2-local 4/3-competitive Algorithm for Multicoloring Hexagonal Graphs

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

May 11, 2009

Abstract

In the frequency allocation problem we are given a cellular telephone network which 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 2-local 4/3-competitive distributed algorithm for a multicoloring of hexagonal graph, which is 1-local, when we exclude some very specific subgraphs.

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 [6]). 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

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

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], [5]), $\chi_m(G) \leq \left\lceil \frac{4\omega(G)}{3} \right\rceil + O(1)$. Since all proofs of the upper 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. 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 vertices at distance at most k from v. For hexagonal graphs the best known 1-local algorithm for multicoloring is 17/12-competitive, and it has been presented in [1]. In this paper we develop a new, easier 2-local algorithm which with the same competitive ratio, but for many hexagonal graphs it would be 1-local. In next sections we present in which case our algorithm has to use information from vertices in distance 2, and we will see that this is rarely situation. In this paper we will prove in different way than in [2] that:

Theorem 1.1. *There is a 2-local distributed approximation algorithm for multicoloring hexagonal graphs which uses at most* $\left\lceil \frac{4}{3}\omega(G) \right\rceil + O(1)$ *colors. Time complexity of the algorithm at each vertex is constant.*

In [7] it was proved that a k-local c-approximate algorithm can be easily converted to a k-local c-competitive algorithm. Hence,

Corollary 1.2. *There is a 2-local 4/3-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 1.1.

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 $x\vec{p} + y\vec{q}$ of two vectors $\vec{p} = (1, 0)$ and $\vec{q} = (\frac{1}{2}, \frac{\sqrt{3}}{2})$. 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).

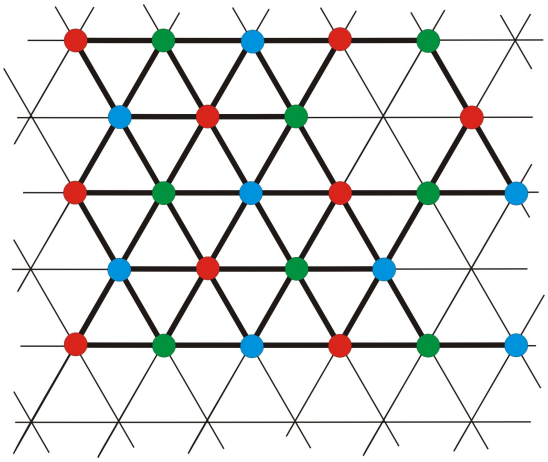

Figure 1: An example of a hexagonal graph

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. $bc(v) \in \{R, G, B\}$.

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 which is not a corner is called a *non-corner* (see Figure 2).

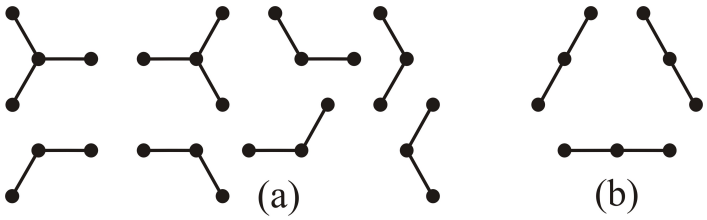

Figure 2: All possibilities for: (a) - corners, (b) - non-corners

From [5] we know that for any weighted bipartite graph H, $\chi_m(H) = \omega(H)$, and it can be optimally multicolored by the following procedure:

Procedure 2.1. *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.

Fact 2.2. *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 base color c, its palette is defined as a set of pairs $\{(c, i)\}_{i \in \aleph}$, $c \in \{R, G, B\}$. Such palettes are called *base color palette*. We will use also *extra color pallete*, $(c \in \{E\})$.

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 in distance 2. 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 three 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 bipartite subgraph of remaining graph, which is induced by all vertices except some corners. We use 2.1 and color such graphs optimally by using colors from extra color palette. First and second phase could be done in 1-local model. In the third phase we need to use information from neighbors in distance 2 to color remaining graph by using free colors from base color palletes.

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 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 the following function p on vertices of G_2:

- if $v = (x, y)$ is a non-corner:
  - if v has up-left or down-right neighbors in G_2 then $p(v) = x \bmod 2$
  - if v has up-right or down-left neighbors in G_2 then $p(v) = y \bmod 2$
  - if v has left or right neighbors in G_2 then $p(v) = x \bmod 2$
- if $v = (x, y)$ is a corner:
  - if $x \bmod 2 = y \bmod 2$ then $p(v) = y \bmod 2$
  - if $x \bmod 2 \neq y \bmod 2$ then $p(v) = 3$

Step 4 Apply Procedure 2.1 to bipartite graph induced by all vertices from G_2 with $p(v) \in \{1, 2\}$ to satisfy all demands in G_2 by colors from extra color palette. Construct a new graph $G_3 = (V_3, E_3, d_3)$ where $d_3 = d_2$ is the same as in G_2, $V_3 \subseteq V_2$ is the set of vertices with $p(v) = 3$ and $E_3 \subseteq E_2$ is the set of all edges in G_2 with both endpoints from V_3 (E_3 is induced by V_3). Such graph contains only isolated vertices and isolated edges.

Step 5 Construct graph $H = (V(H), E(H), d_H)$, where $V(H)$ is sum of vertices from G_3 and its light neighbors in G with different color than its heavy neighbors and $x \bmod 2 \neq$

y mod 2 Graph H contains only isolated path. Apply Procedure 2.1 to graph H and satisfy all demands by color from base color palette of light vertices in skladowa spojnosci!!!.

Step 6 Recolor such ligth vertices $v = (x, y)$ with $x \bmod 2 = y \bmod 2$:

- (a) if v has one neighbor in G_3 or two neighbors in G_3 at angle π then use free colors from $bc(v)$ base color palette
- (b) if v has two neighbors in G_3 at angle $\pi/3$ then use free colors from $bc(v)$ base color palette
- (c) if v has two neighbors in G_3 at angle $2\pi/3$ then use colors from extra color palette as in Step 4 for vertices with $p(v) = x \bmod 2$
- (d) if v has more than two neighbors in G_3 then use colors from extra color palette as in Step 4 for vertices with $p(v) = x \bmod 2$

Correctness proof

At the very beginning of the algorithm there is a 2-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 3.1. *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. □

Fact 3.2.

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

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 3.1 and Fact 2.2 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 also should have $d_2(v) \leq \lceil \omega(G)/3 \rceil$. Indeed, if $d_2(v) \leq \kappa(v)$, then it holds by Fact 2.2. 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. In 2-local model each vertex knows which of his neighbors are heavy, so he also knows if it is a corner or not, and where are situated their neighbors in G_2.

In Step 4 we would like to apply Procedure 2.1 to graph induced on G_2 by all vertices

with $p(v) \in \{1, 2\}$. It is easy to see that such graph is bipartite. For non-corners we use coordinates to find out if vertex is "odd" or "even" while for corners we take only those with compability parity. With value of function d_2 in every neighbors, each vertex has all knoledge to run Procedure 2.1.

After coloring vertices with $p(v) \in \{1, 2\}$ we have only corners from G_2 with $p(v) = 3$ and they induce a new graph G_3. If a corner in G_2 is surrounded by non-corners then it is isolated vertex in G_3. If a corner is adjacent with some other corner in G_2 it must be one of three situation from Figure 3.

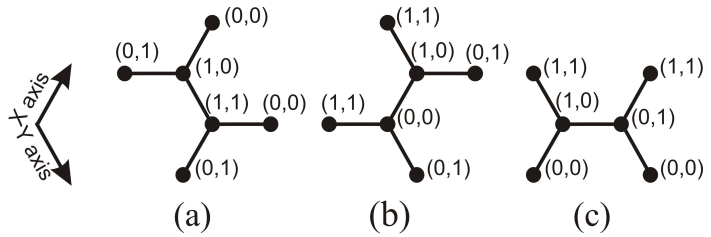

Figure 3: All possibilities for adjacent corners in G_2 (with coordinates mod 2)

As we can see on Figure 3 only connection from (c) can survive after coloring bipartite graph and theese where the only edges in G_3. Since such two edges cannot be adjacent in G_2, in G_3 all edges are isolated.

In Step 5 we take an isolated vertices from G_3 and go back to base color palettes. If vertex $v \in G_3$ is isolated, it means that it has three light 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 $d_3(v) \leq D_B(v)$. Let $\Delta = \{u, v, t\} \in \tau(T)$ be a triangle such that t is the green vertex which is heavy neighbor of v, and u is the blue vertex which is a light neighbor of v. Denote by $s_\Delta(t) = d(t) - a(u, v, t)$. Then we have

$$0 \geq d(v) - a(u, v, t) + d(t) - a(u, v, t) + d(u) - a(u, v, t) \geq$$
$$\geq d(v) - \kappa(v) + s_\Delta(t) + d(u) - \kappa(v) \geq d_1(v) + s_\Delta(t) - D_B(v) \geq$$
$$\geq d_2(v) + s_\Delta(t) - D_B(v) = d_3(v) + s_\Delta(t) - D_B(v)$$

Since t is a heavy neighbor of v, $d_3(v) < D_B(v)$. Therefore, vertex v has as much as $d_3(v)$ free colors from the blue base color palette at his disposal.

In Step 6 we take isolated edges from G_3. If edge $\{v, u\} \in G_3$ is isolated, it has to be situated like on Figure 4.

Notice that all neighbor of both adjacent vertices have the same base color. Without loss of generality, assume that $bc(v) = R, bc(u) = B$ and its slight neighbors' color is green.

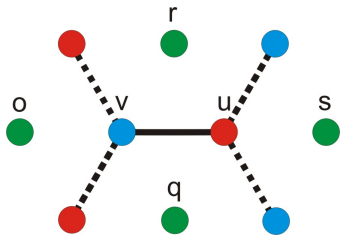

Figure 4: Edge in G_3 (two connected corners in G_2)

Problemy sa przedstawione ponizej na rysunku: ł

1-local model

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. Slight neighbors of v 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. If vertex v has three slight neighbors, then it is a corner.
In this terms we have that v has up-left or down-right neighbors in G_2 if they are not slight, etc.

The only problem is that, under 1-locality assumption, vertices cannot calculate value of d_2 of the neighbors, which is needed in Procedure 2.1 to calculate value $m(v) = \max\{\lceil d_2(u) \rceil : \{u,v\} \in E_2\}$. 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 $m'(v) = \max\{\lceil d_2^v(u) \rceil : \{u,v\} \in E_2\}$. 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:

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

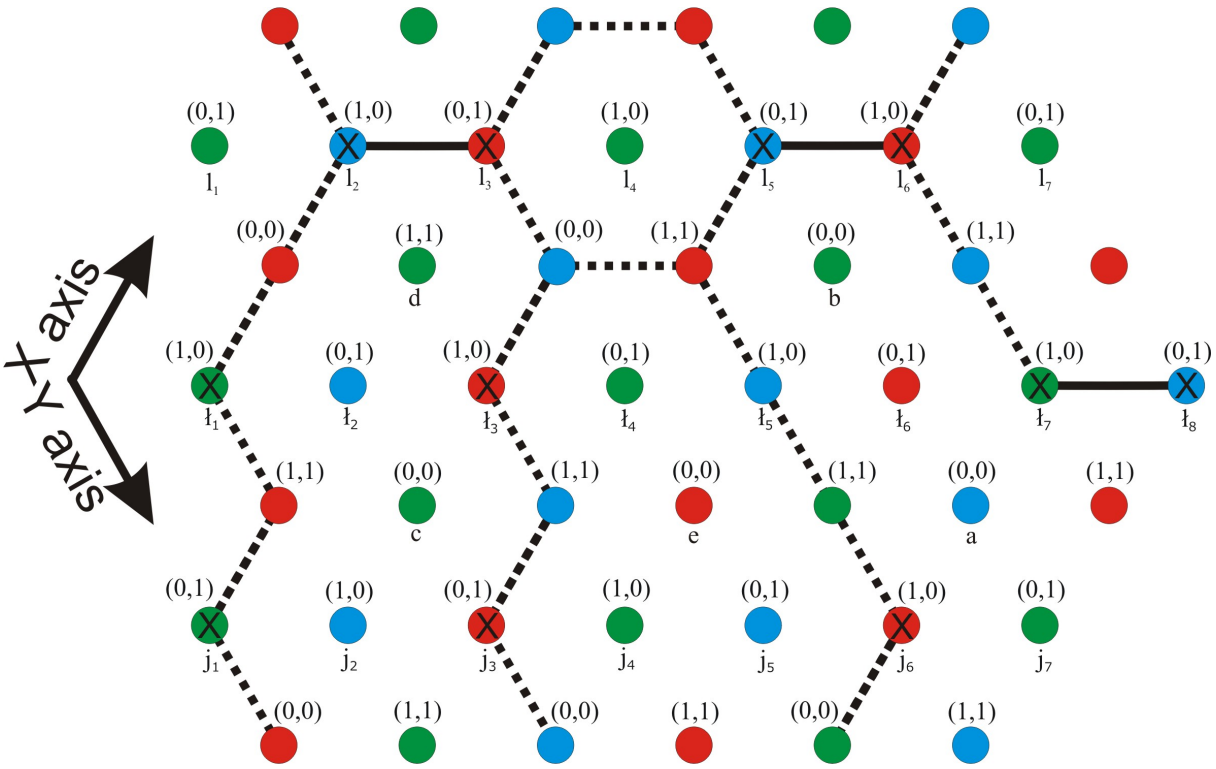

Figure 5: Situation after Step 4

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

a contradiction. Hence, if we use d_2^v instead of d_2 in each vertex from the second set of our bipartition, Procedure 2.1 works and uses at most $\lceil \omega(G)/3 \rceil$ colors.

Notice that in 1-local model vertex in G_3 do not know if it is isolated or not. It has to check, which neighbors of its non slight neighbors are slight. To do this we need to use 2-local communication. Also in Step 6, to calculate value of base function of neighbor corner we need 2-local communication. Therefore, our algorithm is not 1-local. But notice also, that if we know that in G_3 will not be any edges, than we can multicolor our graph G in 1-local model by using our algorithm.

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 Fact 2.2 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.

During the second phase (Step 4) we optimally color some subgraph of G_2 and from Lemma 3.2 use no more than $\omega(G_2)$. During the third phase (Steps 5 and 6) we borrow some colors from the base color palettes that have not been used 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

$$A(G) = \omega(G) + 3 + \omega(G_2) \leq \omega(G) + 3 + \frac{\omega(G)}{3} + 1 \leq \frac{4}{3}\omega(G) + 4.$$

So, the performance ratio for our strategy is 4/3 and we arrived at the thesis of Theorem 1.1.

4 Conclusion

We have given a 4/3-approximation algorithm for multicoloring hexagonal graphs, which is easier than previous known, and it is almost 1-local.

References

[1] R.Witkowski *1-local 17/12-competitive Algorithm for Multicoloring Hexagonal Graphs*, Submitted for publication, 2009.

[2] P. Sparl, J. Zerovnik *2-local 4/3-competitive Algorithm for Multicoloring Hexagonal Graphs*, Journal of Algorithms 55(1) (2005) 29-41

[3] C. McDiarmid, B. Reed *Channel assignment and weighted coloring*, Networks (2000) 114-117

[4] L. Narayanan *Channel assignment and graph multicoloring*, Handbook of wireless networks and mobile computing (2002) 71-94

[5] L. Narayanan, S.M. Shende *Static frequency assignment in cellular networks*, Algorithmica 29 (2001) 396-409

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

[7] J. Janssen, D. Krizanc, L. Narayanan, S. Shende *Distributed Online Frequency Assignment in Cellular Network*, Journal of Algorithms 36 (2000) 119-151