1-local 17/12-competitive Algorithm for Multicoloring Hexagonal Graphs*

Rafał Witkowski

Adam Mickiewicz University,
Faculty of Mathematics and Computer Science,
Poznań, Poland
rmiw@amu.edu.pl

Abstract. In the frequency allocation problem we are given a cellular telephone network whose geographical coverage area is divided into cells where phone calls are serviced by frequencies assigned to them, so that none of the pairs of calls emanating from the same or neighboring cells is assigned the same frequency. The problem is to use the frequencies efficiently, i.e. minimize the span of used frequencies. The frequency allocation problem can be regarded as a multicoloring problem on a weighted hexagonal graph. In this paper we present a 1-local 17/12-competitive distributed algorithm for a multicoloring of hexagonal graph, thereby improving the competitiveness ratio of previously known best 1-local 13/9-competitive algorithm (see [1]).

1 Introduction

The basic problem concerning cellular networks concentrates on assigning sets of frequencies (colors) to transmitters (vertices) in order to avoid unacceptable interference (see [7]). In an ordinary cellular model the transmitters are centers of hexagonal cells and the corresponding adjacency graph is a subgraph of the infinite triangular lattice. In our model to each vertex v of a the triangular lattice T we assign a non-negative integer $d(v)$, called the *demand* (or *weight*) of the vertex v. A *proper multicoloring* of G is a mapping φ from $V(G)$ to subsets of integers (colors) $[n] = \{1, 2, \ldots, n\}$, such that $|\varphi(v)| = d(v)$ for any vertex $v \in G$ and $\varphi(v) \cap \varphi(u) = \emptyset$ for any pair of adjacent vertices u and v in the graph G. The minimal n for which there exists a proper multicoloring of G, denoted by $\chi_m(G)$, is called the *multichromatic number* of G. A *hexagonal graph* $G = (V, E, d)$ is the vertex weighted subgraph of T, induced by the set of its vertices with positive demands (the idea of hexagonal graphs arise naturally in studies concerning cellular networks). The multichromatic number is closely related to the *weighted clique number* $\omega(G)$, which is defined as the maximum over all cliques of G of their weights, where the weight of a clique is the sum of demands on its vertices. Obviously, for any graph, $\chi_m(G) \geq \omega(G)$, while for hexagonal graphs (see, for example, [2], [3], [6]), $\chi_m(G) \leq \left\lceil \frac{4\omega(G)}{3} \right\rceil + O(1)$. Since all proofs of the upper

* This work was supported by grant N206 017 32/2452 for years 2007-2010

bound are constructive, therefore it implies the existence of a *4/3-competitive algorithm*, i.e. algorithms which can online serve calls with the approximation ratio equal to 4/3 respectively to the weighted clique number (see [5], [8]). It should be also mentioned that McDiarmid and Reed showed in [3] that to decide whether $\chi_m(G) = \omega(G)$ is NP-complete.

In distributed graph algorithms a special role plays their "locality" property. An algorithm is *k-local* if the computation at any vertex v uses only the information about the demands of the vertices at distance at most k from v. For hexagonal graphs the best previously known 1-local algorithm for multicoloring is 13/9-competitive, and it has been presented in [1]. In this paper we develop a new 1-local algorithm which uses no more than $\lceil \frac{17}{12}\omega(G) \rceil + O(1)$ colors, thus improving the result from [1]. Our algorithm substantially differs from previous ones. Those algorithms (e.g. [1], [2]) are composed of two stages. At the first stage, a triangle-free hexagonal graph with weighted clique number no larger than $\lceil \omega(G)/3 \rceil$ is constructed from G, while at the second stage an algorithm for multicoloring a triangle-free hexagonal graph is used (see [1], [10], [11]). Our algorithm skips the second stage entirely.

Theorem 1. *There is a 1-local distributed approximation algorithm for multicoloring hexagonal graphs which uses at most* $\lceil \frac{17}{12}\omega(G) \rceil + O(1)$ *colors. Time complexity of the algorithm at each vertex is constant.*

In [8] it was proved that a k-local c-approximate offline algorithm can be easily converted to a k-local c-competitive online algorithm. Hence,

Corollary 1. *There is a 1-local 17/12-competitive algorithm for multicoloring hexagonal graphs.*

In the next Section we formally define some basic terminology, while in Section 3 we present the algorithm and prove Theorem 1.

2 Basic definition and useful facts

Following the notation from [3], the vertices of the triangular lattice T can be described as follows: the position of each vertex is an integer linear combination $x\boldsymbol{p}+y\boldsymbol{q}$ of two vectors $\boldsymbol{p} = (1, 0)$ and $\boldsymbol{q} = (\frac{1}{2}, \frac{\sqrt{3}}{2})$. Thus vertices of the triangular lattice may be identified with pairs (x, y) of integers. Two vertices are adjacent when the Euclidean distance between them is one. Therefore each vertex (x, y) has six neighbors: $(x-1, y), (x-1, y+1), (x, y+1), (x+1, y), (x+1, y-1), (x, y-1)$. For simplicity we refer to the neighbors as: *left*, *up-left*, *up-right*, *right*, *down-right* and *down-left*. We define a *hexagonal graph* $G = (V, E)$ as an induced subgraph of the triangular lattice (see Figure 1).

There exists an obvious 3-coloring of the infinite triangular lattice which gives partition of the vertex set of any hexagonal graph into three independent sets. Let us denote a color of any vertex v in this 3-coloring by $bc(v)$ and call it a *base color* (for simplicity we will use *red*, *green* and *blue* as base colors and their arrangement is given in Figure 1), i.e. $bc(v) \in \{R, G, B\}$.

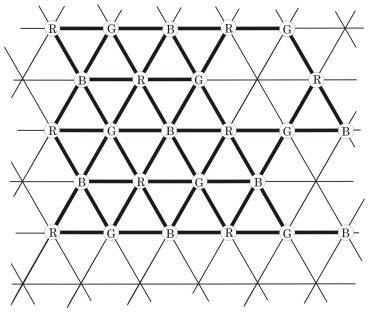

Fig. 1. An example of a hexagonal graph

We call a *triangle-free hexagonal graph* an induced subgraph of the triangular lattice without 3-clique. We define a *corner* in a triangle-free hexagonal graph as a vertex which has at least two neighbors and none of them are at angle π. A vertex is a *right corner* if it has an up-right or a down-right neighbor, and otherwise it is a *left corner* (see Figure 2). A vertex which is not a corner is called a *non-corner*.

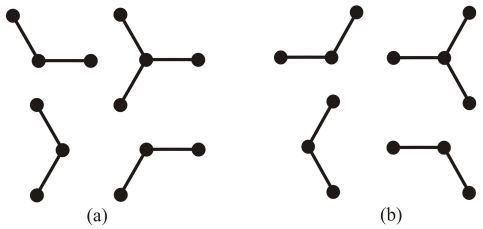

Fig. 2. All possibilities for: (a) - left corners, (b) - right corners

In graph $G = (V, E)$, we call a coloring $f : V \to \{1, \ldots, k\}$ *k-good* if for every odd cycle in G and for every i, $1 \leq i \leq k$, there is a vertex $v \in V$ in the cycle such that $f(v) = i$. A *graph is k-good* if such coloring exists.

Lemma 1. *[4] Consider a 3-coloring of the triangular lattice (R,G,B). Every odd cycle of the triangle-free hexagonal graph G contains at least one non-corner vertex of every color.*

Proof. Assume without loss of generality that there exists an odd cycle in the graph which does not have a non-corner vertex colored red. Notice that in the 3-coloring of the triangular lattice, a corner has all its neighbors colored by the same color (they are at the angle $2\pi/3$ since the graph is triangle-free). Hence, if all neighbors of a red colored corner are blue, we can recolor this corner

by green color and vice-versa. That gives a valid 2-coloring of an odd cycle, a contradiction. □

Note that two successive corners in any cycle cannot be both left (right). By Lemma 1 and since every cycle has at least one left and one right corner, we get the following observation:

Proposition 1. *Any triangle-free hexagonal graph G is 5-good.*

One can also give an explicit 5-good coloring of every triangle-free hexagonal graph by assigning colors in the following way:

- PINK – to non-corner vertices with base color equal to red,
- LIME – to non-corner vertices with base color equal to green,
- AQUA – to non-corner vertices with base color equal to blue,
- WHITE – to left corner vertices,
- YELLOW – to right corner vertices.

We denote color of a vertex v in this 5-good coloring by $ec(v)$ and call it an *extra color* of v (for simplicity we will use *pink*, *lime*, *aqua*, *white* and *yellow* as extra colors, see Figure 3) , i.e. $ec(v) \in \{P, L, A, W, Y\}$.

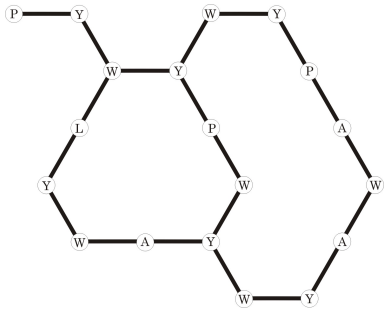

Fig. 3. An example of a triangle-free hexagonal graph with 5-good coloring

Notice that if a graph G is 5-good then after removing vertices colored by any of those five colors, the resulting graph is bipartite. For any weighted bipartite graph H, $\chi_m(H) = \omega(H)$ (see [6]), and it can be optimally multicolored by the following procedure.

Procedure 2 *Let $H = (V', V'', E, d)$ be a weighted bipartite graph. We get an optimal multicoloring of H if to each vertex $v \in V'$ we assign a set of colors $\{1, 2, \ldots, d(v)\}$, while with each vertex $v \in V''$ we associate a set of colors $\{m(v) + 1, m(v) + 2, \ldots, m(v) + d(v)\}$, where $m(v) = \max\{d(u) : \{u, v\} \in E\}$.*

Notice that in any weighted hexagonal graph G, a subgraph of the triangular lattice T induced by vertices with positive demands $d(v)$, the only cliques are triangles, edges and isolated vertices. Note also that we assume that all vertices of T which are not in G have to have demand $d(v) = 0$. Therefore, the weighted clique number of G can be computed as follows:

$$\omega(G) = \max\{d(u) + d(v) + d(t) : \{u, v, t\} \in \tau(T)\},$$

where $\tau(T)$ is the set of all triangles of T.

For each vertex $v \in G$, define *base function* κ as

$$\kappa(v) = \max\{a(v, u, t) : \{v, u, t\} \in \tau(T)\},$$

where $a(u, v, t) = \lceil (d(u) + d(v) + d(t))/3 \rceil$, is an average weight of the triangle $\{u, v, t\} \in \tau(T)$. It is easy to observe that the following fact holds.

Proposition 2. *For each $v \in G$,*

$$\kappa(v) \leq \left\lceil \frac{\omega(G)}{3} \right\rceil$$

We call vertex v *heavy* if $d(v) > \kappa(v)$, otherwise we call it *light*. If $d(v) > 2\kappa(v)$ we call vertex *very heavy*.

To color vertices of G we use colors from appropriate *palette*. For a given color c, its palette is defined as a set of pairs $\{(c, i)\}_{i \in \mathbb{N}}$. A palette is called *base color palette* if $c \in \{R, G, B\}$, while it is called *extra color palette* if $c \in \{P, L, A, W, Y\}$.

In our model of computations we assume that each vertex knows its coordinates as well as its own demand (weight) and demands of all it neighbors. With this knowledge, each vertex has to color itself properly in constant time in a distributed way.

3 Algorithm and its correctness

Our algorithm consists of two main phases. In the first phase vertices take $\kappa(v)$ colors from its base color palette, so use no more than $\omega(G)$ colors. After this phase all light vertices are fully colored while the remaining vertices create a triangle-free hexagonal graph with weighted clique number not exceeding $\lceil \omega(G)/3 \rceil$ (after technical removing very heavy vertices). In the second phase we construct 5-good coloring of the remaining graph. Recall that in 5-good coloring, a graph is bipartite after removing vertices of any of these five colors. If we use Procedure 2 and color such graphs optimally with weight function equal in each vertex to 1/4 of its demands, then we would fully color the remaining graph and use no more than 5/4 colors than it is needed. Due to the proof of Lemma 1, bipartition is easy to find after removing any class of non-corners (pink, aqua or lime vertices). Unfortunately we cannot obtain this bipartition in our 1-local model of computation when we remove any class of corners (white or yellow vertices). We can do it only for non-corners, while corners have to be satisfied in a separate way – by using free colors from base color palettes.

More precisely, our algorithm consists of the following steps:

Algorithm

Step 0 For each vertex $v = (x, y) \in V$ compute its base color $bc(v)$

$$bc(v) = \begin{cases} R & \text{if} \quad x + 2y \bmod 3 = 0 \\ G & \text{if} \quad x + 2y \bmod 3 = 1 \\ B & \text{if} \quad x + 2y \bmod 3 = 2 \end{cases},$$

and its base function value

$$\kappa(v) = \max\left\{ \left\lceil \frac{d(u) + d(v) + d(t)}{3} \right\rceil : \{v, u, t\} \in \tau(T) \right\}.$$

Step 1 For each vertex $v \in V$ assign to v $\min\{\kappa(v), d(v)\}$ colors from its base color palette. Construct a new weighted triangle-free hexagonal graph $G_1 = (V_1, E_1, d_1)$ where $d_1(v) = \max\{d(v) - \kappa(v), 0\}$, $V_1 \subseteq V$ is the set of vertices with $d_1(v) > 0$ (heavy vertices) and $E_1 \subseteq E$ is the set of all edges in G with both endpoints from V_1 (E_1 is induced by V_1).

Step 2 For each vertex $v \in V_1$ with $d_1(v) > \kappa(v)$ (very heavy vertices) assign free colors from the first $\kappa(v)$ of base color palettes of its neighbors in T. Construct a new graph $G_2 = (V_2, E_2, d_2)$ where d_2 is the difference between $d_1(v)$ and the number of assigned colors in this Step, $V_2 \subseteq V_1$ is the set of vertices with $d_2(v) > 0$ and $E_2 \subseteq E_1$ is the set of all edges in G_1 with both endpoints from V_2 (E_2 is induced by V_2).

Step 3 Determine 5-good coloring of G_2: for each vertex $v \in V_2$ compute its extra color $ec(v)$

$$ec(v) = \begin{cases} P & \text{if } v \text{ is non-corner in } G_2 \text{ and } bc(v) = R \\ L & \text{if } v \text{ is non-corner in } G_2 \text{ and } bc(v) = G \\ A & \text{if } v \text{ is non-corner in } G_2 \text{ and } bc(v) = B \\ W & \text{if } v \text{ is left corner in } G_2 \\ Y & \text{if } v \text{ is right corner in } G_2 \end{cases}$$

Step 4 For each class of non-corners (pink, lime, aqua) do as follows: remove from G_2 all pink (lime, aqua) vertices and based on the proof of Lemma 1 find a bipartition of the remaining graph. Apply Procedure 2 to satisfy 1/4 demands in G_2 by colors from pink (lime, aqua) extra color palette.

Step 5 For each class of corners (white, yellow) do as follows: remove from G_2 all white (yellow) vertices and:

- **5a** find a bipartition of non-corners using their positions in the triangular lattice T and apply Procedure 2 to satisfy 1/4 demands in G_2 by colors from white (yellow) extra color palette.
- **5b** for each corner satisfy 1/4 its demands in G_2 by the free colors of first $\kappa(v)$ from base color palettes of its light neighbors in T.

Correctness proof

At the very beginning of the algorithm there is a 1-local communication when each vertex finds out about the demands of all its neighbors. From now on, no more communication will be needed. Recall that each vertex knows its position (x, y) on the triangular lattice T.

In Step 0 there is nothing to prove.

In Step 1 each heavy vertex v assigns $\kappa(v)$ colors from its base color palette, while each light vertex u assigns $d(u)$ colors from its base color palette. Note that G_1 consists only of heavy vertices, therefore G_1 is a triangle-free hexagonal graph. For any $\{v, u, t\} \in \tau(G)$, since $3 \min\{\kappa(v), \kappa(u), \kappa(t)\} \geq d(v) + d(u) + d(t)$ and $\min\{\kappa(v), \kappa(u), \kappa(t)\} \geq \min\{d(v), d(u), d(t)\}$, at most two of $d_1(v), d_1(u), d_1(t)$ are strictly positive and at least one of the vertices u, v and t has all its required colors totally assigned in Step 1. Therefore, the graph G_1 does not contain 3-clique, i.e. it is a triangle-free hexagonal graph. The remaining weight of each vertex $v \in G_1$ is

$$d_1(v) = d(v) - \kappa(v).$$

In Step 2 only vertices with $d_1(v) > \kappa(v)$ (very heavy vertices) are colored. If vertex v is very heavy in G then it is isolated in G_1 (all its neighbors are light in G). Otherwise, for some $\{v, u, t\} \in \tau(T)$ we would have

$$d(v) + d(u) > 2\kappa(v) + \kappa(u) \geq 3a(v, u, t) \geq d(v) + d(u),$$

a contradiction. Without loss of generality we may assume that $bc(v) = R$. Denote by

$$D_G(v) = \min\{\kappa(v) - d(u) : \{u, v\} \in T, bc(u) = G\},$$

$$D_B(v) = \min\{\kappa(v) - d(u) : \{u, v\} \in T, bc(u) = B\}.$$

Obviously, $D_G(v), D_B(v) > 0$ for very heavy vertices $v \in G_1$. Since in Step 1 each light vertex t uses exactly $d(t)$ colors from its base color palette, we have at least $D_G(v)$ free colors from the green base color palette and at least $D_B(v)$ free colors from the blue base color palette, so that vertex v can assign those colors to itself. Then, we would have G_2 with $\omega(G_2) \leq \lceil \omega(G)/3 \rceil$. To prove it, we will need the following lemma:

Lemma 2. *In G_1 for every edge $\{v, u\} \in E_1$ holds:*

$$d_1(v) + d_1(u) \leq \kappa(v), \quad d_1(u) + d_1(v) \leq \kappa(u).$$

Proof. Assume that v and u are heavy vertices in G and $d_1(v) + d_1(u) > \kappa(v)$. Then for some $\{v, u, t\} \in \tau(T)$ we have:

$$d(v)+d(u) = d_1(v)+\kappa(v)+d_1(u)+\kappa(u) > 2\kappa(v)+\kappa(u) \geq 3a(u, v, t) \geq d(u)+d(v),$$

again a contradiction. □

Proposition 3.

$$\omega(G_2) \leq \lceil \omega(G)/3 \rceil.$$

Proof. Recall that in a hexagonal graph the only cliques are triangles, edges and isolated vertices. Since G_1 is a triangle-free hexagonal graph, G_2 also does not contain any triangle, so we have only edges and isolated vertices to check.

For each edge $\{v, u\} \in E_2$ from Lemma 2 and Proposition 2 we have:

$$d_2(v) + d_2(u) \leq d_1(v) + d_1(u) \leq \kappa(v) \leq \lceil \omega(G)/3 \rceil.$$

For each isolated vertex $v \in G_2$ we should have $d_2(v) \leq \lceil \omega(G)/3 \rceil$. Indeed, if $d_2(v) \leq \kappa(v)$, then it holds by Proposition 2. If $d_2(v) > \kappa(v)$, then $d_1(v) > \kappa(v)$, so v has to borrow colors from its neighbors' base color palettes in Step 2. Then, for $bc(v) = R$,

$$d_2(v) = d_1(v) - D_G(v) - D_B(v) \leq d(v) - \kappa(v) - \kappa(v) + d(u) - \kappa(v) + d(t) \leq$$

$$\leq 3a(v, u, t) - 3\kappa(v) \leq 0$$

for some $\{v, u, t\} \in \tau(T)$. Hence, $d_2(v) \leq \lceil \omega(G)/3 \rceil$, and so $\omega(G_2) \leq \lceil \omega(G)/3 \rceil$. □

In Step 3 each vertex v has to decide whether it is a corner in G_2 or not. Only heavy neighbors of v can still exist in G_2. Unfortunately, in 1-local model v does not know which of his neighbors are heavy (and still exist in G_2) and which are light. Vertex v knows only where its neighbors with $d(u) \leq \max\{a(v, u, t) : \{v, u, t\} \in \tau(T)\}$ are located. We call those vertices *slight neighbors* of v. They must be light and, so, they are fully colored in Step 1. Thus, v knows where it cannot have neighbors in G_2 and presumes that all its neighbors which are not slight, still exist in G_2. Based on that knowledge, it can decide whether it is a corner or not. In each triangle in $\tau(T)$ containing v at least one neighbor of v is slight, so v has at least three such neighbors. If vertex v has more than four slight neighbors, then it is a non-corner. If vertex v has four slight neighbors, then the remaining two are not slight. In this case if an angle between those two are π, then v is non-corner, otherwise it is a corner – a right corner if its down-left, up-left and right neighbors are slight, and a left corner if its down-right, up-right and left neighbors are slight. If vertex v has three slight neighbors, then it is a corner and distinction between left and right is determined in the same way as above.

Step 4 strictly depends on a 5-good coloring of graph G_2 (function ec). For simplicity, consider only graph $G_P = (V_P, E_P, \lceil d_2/4 \rceil)$ where V_P is obtained from V_2 by removing pink vertices, $E_P \subseteq E_2$ is the set of edges of G_2 with both endpoints in V_P and weight function is 1/4 of weight function in G_2. (Similarly we can define G_L for lime vertices and G_A for aqua vertices and the analysis is identical.) Since for G_P we remove pink vertices from G_2, therefore by Lemma 1, graph G_P is bipartite. We can easily find bipartition of this graph using base colors (function bc): we put to the first set of the bipartition all non-corners

with base color equal to blue and red corners for which all neighbors in G_2 are green; while to the second set we put all non-corners with base color equal to green and red corners for which all neighbors in G_2 are blue. Next, we can apply Procedure 2 to G_P with bipartition defined above and weight function on each vertex v equal to $\lceil d_2(v)/4 \rceil$, assigning colors from the pink extra color palette. The problem is that, under 1-locality assumption, vertices cannot calculate value of d_2 of the neighbors, which is needed in Procedure 2 to calculate value $m(v) = \max\{\lceil d_2(u)/4 \rceil : \{u, v\} \in E_2\}$. However, we can replace $d_2(u)$ by $d_2^v(u)$, which is the number of expected demands on vertex u in vertex v after Step 2, and take $m'(v) = \max\{\lceil d_2^v(u)/4 \rceil : \{u, v\} \in E_2\}$. More precisely,

$$d_2^v(u) = d(u) - \max\{a(u, v, t) : \{u, v, t\} \in \tau(T)\}$$

Note that $d_2^v(u) \geq d_2(u)$ for any $\{u, v\} \in E_2$. However, for every $\{v, u\} \in E_2$ we have

$$d_2(v) + d_2^v(u) \leq \kappa(v).$$

Assume that this inequality does not hold. Denote by

$$b(u, v) = \max\{a(u, v, t) : \{u, v, t\} \in \tau(T)\}.$$

Then for some $\{t, v, u\} \in \tau(T)$ we have:

$$d(v) + d(u) = d_2(v) + \kappa(v) + d_2^v(u) + b(u, v) > 2\kappa(v) + b(u, v) \geq$$

$$\geq 3a(u, v, t) \geq d(u) + d(v),$$

a contradiction. Hence, if we use d_2^v instead of d_2 in each vertex from the second set of our bipartition, we have new $\omega(G_2)$ and inequality from Proposition 3 still holds. Thus, Procedure 2 works and uses at most $\lceil \omega(G_2)/4 \rceil$ colors in G_P.

In Steps 5 and 5a we proceed almost in the same way. For simplicity, consider only graph $G_W = (V_W, E_W, \lceil d_2/4 \rceil)$ where V_W is obtained from V_2 by removing white vertices, $E_W \subseteq E_2$ is the set of edges from G_2 with both endpoints in V_W and the weight function is 1/4 of the weight function in G_2. (Similarly we can define G_Y for yellow vertices and the analysis is identical.) Since we take G_2 without white vertices, in G_W there are not any left corners and we have only right corners and non-corners. In 5a we apply Procedure 2 to non-corners and assign colors from the white extra color palette. We find a bipartition using *parity function* p on each vertex. Parity is a function which calculates whether $v = (x, y)$ is "even" or "odd" vertex on the line to which it belongs. Formally:

- if v has up-left, up-right, down-left, down-right slight neighbors then

$$p(v) = x \bmod 2$$

- if v has left, up-left, right, down-right slight neighbors then

$$p(v) = y \bmod 2$$

– if v has left, down-left, right, up-right slight neighbors then

$$p(v) = y \bmod 2$$

In Step 5b for G_W we take right corners and go back for a while to the base color palettes. If vertex $v \in G_2$ is a corner, it means that it has three slight neighbors with the same base color. Without loss of generality, assume that $bc(v) = R$ and its slight neighbors' base color is blue. Recall function D_B from Step 2 – we have $D_B(v)$ free colors from blue base color palette. We should have $d_2(v) \leq D_B(v)$. Let $\Delta = \{u, v, t\} \in \tau(T)$ be a triangle such that t is the green vertex which is not slight neighbor of v, and u is the blue vertex which is a slight neighbor of v. Denote by $s_\Delta(t) = d(t) - a(u, v, t)$. Then we have

$$0 \geq d(v) - a(u, v, t) + d(t) - a(u, v, t) + d(u) - a(u, v, t) \geq$$

$$\geq d(v) - \kappa(v) + s_\Delta(t) + d(u) - \kappa(v) \geq d_1(v) + s_\Delta(t) - D_B(v) \geq$$

$$\geq d_2(v) + s_\Delta(t) - D_B(v)$$

Since t is not a slight neighbor of v, $d_2(v) < D_B(v)$. Therefore, vertex v has as much as $d_2(v)$ free colors from the blue base color palette at his disposal, while it needs just $\lceil d_2(v)/4 \rceil$.

In both rounds of Step 5b (white and yellow vertices) we do the same for every right and left corners. We have to be careful not to cause a conflict when some right and left corners are adjacent in G_2. Then we cannot use the same color from the base colors palette from common slight neighbors. To ensure that, for left corners we can take only "even colors" and for right corners only "odd colors" from the base color palettes (recall that we think of colors in a palette as integers). We can do this because we have four times more free colors than we need in each corner, and two times more than may be needed for any two adjacent corners.

During Steps 4 and 5 each vertex v participates in exactly four from five rounds (in each round one extra color is removed from G_2) and $\lceil d_2(v)/4 \rceil$ colors are assigned in each. Therefore, at the end, all demands are satisfied.

Ratio

We claim that during the first phase (Steps 1 and 2) our algorithm uses at most $\omega(G) + 3$ colors. To see this notice that in Step 1 each vertex v uses at most $\kappa(v)$ colors from its base color palette and, by Proposition 2 and the fact that there are three base colors, we know that no more than $3\lceil \omega(G)/3 \rceil \leq \omega(G) + 3$ colors are used. Note also that in Step 2 we use only those colors from base color palettes which have not been used in Step 1, so overall no more than $\omega(G) + 3$ colors are used in total in the first phase.

To count the number of colors used in the second phase (Steps 4 and 5) notice that we divide the demands of each vertex in G_2 into four equal parts. Each vertex v participates in four from five rounds and assigns $\lceil d_2(v)/4 \rceil$ colors in

each round. Since in each round of Step 4 and 5a we use $\omega(G_2)/4 + 2$ colors from extra color palettes, we use only $5(\omega(G_2)/4 + 2)$ colors in total, while in 5b, when vertex cannot use an extra color palette, it borrows some colors from the base color palettes of its neighbors that have not been used by them in the previous steps, in order to avoid an introduction of any new colors.
Let $A(G)$ denote the number of colors used by our algorithm for the graph G. Thus, since $\omega(G_2) \leq \lceil \omega(G)/3 \rceil \leq \omega(G)/3 + 1$, the total number of colors used by our algorithm is at most

$$A(G) \leq \omega(G)+3+5\left(\frac{\omega(G_2)}{4}+2\right) \leq \omega(G)+3+\frac{5\omega(G)}{12}+\frac{5}{4}+10 \leq \frac{17}{12}\omega(G)+15.$$

So, the performance ratio for our strategy is 17/12 and we arrived at the thesis of Theorem 1.

4 Conclusion

We have given a 17/12-approximation algorithm for multicoloring hexagonal graphs. This implies a 17/12-competitive solution for the online frequency allocation problem, which involves servicing calls in each cell in a cellular network. The distributed algorithm is practical in the sense that frequency allocation for each base station is done locally, based on the information about itself and its neighbors only, and the time complexity is constant.

References

1. Chin, F.Y.L., Zhang, Y., Zhu H.: A 1-local 13/9-competitive Algorithm for Multicoloring Hexagonal Graphs, The 13th Annual International Computing and Combinatorics Conference COCOON 2007, LNCS, vol. 4598/2007, pp 526-536, Springer, Heidelberg (2007) .
2. Sparl, P., Zerovnik, J.: 2-local 4/3-competitive Algorithm for Multicoloring Hexagonal Graphs, Journal of Algorithms, vol. 55(1), pp 29-41 (2005)
3. McDiarmid, C., Reed, B.: Channel assignment and weighted coloring, Networks, vol. 36(2), pp. 114-117 (2000)
4. Sudeep, K.S., Vishwanathan, S.: A technique for multicoloring triangle-free hexagonal graphs, Discrete Mathematics, vol. 300, pp. 256-259 (2005)
5. Narayanan, L.: Channel assignment and graph multicoloring, Handbook of wireless networks and mobile computing, pp 71-94, Wiley, New York, (2002)
6. Narayanan, L., Shende, S.M.: Static frequency assignment in cellular networks, Algorithmica, vol. 29(3), pp 396-409 (2001)
7. Hale, W.K.: Frequency assignment: theory and applications, Proceedings of the IEEE, vol 68(12), pp 1497-1514 (1980)
8. Janssen, J., Krizanc, D., Narayanan, L., Shende, S.: Distributed Online Frequency Assignment in Cellular Network, Journal of Algorithms, vol. 36(2), pp 119-151 (2000)
9. Havet, F.: Channel assignment and multicoloring of the induced subgraphs of the triangular lattice, Discrete Mathematics, vol. 233, pp 219-231 (2001)

10. Sparl, P., Zerovnik, J.: 2-local 5/4-competitive algorithm for multicoloring triangle-free hexagonal graphs, Information Processing Letters, vol. 90(5), pp 239-246 (2004)
11. Zerownik, J.: A distributed 6/5-competitive algorithm for multicoloring triangle-free hexagonal graphs, International Journal of Pure and Applied Mathematics, vol. 23(2), pp 141-156 (2005)