A Linear Time Algorithm for $7-[3]$-coloring Triangle-free Hexagonal Graphs

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

Janez Žerovnik
University of Ljubljana,
Faculty of mechanical engineering,
Aškerčeva 6,
SI-1000 Ljubljana, Slovenia
and
Institute of Mathematics, Physics and Mechanics,
Ljubljana, Slovenia
janez.zerovnik@fs.uni-lj.si, janez.zerovnik@imfm.si

March 6, 2011

Abstract

Given a graph G, a proper $n-[p]$-coloring is a mapping $f : V(G) \to 2^{\{1,\ldots,n\}}$ such that $|f(v)| = p$ for any vertex $v \in V(G)$ and $f(v) \cap f(u) = \emptyset$ for any pair of adjacent vertices u and v. $n-[p]$-coloring is closely related to multicoloring. Finding multicoloring of induced subgraphs of the triangular lattice (called *hexagonal graphs*) has important applications in cellular networks. In this article we provide an algorithm to find a 7-[3]-coloring of triangle-free hexagonal graphs in linear time, which solves the open problem stated in [10] and improves the result of Sudeep and Vishwanathan [11], who proved the existence of a 14-[6]coloring.

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

1 Introduction

A fundamental problem concerning cellular networks is to assign sets of frequencies (colors) to transmitters (vertices) in order to avoid unacceptable interferences [1]. The number of frequencies demanded at a transmitter may vary between transmitters. In a usual cellular model, transmitters are centers of hexagonal cells and the corresponding adjacency graph is a subgraph of the infinite triangular lattice. An integer $d(v)$ is assigned to each vertex of the triangular lattice and will be called the *demand* of the vertex v. The vertex weighted graph induced on the subset of the triangular lattice of vertices of positive demand is called a *hexagonal graph*, and is denoted $G = (V, E, d)$. Hexagonal graphs arise naturally in studies of cellular networks. A *proper multicoloring* of G is a mapping f from $V(G)$ to subsets of integers such that $|f(v)| \geq d(v)$ for any vertex $v \in V(G)$ and $f(v) \cap f(u) = \emptyset$ for any pair of adjacent vertices u and v in the graph G. The minimal cardinality of a proper multicoloring of G, $\chi_m(G)$, is called the *multichromatic number*. Another invariant of interest in this context is the *(weighted) clique number*, $\omega(G)$, defined as follows: The weight of a clique of G is the sum of demands on its vertices and $\omega(G)$ is the maximal clique weight on G. Clearly, $\chi_m(G) \geq \omega(G)$ and it is known that the upper bound $\chi_p(G) \leq (4/3)\omega(G) + O(1)$ is best possible in general [7, 5, 6].

Better bounds can be obtained for triangle-free hexagonal graphs. The conjecture due to McDiarmid and Reed [5] is that $\chi_m(G) \leq (9/8)\omega(G) + O(1)$ holds for triangle-free hexagonal graphs. In [3] a distributed algorithm with competitive ratio 5/4 is given. In [8] the authors report the existence of 2-local distributed algorithm with competitive ratio 5/4, while an inductive proof for ratio 7/6 is reported in [2]. A 2-local 7/6-competitive algorithm for a sub-class of triangle-free hexagonal graphs is given in [9]. A special case of a proper multicoloring is when d is a constant function. For example, a $7 - [3]$-coloring is an assignment of three colors between 1 and 7 to each vertex.

An elegant idea that implies the existence of a $14 - [6]$-coloring as well as linear time algorithm to fnd it, was presented in [11]. In [2] the existence of a $7 - [3]$-coloring was shown, and the inductive proof can be transformed in a polynomial time algorithms. A shorter proof is providede in [10], but main idea of proof is based on the 4-colour theorem. In this paper we give an linear time algorithm for $7 - [3]$-coloring an arbitrary triangle-free hexagonal graph G. The main contribution is that we provide a linear coloring algorithm for the auxiliary graphs used in [10]. Our proof is independent of the 4-color theorem and the construction gives rise to a polinomial time algorithm for $7 - [3]$-coloring of triangle-free hexagonal graphs.

The paper is organized as follows. In the next section we formally define some basic terminology. In Section 3 we present an algorithm for $7 - [3]$-coloring of arbitrary triangle-free hexagonal graph G and prove its correctness.

2 Basic definitions and useful facts

A vertex weighted graph is given by a triple $G(E, V, d)$, where V the set of vertices, E is the set of edges and $d : V \to \mathbb{N}$ is a weight function assigning (nonnegative) integer demands to vertices of G.

Following the notation from [5], 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*. Assume that we are given a weight function $d : V \to \{0, 1, 2, \ldots\}$ on vertices of triangular lattice. We define a *weighted hexagonal graph* $G = (V, E, d)$ as an induced subgraph on vertices of positive demand on the triangular lattice (see Figure 1). Sometimes we want to work with (unweighted) hexagonal graphs $G = (V, E)$ that can be defined as induced graphs on subsets of vertices of the triangular lattice.

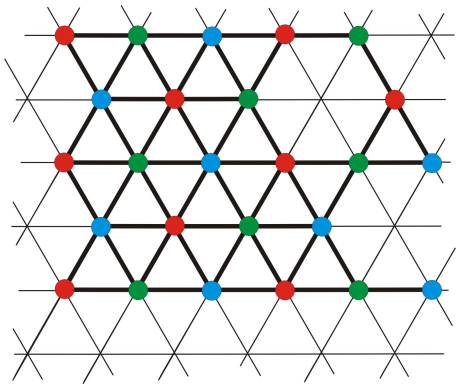


Figure 1: An example of a hexagonal graph (with base coloring)

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 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*.

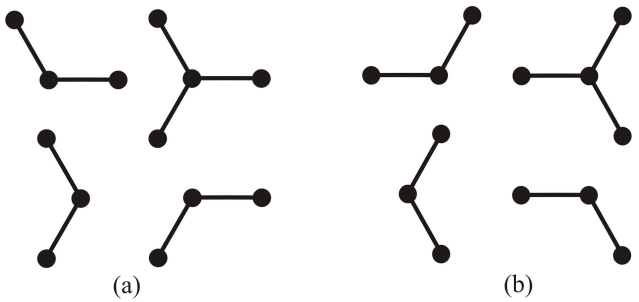


Figure 2: All possibilities for: (a) - left corners, (b) - right corners

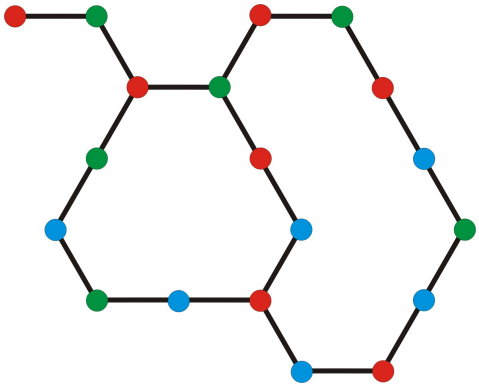


Figure 3: An example of a triangle-free hexagonal graph (with base coloring)

In triangle-free hexagonal graph each corner has all of its neighbors with the same base color – in this case we call third color the *free color*.

A *vertical path* is a path in triangle-free hexagonal graph which does not contain edges from any vertex to its right or left neighbor.

Bridge is a path in triangle-free hexagonal graph which contains only edges from vertices to its right or left neighbor.

Each edge in triangle-free hexagonal graph is in some vertical path or bridge.

Our algorithm is based on the following observations:

Lemma 2.1 *Let P be an arbitrary path of length at least 2. Let $s, t \in P$ be the endpoints of the path P. Denote by $n(s), n(t)$ neighboring vertices in P of respectively s and t. Assume that with s and t we colligate sets of two elements, respectively S and T, $S, T \subset \{1, 2, 3, 4\}$. We can find in linear time $4 - [2]$ coloring of path $P \backslash \{s, t\}$ such that we can choose in s and t one color from sets respectively S and T to avoid conflict with vertices $n(s)$ and $n(t)$.*

Proof: Recall that path is bipartite graph, and bipartition can be easily found in linear time. Assume S and T are given. A procedure of finding $4-[2]$-coloring and choosing correct element from sets S and T can be as follows:

Step 1 Find the bipartition of the path.

Step 2 Choose one element $i \in S$.

Step 3 Choose one element $j \in T$, $j \neq i$.

Step 4 If s and t are in the same set of bipartition (path is of even length) then for all vertices of this set put colors $\{i, j\}$ and for vertices from other set put colors $\{1, 2, 3, 4\} \backslash \{i, j\}$.

If s and t are in the different sets of bipartition (path is of odd length) then for all vertices of set with s put colors $\{i\}$ and for vertices from other set use color $\{j\}$. Remaining colors take arbitrary such that coloring is proper.

Observation 2.1 *Lemma 2.1 is also correct, if in the set S we put only one element.*

Lemma 2.2 *Let P be an arbitrary path. Let $X \in P$ be a subset of vertices of this path. We call vertices $x \in X$* good *vertices. Assume that with each good vertex we associate sets of two elements (*good set*), subset of $\{1, 2, 3, 4\}$. We can find in linear time a $4-[2]$coloring of path $P \backslash X$ such that for each good vertex at least one color from its good set is not used by its neighbors.*

Proof: We can put an order on path from one endpoint to the second one. Now we can use procedure from Lemma 2.1 to color path from the first good vertex to the second good vertex. Then, using Observation 2.1 we can extend this coloring to path from the second good vertex to the third good vertex, and so on. If path starts or ends with vertices that are not good, we can easily extend our coloring to those vertices.

3 Algorithm and its correctness

Our algorithm consists of two main phases. In the first phase all vertices take its base colors, and left corners take also base color that is not used by its neighbors (the free color). In the second phase we remove horizontal edges and remaining graph is set of independent paths. Roughly speaking, we can start from the most right vertical path and make its $4-[2]$-coloring arbitrary. Then we extend the coloring to any horizontal path that meets the already colored vertical paths which gives sets of two colors to left corners of some vertical paths that were not colored yet. The procedure repeats by choosing another vertical path with all good left corners until all the graph is colored.

More precisely, our algorithm consists of the following steps:

Algorithm

Step 1 For each vertex $v \in V$ assign to v its base color.

Step 2 For each left corner $v \in V$ assign to v its free color.

Step 3 For each left corner $v \in V$ that has no right neighbor, assign set of two number from $\{1, 2, 3, 4\}$ arbitrary (make it good).

Step 4 Divide G into vertical paths and bridges.

Step 5 While exist a vertex not fully colored do Step 5.1.

Step 5.1 Take a non colored vertical path in which all left corners are good and use procedure from Lemma 2.1 to color it. Extend the coloring to all not colored bridges connected with this vertical path.

If some bridge is connected with left corners then assign to this left corner set of two number from $\{1, 2, 3, 4\}$ (make it good), which is forced by coloring.

4 Correctness proof

Step 1: nothing to prove.

Step 2: nothing to prove.

Step 3: After this step always exist at least one vertical path with all left good corners. It is since the transitive closure of relation "being on left" is strict partial order. Formally:

Definition 4.1 *We can say that vertical path P_1 is* on left *of vertical path P_2 if they are connected with bridge M such that $v = M \cap P_1$ is a right corner, and $u = M \cap P_2$ is a left corner. In this case we will write $P_1 < P_2$.*

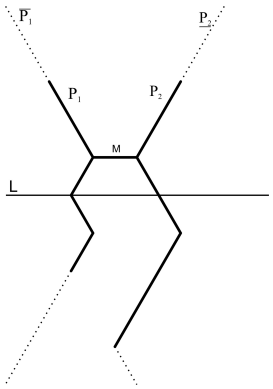
Lemma 4.1 *The transitive closure of relation from Definition 4.1 is strict partial order.*

Proof: To prove the lemma we will show that relation $<$ on vertical paths is irreflexive and asymmetric. We will also show that the transitive closure of this relation is well defined.

Irreflexivity ($\neg(P_1 < P_1)$): is easy just from definition – we cannot create bridge from a vertical path to itself.

Asymmetry ($P_1 < P_2 \Rightarrow \neg(P_2 < P_1)$): consider two finite vertical paths P_1 and P_2 such that $P_1 < P_2$, i.e. exist a bridge M such that $v = M \cap P_1$ is a right corner, and $u = M \cap P_2$ is a left corner. Expand those vertical paths into infinite vertical paths $\overline{P_1}$ and $\underline{P_2}$ in such a way that to the top vertex of P_1 we attach infinite up-left path, and to the bottom vertex of P_1 we attach infinite down-left path (see Figure 4). In the same way, we expand P_2

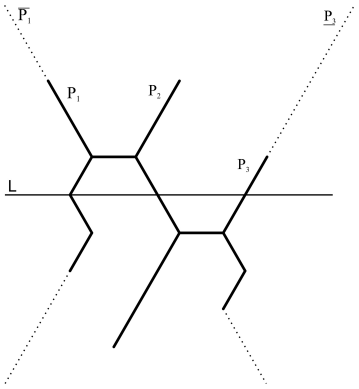
using up-right path at the top, and down-right at the bottom. Now, if we create infinite horizontal line L, it always has exactly one common point with $\overline{P_1}$ ($\overline{v} = L \cap \overline{P_1}$) and one with $\underline{P_2}$ ($\underline{u} = L \cap \underline{P_2}$). Let $dist_L(\overline{P_1}), \underline{P_2}) = x(u) - x(v)$ denote the distance on line L. For the line L that covers bridge ($M \in L$) we know that $dist_L(\overline{P_1}, \underline{P_2}) > 0$. If we move the line by 1 up or down, the *dist* can either remain the same or it can change by 1. Therefore, $dist_L(\overline{P_1}, \underline{P_2}) > 0$ for all lines L, because if it goes down to 0 ($\overline{v} = \underline{u}$) there has to be a line L' above (or below) that has $dist_{L'}(\overline{v'}, \underline{u'}) = 1$, and then $(\overline{v}, \overline{v'}, \underline{u'})$ form a triangle, which is not possible in triangle-free graph. Thus, P_1 and P_2 never meet, and the set of vertices that are to the right of $\underline{P_2}$ and the set of vertices that are to the left of $\overline{P_1}$ do not intersect. Formally, vertex u is to the right of extended vertical path P if there is a horizontal line L, $u \in L$, $L \cap P = v$ and $x(v) < x(u)$. Similarly, vertex u is to the left of extended vertical path P if there is a horizontal line L, $u \in L$, $L \cap P = v$ and $x(v) > x(u)$. Hence, any bridge starting at right corner of P_2 goes to the right of P_2 and hence cannot end in the left corner of P_1, so $\neg(P_2 < P_1)$.



Now, we will prove that a transitive closure of this relation is well defined ($P_1 < P_2 \wedge P_2 < P_3 \Rightarrow \neg(P_3 < P_1)$). Consider three finite vertical paths P_1, P_2, P_3 such that $P_1 < P_2$ and $P_2 < P_3$. Expand into infinite vertical paths $\overline{P_1}$ and $\underline{P_3}$ like before (see Figure 4). Consider horizontal lines that intersect P_2. Each of these lines meets all of three vertical paths $\overline{P_1}$, P_2 and $\underline{P_3}$. As in previous paragraph, we can see that for each such line L $x(u) < x(v) < x(w)$ where $u = L \cap \overline{P_1}$, $v = L \cap P_2$ and $w = L \cap \underline{P_3}$. As $dist_L(\overline{P_1}, \underline{P_3}) > 0$ for the line L that meets the bottom vertex of P_2 it is true for any line below. Similarly $dist_L(\overline{P_1}, \underline{P_3}) > 0$ for any line above the top vertex of P_2. Hence, any bridge starting at right corner of P_3 goes to the right of P_3 and cannot end in the left corner of P_1, so $\neg(P_3 < P_1)$.

Step 4: We can divide G into well separated vertical paths. Argument is the same as in Step 3, and it was also shown on [11].

Step 5: If some vertex are not colored, there always exist a path with all left good corners. Argument is the same as in Step 3 – it is impossible to create cycle in such way that bridges connect only not colored left and right corners in one order.



Step 5.1: It works, since we proved Lemma 2.1. Bridges are easy to extend cause they are bipartitie.

Concerning the running time, it is easy to check that all Steps 1-4 of the algorithm run in time linear on $|V(G)|$. In Step 4 we can remember all vertical paths and for each we can associate number of its left corners. Then if for every time we make some left corners good, we add number of good vertices on path, so we can easily check how many good left corner are in it, and if those numbers are the same, we can take such path. So for each path we can decide in constant time whether all its left corners are good or not. Procedure in Lemma 2.1 works in linear time according to number of vertices on path. Extention to bridges also work in linear time refers to number of vertices on bridge. In each path each vertex is only once in Step 5.1, so time complexity of loop in Steps 5 is linear on $|V(G)|$.

5 Conclusion

In this article we provided an algorithm for $7-[3]$-coloring triangle-free hexagonal graphs. The described $7-[3]$-coloring can be extended to a proper multicoloring with at most $\lceil(7/6)\omega(G)\rceil + O(1)$ colors of any weighted triangle-free hexagonal graph. The main idea (used in for example in [4, 8, 3, 6]) is to divide the set of colors into 7 palettes and to use the algorithm for $7-[3]$-coloring to define the order of color palettes from which vertices will take colors from. We omit the details.

Recall that both $5-[2]$-coloring of triangle-free hexagonal graphs [8] and a 4/3-competitive algorithm for multicoloring hexagonal graphs [7] are distributed and time complexity is constant. For the $7-[3]$-coloring algorithm presented in in this paper it is no easy way to make it in constant time in distributed model, since Procedure 2.1 doesn't work for long path with many good vertices. Nevertheless, most steps of the algorithm can be easily performed locally. Therefore, it is an interesting question whether one can find a coloring of our auxiliary graph G in a distributed way, using its rich structural properties.

References

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

[2] Havet, F. *Channel assignment and multicoloring of the induced subgraphs of the triangular lattice*, Discrete Mathematics, vol. 233, pp 219-231 (2001)

[3] Havet, F., Žerovnik, J. *Finding a Five Bicolouring of a Triangle-free Subgraph of the Triangular Lattice*, Discrete Mathematics vol. 244, pp 103-108 (2002)

[4] 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)

[5] McDiarmid, C., Reed, B. *Channel assignment and weighted coloring*, Networks, vol. 36(2), pp. 114-117 (2000)

[6] Narayanan, L., Shende, S.M. *Static frequency assignment in cellular networks*, Algorithmica, vol. 29(3), pp 396-409 (2001)

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

[8] Šparl, P., Žerovnik, J. *2-local 5/4-competitive algorithm for multicoloring triangle-free hexagonal graphs*, Information Processing Letters, vol. 90(5), pp 239-246 (2004)

[9] Šparl, P., Žerovnik, J. *2-local 7/6-competitive algorithm for multicoloring a sub-class of hexagonal graph*, International Journal of Computer Mathematics,, First published on: 22 July 2009 (iFirst) doi: 10.1080/00207160802562531

[10] Sau, I., Šparl, P., Žerovnik, J. *7/6-approximation Algorithm for Multicoloring Triangle-free Hexagonal Graphs*, submitted for publication.

[11] Sudeep, K.S., Vishwanathan, S. *A technique for multicoloring triangle-free hexagonal graphs*, Discrete Mathematics, vol. 300, pp. 256-259 (2005)