigned neighbors that lie along exactly one of the three directional axes. The processor can easily compute the identity of the particular axis from the information gathered after round 3. Depending on whether it is an even or odd vertex along this axis, it assigns itself the necessary hues from the beginning or end of the set of purple hues, as in the coloring for bipartite graphs.

THEOREM 5.1. *An approximate multicoloring that uses no more than* $4\lceil D_G^{[3]}/3\rceil$ *colors for any hexagon graph* $G = (V, E, w)$*, can be efficiently computed in constant time in a completely distributed manner, after an initial constant time communication protocol where each processor exchanges five messages with each of its neighbors.*

**6. Discussion.** In this paper we have cast the problem of frequency assignment in cellular networks as a multicoloring problem for hexagon graphs. For some particular induced subgraphs of hexagon graphs, i.e. cycles and outerplanar graphs, we show efficient algorithms for multicoloring them using an optimal number of colors. In Section 4

we describe a multicoloring algorithm that uses at most $4\lceil D_G^{[3]}/3\rceil$ colors where $D_G^{[3]}$, the maximum weight on any 3-clique in $G$, is a trivial lower bound on the minimum number of colors required. We showed also a hexagon graph that requires $\lceil 9D_G^{[3]}/8\rceil$ colors. Determining an exact bound on $\chi(G)$, the weighted chromatic number of an arbitrary hexagon graph is NP-hard; our results do establish that for all hexagon graphs $G$, $\chi(G) \leq 4\lceil D_G^{[3]}/3\rceil$. Whether or not there is an approximation algorithm for hexagon graphs which always uses at most $\lceil 9D_G^{[3]}/8\rceil$ colors remains an intriguing open problem. A useful feature of our algorithm is that it can be implemented in a distributed manner. In contrast, the algorithm of McDiarmid and Reed [7], which has the same performance ratio as ours, seems inherently centralized and cannot be made distributed in any obvious way.

An interesting avenue for future research is the generalized version of the problem, where the frequencies assigned at a particular vertex or at adjacent vertices are required not merely to be different, but also to be far enough apart [3]. Another recently proposed model [1] considers arbitrary interference graphs with predefined costs on the edges; these costs reflect the interference penalties when the same or adjacent frequencies are assigned to neighboring nodes. The objective here is to minimize the total cost, or, equivalently, the net interference of an assignment. Finally, the dynamic version of the problem involves changing weights at vertices. It would be interesting to see if the distributed algorithm we describe in Section 5 can be adapted to work in this setting and what bounds can be proved on its performance.

**Acknowledgments.** We thank Jeannette Janssen for introducing us to the problem of channel assignment, and for comments that greatly improved the presentation of Section 3. We are grateful to the anonymous referees for their useful comments, and for suggesting the current form of Corollary 3.3.

## References

[1] R. Borndörfer, A. Eisenblätter, M. Grötschel, and A. Martin. Frequency Assignment in Cellular Phone Networks. Technical Report, Knorad-Zuse Zentrum für Informationstechnik, Berlin, 1997.

[2] D. Dimitrijević and J. Vučetić. Design and performance analysis of algorithms for channel allocation in cellular networks. *IEEE Transactions on Vehicular Technology*, 42(4):526–534, 1993.

[3] W. K. Hale. Frequency assignment: theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

[4] J. Janssen, K. Kilakos, and O. Marcotte. Fixed preference frequency allocation for cellular telephone systems. Unpublished manuscript, April 1995.

[5] 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.

[6] S. Kim and S. L. Kim. A two-phase algorithm for frequency assignment in cellular mobile systems. *IEEE Transactions on Vehicular Technology*, 1994.

[7] C. McDiarmid and B. Reed. Channel assignment and weighted coloring. Submitted for publication, 1997.

[8] P. Raymond. Performance analysis of cellular networks. *IEEE Transactions on Communications*, 39(12):1787–1793, 1991.

[9] W. Wang and C. Rushforth. An Adaptive Local-Search Algorithm for the Channel-Assignment Problem. Technical Report, August 1995.