Distributed On-line Frequency Assignment in Cellular Networks *

Jeannette Janssen† Danny Krizanc ‡ Lata Narayanan§ Sunil M. Shende¶

October 22, 1999

Abstract

A cellular network is generally modeled as a subgraph of the triangular lattice. The *distributed online frequency assignment* problem can be abstracted as a multicoloring problem on a weighted graph where the weight vector associated with the vertices models the number of calls to be served at the vertices, and is assumed to change over time. In this paper, we develop a framework for studying distributed online frequency assignment in cellular networks. We present the first distributed online algorithms for this problem with proven bounds on their competitive ratios. We show a series of algorithms that use at each vertex, information about increasingly larger neighborhoods of the vertex, and achieve better competitive ratios. In contrast, we show lower bounds on the competitive ratios of some natural classes of online algorithms. Some of our algorithms are optimal for the classes of algorithms studied.

*Research supported by NSERC, Canada. A preliminary version of this paper will appear in Proceedings of STACS '98.

†Department of Mathematics and Statistics, Dalhousie University, Halifax, NS B3H 3J5 Canada, email: janssen@mscs.dal.ca.

‡School of Computer Science, Carleton University, Ottawa, ON Canada, email: krizanc@scs.carleton.ca.

§Department of Computer Science, Concordia University, Montreal, Quebec, Canada, H3G 1M8. email: lata@cs.concordia.ca, FAX (514) 848-2830.

¶Department of Computer Science, Rutgers University, Camden, NJ 08102, USA. email: shende@crab.rutgers.edu, FAX (856) 225-6624.

1 Introduction

Cellular data and communication networks are usually modeled as graphs with each node representing a base station (sometimes called a *cell*) in the network. At any given time, a certain number of active connections (or *calls*) are serviced by their nearest base station. In most cases (especially in FDMA technology), service involves the assignment of a frequency to each client call in a manner that minimizes or avoids radio interference between different calls in the network. A similar problem also arises in cellular networks employing CDMA technology. A common abstraction is the assumption that if two calls are assigned the same frequency, they would interfere with one another if and only if they originate in the same cell or in physically adjacent cells. For this reason, we refer to the graphs that model these networks as *interference graphs*. However, cellular networks have a limited spectrum of radio frequencies available to handle calls and the efficient shared utilization of the bandwidth is critical to the smooth operation of the network. In this paper we study the *distributed online frequency assignment* problem which consists of designing a distributed online interference-free frequency allocation protocol for a network where the number of calls per cell changes over time.

The graphs most often used to model cellular networks are finite portions of the infinite triangular lattice. The reason for adopting this particular geometry stems from the fact that cells are uniformly distributed in the geographic area of the network. The attenuation of radio signals occurs in a circular manner, and hence, the cell's calling area can be idealized as a regular hexagon. The triangular lattice representing the network is simply the planar dual of the resulting Voronoi diagram. We refer to a finite induced subgraph of the triangular lattice as a *hexagon graph* (see Figure 1 for an example). Unless otherwise specified, in the rest of this paper, the interference graphs we consider are always *hexagon graphs*.

The static frequency assignment problem incorporating interference constraints can be abstracted as follows. Let $G = (V, E, w)$ denote an interference graph where each node $v \in V$ has an associated nonnegative integer weight, $w(v) \geq 0$. The graph G models a static snapshot of the network at some instant in time, with the nodes representing cells and the weights representing the number of calls that require service in the cell. Our problem is to properly *multicolor* the graph G, *i.e.* we are required to assign $w(v)$ *distinct* colors to each v such that for every edge, $(u, v) \in E$, the set of colors assigned to the endpoints u and v are *disjoint*. The *span* of a multicoloring is the total number of colors used in the coloring. In particular, we are interested in a proper multicoloring of G whose span is equal to the minimum number of colors required to multicolor G, denoted by $\chi(G)$. 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 is convenient to treat the color palette of available colors as the set of natural numbers; we further assume without loss of generality that any such palette can be suitably reordered or partitioned.

The complexity of the static version of the problem has received considerable recent attention. Let the weight of a maximal clique in G be defined as the sum of the weights of the nodes belonging to the clique; note that G being a subgraph of the triangular lattice, the only maximal cliques are isolated nodes, edges or triangles. It is easy to see that $\chi(G)$

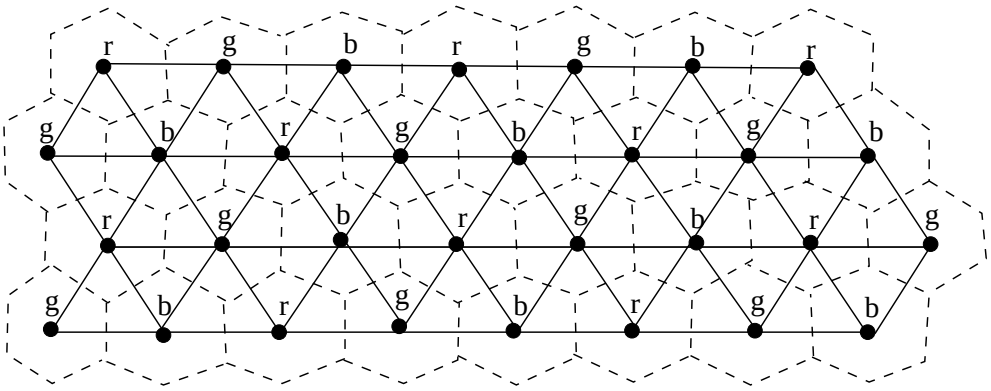

Figure 1: A hexagon graph and a 3-coloring of the nodes of the graph. The hexagonal area around each node represents the calling area it serves.

must be at least the weight of any maximal clique in the graph. It has been shown recently that the problem of multicoloring hexagon graphs optimally is NP-hard [9]. In terms of upper bounds, there is a vast literature on algorithms for frequency assignment on graphs (especially hexagon graphs) which claim to use few colors in practice, but have no proven bounds on their performance [1, 5, 7, 11, 13]. A well-known algorithm often referred to as *Fixed Allocation (FA)*, 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 node that has base color 1 uses colors from the first set, and a node with base color 2 or 3 uses colors from the second or third sets respectively. It is easy to see that this algorithm is an approximation algorithm with performance ratio 3. Janssen *et. al.* [4] propose a different algorithm called *Fixed Preference Allocation (FPA)* that is guaranteed to use no more than $\frac{3}{2}$ times the minimum number of colors required. Approximation algorithms with performance ratio $\frac{4}{3}$ have recently been shown in [9] and [10].

In the online case, the interference graph to be multicolored changes over time. We model these changes as an ordered sequence of interference graphs, $\{G_t = (V, E, w_t) : \ t \geq 0\}$, where w_t represents the set of calls to be serviced at time t. At time instant t, an online algorithm must arrange to color the graph G_t before moving on to the graph G_{t+1} at the next time instant $t+1$. It must perform this coloring with no knowledge of the later graphs in the sequence. Very little is known about the online version of the problem. There has been a lot of work on online graph *coloring* (see for example, [12, 8, 2]), where the graph to be colored is revealed one node at a time in every time step, and the algorithm must assign a color to the newly revealed node. However, we are interested in the *multicoloring* problem, where the entire graph is known in advance, and it is the weights on each node that change in every time step. The results of [3] imply that for general k-colorable graphs, no online multicoloring algorithm can have a competitive ratio smaller than $k/2$, a bound which is met by an online version of FPA. They also show that every algorithm which is not allowed to recolor has a competitive ratio at least k on such graphs, a bound which is met by FA. However, the lower bounds involve constructions of non-planar graphs; there are no known lower bounds on the competitive ratio of algorithms on hexagon graphs. Furthermore, the online algorithms that have proven upper bounds on the competitive ratio use information

about the changing state of the entire graph and are therefore not distributed.

In this paper we develop a reasonable operational model in which the problem of distributed online frequency assignment in cellular networks can be studied. In particular, this involves a number of considerations: a precise delineation of the various kinds of admissible online algorithms for the problem (Section 1.1), the framework and efficiency measures under which the performance of any such algorithm can be meaningfully evaluated (Section 1.2), and models in which lower bounds can be shown (Section 1.3). A brief summary of our results is given in Section 1.4. We present our upper bounds in Section 2 and lower bounds in Section 3. Conclusions and future directions for research are discussed in Section 4.

1.1 Admissible distributed online algorithms

Since most practical frequency assignment algorithms are required to be distributed in nature, it is convenient to describe the algorithm as though it were running simultaneously on servers, one server per active node in the network. In fact, the scope of our paper is limited to *distributed* and *deterministic* algorithms: each server running the algorithm is independently responsible for the color assignment at its resident node at any given time instant. Furthermore, this local assignment is computed deterministically based upon a limited amount of information. The time-indexed sequence of interference graphs is the online input to the algorithm - but presented in a manner that is completely distributed, *i.e.* as though each node v is presented synchronously with weight $w_t(v)$ at time $t \geq 0$ indicating the number of calls that need color assignment. Additionally, v may get requests to *drop* certain calls, which can be identified by the colors assigned to them in the previous step. For example, if $w_t(v) < w_{t-1}(v)$, then at least $w_{t-1}(v) - w_t(v)$ calls to be dropped are specified as part of the input at time t. In response, v may gather some local information and use it to provide service at time t by allocating $w_t(v)$ distinct colors to its local calls without conflicting with assignments at neighboring nodes. Overall, between successive time steps only a constant amount of communication and computation is permissible.

Various subtle issues arise in this framework. For instance, what kind of information ought to be reasonably admitted? While it is difficult to answer this question in its full generality, we propose the following restrictions motivated by practical considerations typical in most cellular networks. We insist that no *global* knowledge of the current state of the network be available at any node or small set of nodes in the network. However, it is assumed that nodes may be permitted to gather some limited amount of information from their local neighborhood between successive time instants. In particular, for integers $k \geq 1$, we define the *k-locality* of a node v to be the induced subgraph consisting of those nodes in G whose graph distance from v is less than or equal to k. The maximum weight taken over all the maximal cliques in the k-locality of v, is denoted by $D_k(v)$. It is easy to see that the maximum of $D_k(v)$ over all nodes v in the graph, is a lower bound on $\chi(G)$. For some small constant $k \geq 0$ independent of the input weight sequence, we say that an algorithm is *k-local* if between successive time instants, the values of the weights at time t at every node in v's k-locality, and only those weights, are available at v to decide its allocation for time t. Furthermore, it can be assumed that at the very beginning (at time $t = 0$), some "hard-wired" or pre-computed information, independent of the input weight sequence, is available to each node for free. In general, this pre-computed information may be an ar-

bitrary finite function of a node's label in a fixed labeling of the triangular lattice. The algorithms discussed below use only the fact that the nodes of the lattice can be partitioned into a small constant number of stable sets depending on their distance from a fixed origin. For example, the Fixed Allocation Local (FA-Local) and Fixed Preference Allocation Local (FPA-Local) algorithms discussed below depend only on the existence of a 3-coloring of the lattice. On the other hand, the remaining two algorithms additionally use a 2-coloring of every directional axis in the graph.

A second important issue concerns whether or not a node, when allocating colors for the next time step, can change the colors it has already assigned to its local calls on previous steps. Recall that in practice, this means changing the frequency previously assigned to an ongoing call. We say an algorithm is *non-recoloring* if once having assigned a color in response to a particular new call it never changes that assignment (*i.e.*, recolors the call). The algorithm FA-Local is an example of a non-recoloring algorithm. Recent technical developments, however, allow for a limited rearrangement of frequencies. All the other algorithms discussed below are *recoloring* algorithms, *i.e.* a node may change the assigned color of a call. For the k-local algorithms that we discuss in this paper, such color changes occur only in response to changes in demand within the node's k-locality.

1.2 Performance measures

We adapt a standard yardstick for measuring the efficacy of online algorithms: that of *competitive ratios* [6]. Given an online algorithm P that processes a sequence of N interference graphs G_t, $t = 0, \ldots, N$, let $\mathcal{S}(P_t)$ denote the span of the multicoloring computed by P after step t, *i.e.* after graph G_t has been processed. Let $\mathcal{S}_N(P) = \max_t\{\mathcal{S}(P_t)\}$ and $\chi_N(G) = \max_t\{\chi(G_t)\}$. We say that P is a *c-competitive* algorithm if and only if there is a constant b independent of N such that for any input sequence,

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

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

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

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

1.3 Lower bound models

We provide lower bounds on the competitive ratio of algorithms in a number of models. Notice that if the online algorithm is provided at each step with a complete description of the weights at every node in G, then the problem reduces to that of solving a series of static frequency assignment or multi-coloring problems. To capture the distributed nature of the problem we must restrict the possible actions taken by a node during the execution of a

k-local algorithm. In particular, nodes are only permitted to access information about the current state of their respective k-localities. By constraining the algorithms in very natural ways, we are able to provide lower bounds in models that capture the properties of most reasonable distributed algorithms including the algorithms discussed in this paper. In some cases the algorithms we provide are optimal for their class.

The first restriction we consider is that of the recoloring ability of the online algorithms. We show lower bounds for both recoloring and non-recoloring algorithms. In the recoloring case we add the constraint that recoloring can only occur in response to a change in demand within a node's immediate neighborhood. We say a recoloring algorithm has *recoloring distance* ℓ if a node recolors its calls during a time step only if a change of demand has occurred within its ℓ-locality.

We further make a distinction between models based upon the kind of information the nodes can use in making their assignments. In particular we consider a class of algorithms in which the pre-computed information is limited to a fixed 3-coloring of the lattice and for which nodes with similar localities act the same. This class includes the algorithms FA-Local and FPA-Local discussed below. More precisely, assume a fixed 3-coloring of the triangular lattice. Call two nodes *k-view-equivalent* if they are in the same color class and there is an isomorphism which maps one node's k-locality to the other's preserving the colors assigned to calls. An algorithm is said to be *k-view and color class determined* if on each step, k-view-equivalent nodes make precisely the same color assignments.

1.4 Our results

In this paper we present the first distributed online algorithms for frequency assignment on hexagon graphs with proven bounds on their competitive ratio along with lower bounds on the performance of online algorithms falling in naturally constrained classes.

All of our upper bounds are obtained by modifying known (global) approximation algorithms for the static frequency assignment problem. The required modifications have the effect of making the coloring decisions depend only on local information. The results indicate that the larger the locality taken into account, the better the competitive ratio the algorithm attains. The algorithm FA-Local is a straightforward modification of FA which performs no recoloring, is 0-local and has a competitive ratio of 3. In contrast, FPA-Local (a modification of FPA) is a recoloring 1-local algorithm with competitive ratio bounded by $\frac{3}{2}$. Two further algorithms are presented which are modifications of the global static algorithm first described by Narayanan and Shende [10]. The first of these is a 2-local recoloring algorithm with a competitive ratio of $\frac{17}{12}$. By expanding the locality under consideration to radius 4, we give a 4-local recoloring algorithm which achieves a competitive ratio of $\frac{4}{3}$. It should be noted that all the recoloring k-local algorithms described in the paper have recoloring distance at most k.

We present lower bounds for recoloring distance bounded algorithms and for view and color class determined recoloring and non-recoloring algorithms. For recoloring algorithms limited to recoloring distance k, we show a lower bound of $1 + \frac{1}{4(k+1)}$. (For the special case of $k = 0$ this can be improved to $\frac{9}{7}$.) This implies that any algorithm that depends only on information from a constant radius neighborhood around each node in making recoloring decisions for that node, can never achieve competitive ratio 1. Note that all the local

recoloring algorithms described above have recoloring distance limited to the locality that they have knowledge of. In the setting of arbitrary non-recoloring algorithms we show a lower bound of 2 on the competitive ratio. We show that for any k and any $\epsilon > 0$, a k-view and color class determined non-recoloring algorithm must have competitive ratio at least $3 - \epsilon$. This implies that the algorithm FA-Local is optimal for this class. We also show that for any k, a k-view and color class determined recoloring algorithm (with any recoloring distance ℓ) must have competitive ratio at least $3/2$, showing that FPA-local is optimal for this class. Our results imply that both the ability to recolor and the restriction to the k-view and color class determined algorithms have *provable* effects on the performance of an algorithm for multicoloring on hexagon graphs.

2 Online multicoloring of the triangular grid

In this section, we give online algorithms for frequency assignment on an induced subgraph of the triangular lattice. We describe k-local online algorithms for $k = 0, 1, 2$, and 4 and show upper bounds on the competitive ratio of these algorithms. We begin with a key technical definition. A *static* k-local distributed algorithm for multicoloring gets as input a graph G corresponding to a snapshot of the network at some time step; it has the further property that the color assignment at any node depends only on the weights in the k-locality of the node and some pre-computed information about the lattice. The following general lemma enables us to derive online algorithms with performance guarantees:

Lemma 1 *Let A be a k-local static approximation algorithm for multicoloring with performance ratio α. Then A can be converted to a k-local α-competitive online algorithm for multicoloring.*

Proof: For the online case, each node runs the k-local static algorithm independently at every step. If the color spectrum obtained by node v is the same as the one used by it in the previous step, then v does not have to recolor. Otherwise, if some colors previously assigned to currently active calls do not appear in the newly computed set of colors, then the algorithm has to recolor those calls. Since the number of colors used by the static algorithm (and therefore, the online algorithm) at any step is at most α times the minimum number of colors required at that step, the online algorithm is α-competitive. □

To paraphrase Lemma 1, we only need describe a correct static algorithm which then translates to a corresponding online algorithm with the same competitive ratio. We note, however, that the conversion is only guaranteed if *the algorithm has the ability to recolor.*

The fact that FA-Local (the local version of FA) is a 0-local algorithm with performance ratio 3 is a folklore result; to color new calls at time t, red, blue, and green nodes simply use the smallest available colors from the sets 0, 1 and 2 mod 3 respectively. Note that previously existing calls are never recolored: FA-Local is a non-recoloring algorithm. In the sequel, we describe three static algorithms for multicoloring, and prove bounds on their performance ratio.

2.1 A 1-local static algorithm with performance ratio 3/2

In this section, we show that a 1-local version of FPA described below has a performance ratio of 3/2.

The FPA-Local Algorithm
Local Information: The colors are divided into three palettes: the red colors are the colors 0 (mod 3), the blue colors are 1 (mod 3) and the green colors are 2 (mod 3). Each node v knows its base color (red, blue or green), its weight and its neighbors' weights.

(1) Let $D_1(v)$ be the 1-local maximal clique weight as computed by v.
(2) v constructs a local spectrum of size $3\lceil D_1(v)/2 \rceil$ equally split into red, blue and green colors as described above.
(3) If v's base color is

red: v uses the first $\lceil w(v)/2 \rceil$ colors from its red spectrum and the last $\lfloor w(v)/2 \rfloor$ colors from its blue spectrum.

blue: v uses the first $\lceil w(v)/2 \rceil$ colors from its blue spectrum and the last $\lfloor w(v)/2 \rfloor$ colors from its green spectrum.

green: v uses the first $\lceil w(v)/2 \rceil$ colors from its green spectrum and the last $\lfloor w(v)/2 \rfloor$ colors from its red spectrum.

Theorem 1 FPA-Local *is a* 1*-local approximation algorithm with a performance ratio of* 3/2*.*

Proof: It is straightforward to see that FPA-Local is a 1-local algorithm and uses at most $3/2 \max_v D_1(v) \leq 3/2\ \chi(G)$ colors. We next argue that two adjacent nodes can never assign common colors to their local demands. Without loss of generality, consider two such nodes, a red node v and a blue node u both of whom assign blue colors to their demands. From the protocol described above, it is clear that the nodes assign their blue colors from *opposite directions* of their respective local blue spectra. Since $w(v) + w(u) \leq \min\{D_1(v), D_1(u)\}$, it follows that the total number of blue colors used between u and v, $(\lfloor w(v)/2 \rfloor + \lceil w(u)/2 \rceil)$, is at most $\min\{\lceil D_1(v)/2 \rceil,\ \lceil D_1(u)/2 \rceil\}$; the latter quantity is the minimum among the sizes of local blue spectra at u and v. □

2.2 A 2-local static algorithm with performance ratio 17/12

In this section and in Section 2.3, we modify an offline algorithm discovered by Narayanan and Shende [10] (with a performance ratio of $\frac{4}{3}$) to obtain k-local algorithms (for values $k = 2$ and $k = 4$ respectively). Intuitively, the idea behind these algorithms is to exploit the regular geometry of the grid so that a node can either satisfy its local demand completely using a pre-allocated subset of colors, or can find sufficiently many colors by borrowing colors from neighboring nodes or using by colors from among a pre-allocated subset of auxiliary colors. There are two critical points that are worth re-emphasizing. First, the

color spectrum that is "seen" at a node can change over *time* as the weights of nodes in its locality change. Second, in our algorithms, neighboring nodes may have different local clique bounds and therefore, may see different color spectra. Yet, any sharing of colors must be accomplished without conflict in a distributed manner: once the weights in the locality are known, there is no need for critical sections for exclusive sequential access to the color spectra.

We start by assuming that each node is assigned a *base color* (red, green or blue). For technical reasons that will become clear later, we arbitrarily impose a *priority scheme* over the nodes: red nodes have priority over blue nodes which in turn have priority over green nodes. Consider an arbitrary system of three *directional axes* centered at some fixed origin node in the grid. With respect to one of the axial directions - say the horizontal axis - we assume that every node is pre-assigned a fixed *parity* in that direction so that in any straight line of nodes oriented parallel to the axis, nodes have alternating parities. A similar parity assignment can be fixed for each of the other two axial directions, *i.e.* at each node, three additional directional parity bits of information are pre-computed and stored in order.

The algorithm uses a color spectrum that is partitioned into *red, blue, green, purple,* and *yellow* palettes. The yellow colors are the colors (natural numbers) 16 mod 17. From the remaining colors, we assign the following palettes:

- Red colors are $0, 1, 2, 3 \bmod 17$.
- Blue colors are $4, 5, 6, 7 \bmod 17$.
- Green colors are $8, 9, 10, 11 \bmod 17$.
- Purple colors are $12, 13, 14, 15 \bmod 17$.

To avoid cumbersome notation, we use the term "neighbor" to denote a node in the 1-locality. At the beginning of each time step, appropriate messages are exchanged among the nodes so that every node v gets to know the current weights (number of calls) at each of the nodes in its 2-locality. Recall that $D_1(v) \leq D_2(v)$ are the respective maximum weights of cliques (triangles) in v's 1- and 2-localities; these two values can be locally computed by v knowing all the weights in its 2-locality.

Next, v categorizes nodes from *its local vantage point* according to the following criteria:

Definition 1

1. *Consider a node u in v's 1-locality. Then u is said to be v-light if $w(u)$ is at most $\lceil D_1(v)/3 \rceil$, v-heavy if $\lceil D_1(v)/3 \rceil < w(u) \leq 2\lceil D_1(v)/3 \rceil$ and v-superheavy otherwise. In particular, v is light (resp. heavy, superheavy) if it is v-light (resp. v-heavy, v-superheavy).*
2. *A node v is persistent if $w(v) > \lceil D_2(v)/3 \rceil$.*
3. *v is a* borrower *if it is heavy and at least two of its neighbors, both of the same base color, are v-heavy.*

4. *v is a* priority *borrower if it is a borrower, and either has no neighboring borrower or has priority (with respect to its base color) over its neighboring borrowers.*

5. *v is a* secondary *borrower if it is a borrower such that (a) it has exactly three neighbors all of whom are v-heavy and heavy, (b) exactly one of those neighbors is a priority borrower.*

It should be noted that v's categories for nodes in its 2-locality are purely *local* ones; its self-identification as being light, heavy or super-heavy may be different from what other nodes in its 2-locality may consider it to be. Moreover, v can categorize its neighbors (and itself) based only upon knowledge of weights in its 2-locality. For example, v's self-identification as a borrower depends only weights in its 1-locality. Since v has knowledge of the weights in its 2-locality, it can also identify its neighbors that are borrowers. Likewise, if v is a borrower, it can also determine which of its neighbors are priority borrowers. The next lemma describes some useful properties implied by Definition 1:

Lemma 2

1. *If v is superheavy, then all its neighbors are light.*

2. *Consider a triangle formed by three mutually adjacent nodes in the grid. Then at least one among these three nodes is light.*

3. *If v is heavy, then it has at most three* mutually non-adjacent *v-heavy neighbors.*

4. *The set of priority borrowers (and likewise, secondary borrowers) is an independent set in the graph. Furthermore, every borrower node is either a priority borrower or is adjacent to a priority borrower.*

Proof: We verify each assertion in the lemma in turn.

1. Suppose that v is superheavy but has a neighbor u that is not light. Then,

$$\begin{aligned} w(u) &> \lceil D_1(u)/3 \rceil \\ &\geq \lceil (w(u)+w(v))/3 \rceil, \qquad \text{and} \\ w(v) &> 2\lceil D_1(v)/3 \rceil \\ &\geq 2\lceil (w(u)+w(v))/3 \rceil. \end{aligned}$$

   This implies that $w(u)+w(v) > w(u)+w(v)$, an obvious contradiction.

2. Consider a triangle containing nodes s, t, and u with $D = w(s)+w(t)+w(u)$ being the weight of the triangle. By the pigeonhole principle, at least one node, say t, has weight less than or equal to $D/3$. Since D is a lower bound on $D_1(t)$, it follows that t is a light node.

3. Suppose instead that there is a triangle v, t, u, such that all three nodes are v-heavy. Then $w(v)+w(t)+w(u) > D_1(v)$ which contradicts the definition of $D_1(v)$.

4. Let u and v be two borrower nodes that are adjacent. Then exactly one of them is a priority borrower node, based on their base colors. This proves that the set of priority borrower nodes is an independent set, and also that each borrower node is either a priority borrower, or is adjacent to one. Finally, let u and v be blue and green (respectively) secondary borrower nodes that are adjacent. Then by assertion 2, and by the definition of secondary borrower nodes, u has three green neighbors which are all u-heavy, and of which one is a priority borrower. However, since blue nodes have priority over green ones, we obtain a contradiction. Thus the set of secondary borrowers is an independent set.

□

During the current time step as seen from v's vantage point, the local spectrum available for its use consists of the first $4\lceil\frac{D_2(v)}{12}\rceil$ red, blue, green and purple colors, and the first $\lceil\frac{D_2(v)}{12}\rceil$ yellow colors; in all, v "sees" $17\lceil\frac{D_2(v)}{12}\rceil$ distinct colors divided as above into the five palettes. For simplicity, we hereafter omit any mention of floors and ceilings from our calculations: this only affects the additive constant in our definition of competitiveness and has no significant bearing on our results. In particular, we assume that v has $D_2(v)/3$ colors in its local blue, green, red and purple palettes and $D_2(v)/12$ colors in its local yellow palette.

We describe the algorithm as proceeding in five sequential *phases* during each time step: note that this is just a technical device to make the proof of correctness of the algorithm clear. No communication with neighbors is required in between the phases, which are described below:

Phase 1: Each node uses as many initial colors as possible from the local palette corresponding to its color class, *i.e.* a blue node v uses the first $min\{w(v), D_2(v)/3\}$ blue colors from its palette. Notice that the base-color palette of a non-persistent node (with weight at most $D_2(\cdot)/3$) suffices to color it completely since there are $D_2(v)/3$ base color palette colors available at each node v. No conflicts can have occurred since the colors used by neighbors are from different palettes.

Phase 2: Each persistent node v has a remaining demand after the assignment of colors in the previous paragraph. We defer the coloring of heavy nodes to subsequent phases, but finish coloring superheavy nodes in this phase as detailed below. Without loss of generality consider a blue superheavy node v. We assign as many purple colors to v from the initial part of its local purple palette as needed. If $w(v) \leq 2D_2(v)/3$, then the first $w(v) - D_2(v)/3$ purple colors suffice to completely color v. If not, then v has used $D_2(v)/3$ blue colors and $D_2(v)/3$ purple colors, and has a remaining demand of $a(v) = w(v) - 2D_2(v)/3$. However, by assertion 1 of Lemma 2, all of v's neighbors are light nodes. Without loss of generality, let u be red neighbor of v of maximum weight over all the neighbors of v. Clearly, $w(v) + w(u) \leq D_1(v)$ and $w(u) \leq D_1(u)/3$ since u is a light node. Thus,

$$\begin{aligned} a(v) &= w(v) - 2D_2(v)/3 \\ &\leq w(v) - 2D_1(v)/3 \end{aligned}$$

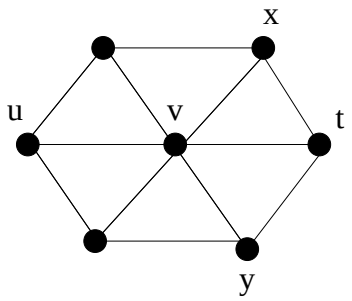

Figure 2: The vicinity of a secondary borrower v. The nodes u, x, y are all v-heavy as well as heavy, and one of them is a priority borrower.

$$\begin{aligned} &\leq D_1(v)/3 - w(u) \\ &\leq D_2(v)/3 - w(u) \end{aligned}$$

This implies that the *last* $a(v)$ colors from the red palette at v can be safely *borrowed* by v without causing color conflicts at any of its red neighbors. Thus, at the end of phase 2, we have a conflict-free assignment of colors to the non-persistent as well as persistent superheavy nodes, and a partial conflict-free assignment to the persistent heavy ones.

Phase 3: Among the persistent nodes, every *priority borrower* node v now borrows colors from a v-light neighbor's color class to finish coloring its calls; this is accomplished in a manner similar to that used for the superheavy nodes in Phase 2. Specifically, if $w(v) = D_2(v)/3 + a(v)$, then v borrows the *last* $a(v)$ colors from its v-light neighbors' base color palette. To see the validity of this step, let v be a red priority borrower. Since it has at least 2 blue v-heavy neighbors, assertion 3 of Lemma 2 implies that all three of v's green neighbors are v-light. Moreover, the color priority ensures that no blue neighbor of v can simultaneously be a priority borrower. Let s be a green neighbor of v with the largest weight among all its green neighbors. Then, there must exist a triangle $\Delta = \{s, t, v\}$ such that t is a v-heavy blue neighbor of v. Using the relations $w(\Delta) \leq D_1(v)$, $w(t) \geq D_1(v)/3$ and $w(s) \leq D_1(v)/3$, we obtain:

$$\begin{aligned} a(v) &= w(v) - D_2(v)/3 \\ &\leq w(v) - D_1(v)/3 \\ &= w(\Delta) - w(t) - w(s) - D_1(v)/3 \\ &\leq D_1(v)/3 - w(s) \\ &\leq D_2(v)/3 - w(s). \end{aligned}$$

We conclude that if v colors its additional $a(v)$ demands using the last a green colors from its local palette, then the resulting color assignment will not conflict on the green colors with the assignment at node s (and hence, at all other green neighbors of v).

Phase 4: We color all the persistent secondary borrower nodes in this phase. For ease of explanation, let v be a blue persistent secondary borrower that is adjacent to a red priority

borrower u. Figure 2 provides a snapshot of the nodes in the vicinity of v. By Definition 1.5, the other two red neighbors of v, namely x and y are v-heavy and heavy, and by assertions 2 and 3 of Lemma 2, the three green neighbors of v must be light as well as v-light. We also note that the red neighbors x and y are *not* borrower nodes. Let $w(v) = D_2(v)/3 + a(v)$. Then v assigns the first $min\{a(v), D_2(v)/12\}$ colors from its yellow palette. By assertion 4 of Lemma 2, v cannot be adjacent to another secondary borrower node, and thus this cannot cause any color conflict. If v's demand is satisfied at this stage, then v exits Phase 4.

Otherwise, there is some integer $b(v) > 0$ defined by

$$\begin{aligned} b(v) &= a(v) - D_2(v)/12 \\ &= w(v) - 5D_2(v)/12 \end{aligned} \tag{1}$$

that denotes the remaining demand of node v. Using Figure 2 to locate the red neighbors x, y and u of v, we summarize the situation at this stage. The node u has been completely colored using its local red palette and some borrowed green colors. In particular, the number of green colors borrowed by u is exactly $w(u) - D_2(u)/3$, and hence is at most $w(u) - D_1(v)/3$. All three green neighbors of v, including the node t, are light and hence, have been completely colored in Phase 1 after using an initial portion of their respective green palettes. For simplicity, let

$$d = \max\{w(x),\ w(y)\} - D_1(v)/3 \tag{2}$$

be an upper bound on the maximum remaining weight on x and y after Phase 1. The following case analysis is exhaustive and yields, in each case, a conflict-free coloring of v's remaining demand of $b(v)$:

Case 1: $b(v) \leq D_1(v)/3 - (w(t) + w(u) - D_1(v)/3)$.

Let w_{uv} be the maximum weight among the two common green neighbors of u and v shown in Figure 2. Then v assigns the colors $[\max\{w_{uv}, w(t)\} + 1,\ \max\{w_{uv}, w(t)\} + b(v)]$ from its green spectrum. Clearly v cannot conflict have a color conflict with any of its green neighbors. Further, since u has used at most $w(u) - D_1(v)/3$ green colors from the end of its green spectrum, it can be verified that there is no color conflict between u and v.

Case 2: $b(v) \leq D_1(v)/3 - 2d$, where d is defined by Equation 2.

In this case, v becomes the first node in its 1-locality to use purple colors by assigning to itself the colors $[d + 1, d + b(v)]$ from its local purple palette. Further, we note that the first d and the last d colors from the local purple palettes of nodes x and y are disjoint from the colors assigned by v in this phase. Hence, if x and y use purple colors in Phase 5 in future from either the beginning or the end of their respective purple palettes, no color conflict will occur over the purple colors.

Case 3: The inequalities

$$b(v) > D_1(v)/3 - (w(t) + w(u) - D_1(v)/3) \tag{3}$$

$$b(v) > D_1(v)/3 - 2d \tag{4}$$

hold simultaneously.
From Equation 1 and the fact that $w(u) + w(v) \leq D_1(v)$, we get:

$$w(u) + b(v) \quad \leq \quad 7D_1(v)/12 \tag{5}$$

Combining Equations 3 with 5 yields

$$w(t) \quad > \quad D_1(v)/12. \tag{6}$$

which in turn can be combined with Equation 4 to conclude that

$$w(t) + (d + D_1(v)/3) + b(v) + 5D_1(v)/12 \quad > \quad D_1(v) + b(v)/2.$$

Replacing d with its right hand side from Equation 2 and simplifying, we arrive at

$$w(t) + max\{w(x), w(y)\} + w(v) \quad > \quad D_1(v),$$

which contradicts the definition of $D_1(v)$.

We conclude, hence, that case 3 is impossible and that v can complete its color assignment by using $b(v)$ additional colors either from its green or purple spectra. Thus all secondary borrowers finish in this phase, and do not participate any further.

Phase 5: The only remaining nodes that do not yet have a complete color assignment are persistent nodes v which were neither priority nor secondary borrowers. We claim that any such node v can have at most two neighbors that could possibly participate in this phase, and if v has two such neighbors, then they must all be *along the same directional axis*. Furthermore, v can identify these potential participants using local information.

Let v be a node that has an incomplete color assignment after Phase 4. Observe that any neighbor u of v that is v-light cannot be persistent , and therefore received a complete assignment in Phase 1. If v was not a borrower, by assertion 3 of Lemma 2, v has at most 2 v-heavy neighbors along the same directional axis. Thus v identifies these as potential participants in this phase. If instead v was a borrower, it identifies all of its neighbors that were v-heavy and heavy but were *not* priority borrowers as potentially participating in this phase. Suppose that v has two neighbors u and w of the same base color class that are both v-heavy and heavy, and that were not priority borrowers. Since by assumption, v was not a priority borrower, it follows from assertion 4 of Lemma 2 that v is adjacent to a priority borrower. Since u and w are not priority borrowers, it must be that the remaining neighbor of v of the same color class as u and w was a priority borrower. This means that v was a secondary borrower, a contradiction.

This establishes that any node v with an incomplete color assignment may have at most two neighbors that may have incomplete assignments. Further, if v has two such neighbors, they must all be along the same directional axis, which v can compute locally based on its knowledge of weights in its 2-locality. Depending on the parity of v along this axis, v uses either the first or the last $w(v) - D_2(v)/3$ colors from its purple spectrum. Since any neighbor u of v along this axis has the opposite parity, there can be no color conflict in this phase. Finally, v cannot have conflicts with either secondary borrowers using purple colors in Phase 4, or superheavy nodes using purple colors, as explained earlier.

This completes the description of the NSA-Local algorithm. The discussion accompanying the algorithm description leads to the following theorem:

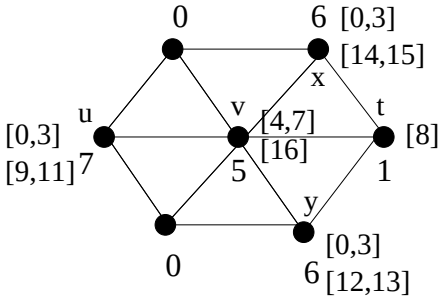

Figure 3: The number next to a node is its weight. The colors assigned to a node are given within brackets, assuming that $D_1(.) = D_2(.) = 12$ for v and its neighbors. The secondary borrower v cannot borrow either purple or green colors.

Theorem 2 NSA-Local *is a* 2*-local approximation algorithm with performance ratio* 17/12*.*

Remarks: The only nodes that use the yellow palette are the secondary borrower nodes in Phase 4. Before the phase, such nodes have already exhausted their base color palette, and may also borrow from green or purple neighbors during the phase. To increase the competitiveness of the algorithm, it is natural to examine if the borrowed green and purple colors would suffice to completely color the secondary borrower nodes. However, the example in Fig 3 shows that a blue secondary borrower node v with weight $5D_2(v)/12$ may have access to no purple or green colors and may still require $D_2(v)/12$ colors. Let u be the red priority borrower neighbor of v, and x and y the remaining heavy as well as v-heavy neighbors of v. It is easy to see from the figure that $D_1(v) = 12$, and it is possible to extend the graph so that the $D_2(.)$ clique bounds of all nodes in v's 1-locality are also at most 12, so that base color and purple palettes of all nodes in v's 1-locality are of size 4 or less. Thus, u being a priority borrower would borrow the last three green colors, and t would use the first green color, so that v cannot borrow any green colors without conflict. Furthermore, x and y would borrow 2 colors each from the purple spectrum but since they may use colors from different ends of the spectrum, as shown, v cannot borrow any purple colors either. Yet, v has a remaining demand of 1. Furthermore, the situation can be amplified by multiplying the demands on every vertex by any constant c.

2.3 A 4-local static algorithm with performance ratio 4/3

A further modification allows us to convert the offline algorithm that formed the basis of NSA-Local to a 4-local algorithm with an even better competitive ratio. In essence, we eliminate the local yellow palette used at a node in the algorithm NSA-Local by allowing each node to access its 4-locality. In particular, in the example shown in Figure 3, if the light node t had used the last green colors instead of the first ones, then the number of green colors available for v to borrow would suffice to completely color v. As in [10], we will show that this will not affect any of the neighbors of such a light node t, except perhaps a super-heavy node, which may then have to follow a different strategy from the one in the NSA-Local algorithm. Our argument differs from the proof of the algorithm in [10] in that nodes must identify themselves as light, borrowers and so on, based on local

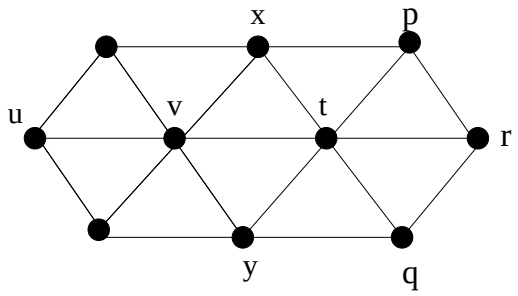

Figure 4: The vicinity of a special light node t. The node v is a persistent secondary borrower and u is a priority borrower. The nodes x and y are persistent but not priority borrowers.

information, and not on global knowledge of weights in the entire graph. We will show that it is possible for a light node to determine when to use colors from the end of the spectrum, with knowledge of only its 3-locality. Correspondingly, a super-heavy node needs knowledge of its 4-locality to anticipate such a choice made by one or more of its light neighbors. We proceed to show how such a correct (conflict-free) protocol can be derived.

As in the NSA-Local algorithm, each node determines if it is heavy, a borrower, a priority or a secondary borrower. Recall that we call a node v persistent if $w(v) > D_2(v)/3$. It is easy to see that a node v may determine if v itself is persistent based on its 2-locality, and if a neighbor of v is persistent based on v's 3-locality. Before proceeding to determine its color assignment, each node makes the following additional identification based on knowledge of its 3-locality. Figure 4 may be helpful in understanding the following definition, and will be used as a reference throughout this section.

Definition 2 *A node t is a* special light node *if it is a light node adjacent to a* persistent *secondary borrower node v with the following properties:*

1. *t is not adjacent to the priority borrower neighbor u of v.*
2. *t is adjacent to the two remaining v-heavy neighbors of v, and these two neighbors are persistent nodes.*
3. $w(u) + w(v) + w(t) > D_1(v)$

It is easy to verify that a node may identify itself as being a special light node using only its knowledge of its 3-locality. We prove the following properties of special light nodes for later use:

Lemma 3 *A special light node t can be adjacent to at most one persistent secondary borrower node. Furthermore, if t is adjacent to a persistent priority borrower node with two persistent neighbors, then they are of the same color class as t.*

Proof: Let t be a special light node adjacent to a persistent secondary borrower node v with properties listed in Definition 2. For ease of explanation, we refer to Figure 4. From

the definition, x and y must be persistent , but since they cannot be borrowers, p and q must be x-light and y-light respectively. In other words, p and q are not persistent . Thus, none of p, q, and r can be persistent secondary borrowers. Finally, if r is a persistent priority node with two persistent neighbors, they must be of the same color class as t, finishing the proof of the lemma. □

Phase 1: Each node v that is not a special light node assigns itself the first $min\{w(v), D_2(v)/3\}$ colors from its base color palette. Let t be a special light node adjacent to a persistent secondary borrower node v. From Lemma 3, this neighbor of t is unique. Let w_{uv} be the maximum weight on the common neighbors of v and the priority borrower u adjacent to v. Then t uses the first w_{uv} colors, and the last $w(t) - w_{uv}$ colors from its base color palette.

It is easy to see that all the computations done in this phase are 3-local. After this phase, the only nodes that remain to be colored are the persistent nodes, and only these participate further.

Phase 2: In this phase, we color all the super-heavy nodes. Without loss of generality, consider a super-heavy node v with base color red. All its neighbors must be light by Lemma 2, and in fact, they are v-light as well. First v assigns $\max\{w(v) - D_2(v)/3, D_2(v)/3\}$ purple colors to v thus disposing of the super-heavy nodes with weight $\leq 2D_2(v)/3$. If the residual weight $\delta = w(v) - 2D_2(v)/3 > 0$, then we need to find δ additional colors to finish coloring v. We will demonstrate that δ colors can be borrowed by v from either the blue or the green palettes, keeping in mind that that some of v's light neighbors may be special light nodes.

Let $b_3 \leq b_2 \leq b_1 \leq D_1(v)/3$ and $g_3 \leq g_2 \leq g_1 \leq D_1(v)/3$ be the weights of v's blue and green neighbors in G respectively. The number of colors available to v from the blue palette is at least $D_1(v)/3 - (b_1 + b_2)$ and similarly there are at least $D_1(v)/3 - (g_1 + g_2)$ green colors available to g. Thus a total of $2D_1(v)/3 - (b_1 + b_2 + g_1 + g_2)$ colors are available to v to borrow without conflict. Notice that the node with weight g_1 must be adjacent to either the node with weight b_1 or b_2. In either case, $\delta \leq D_1(v)/3 - (b_2 + g_1)$. A similar argument shows that $\delta \leq D_1(v)/3 - (g_2 + b_1)$. Adding the two, we obtain:

$$2\delta \leq 2D_1(v)/3 - (b_1 + b_2 + g_1 + g_2)$$

Since v needs δ colors and the right hand side is equal to the number of colors available to v, consequently v can be colored completely by borrowing either blue or green colors. Since v has access to its 4-locality, it can determine which of its neighbors are special light nodes, and exactly what colors they use, and thus choose a conflict-free set of colors to borrow, completing its color assignment.

Phase 3: Each persistent priority borrower v now determines how many of its v-heavy neighbors are persistent . If it has at least 2 persistent neighbors, then it borrows the remaining colors from the end of the light neighbors' base color palette. It follows from Lemma 3, if v is adjacent to a special-light node t, then it does not borrow from t's base color palette, and thus can have no conflict with t. The argument that v can find sufficient

colors to borrow without conflict is the same as in Phase 3 of the NSA-Local algorithm, and we do not repeat it here.

It is easy to see that all computations done in this phase are 3-local. After this phase, all persistent priority borrowers with at least 2 persistent neighbors have a complete assignment.

Phase 4: Let v be a persistent secondary borrower node with a priority borrower neighbor u and two v-heavy as well as heavy neighbors x and y, as in Figure 4. If x and y are both persistent , then v borrows $w(v) - D_2(v)/3$ colors from t's base color palette, starting with the color $w_{uv}+1$ where w_{uv} is the maximum weight on the common neighbors of u and v. As in the NSA-Local algorithm, v cannot have a conflict with u or with the common neighbors of u and v. If $w(t) < w_{uv}$ then t cannot have a conflict with v. If instead $w(t) > w_{uv}$ then t uses only the first w_{uv} colors and the remaining colors from the end of its palette. Since x is v-heavy,

$$w(t) + w(v) \leq 2D_1(v)/3$$

which implies that

$$\begin{aligned} w(t) + w(v) - D_2(v)/3 &\leq w(t) + w(v) - D_1(v)/3 \\ &\leq D_1(v)/3 \end{aligned}$$

Thus v and t cannot have any color conflicts.

It is easy to see that all computations done in this phase are 3-local. After this phase, all persistent secondary borrowers with at least 2 persistent neighbors have a complete conflict-free color assignment.

Phase 5: Each node that has an incomplete color assignment that is neither a priority nor a secondary borrower performs Phase 5 exactly as in the NSA-Local algorithm. Since the only nodes to have previously used purple colors were super-heavy nodes, which by Lemma 2 can have only light neighbors, the analysis of this phase is identical to that in the NSA-Local algorithm. It follows that all computations in this phase are 2-local.

Phase 6: The only nodes that remain for this phase are solid priority or secondary borrowers with at most one persistent neighbor. If such a node v has no persistent neighbors, it uses $w(v) - D_2(v)/3$ colors from the purple palette without causing conflicts. If instead a persistent priority borrower v has a persistent secondary borrower neighbor u, then neither u nor v has any other neighbors, and they can use the parity scheme to determine which end of the purple spectrum to use colors from. Finally, if v has a persistent neighbor u that is neither a priority nor a secondary borrower, then v can determine the purple colors used by u using u's 2-locality, and assign itself colors from the opposite end of its purple spectrum without color conflict. Thus, all computations done in this phase are 3-local computations.

The analysis above leads to the the following theorem:

Theorem 3 NSB-Local *is a* 4*-local approximation algorithm with performance ratio* 4/3.

2.4 From static to online algorithms

We note that for each of the recoloring algorithms we described, the color assignment is decided by each vertex v solely on the basis of pre-computed information like base color classes, and the weights of nodes in v's k-locality. For example, in the algorithm NSA-Local , if the weights in the 2-locality of a vertex v in steps t and $t+1$ remain the same, then the color assignment computed by v in these steps is identical, and v will not recolor any calls. Thus, NSA-Local has recoloring distance 2.

As a corollary to Theorems 1, 2, 3, Lemma 1 and the discussion about *FA-Local* at the beginning of the section, we obtain the following result:

Corollary 1 *There is a distributed online algorithm for multicoloring that is:*

1. 0*-local, 3-competitive, and non-recoloring,*
2. 1*-local and 1.5-competitive, with recoloring distance 1,*
3. 2*-local and* $\frac{17}{12}$*-competitive, with recoloring distance 2, and*
4. 4*-local and* $\frac{4}{3}$*-competitive, with recoloring distance 4.*

3 Lower bounds

In this section, we show lower bounds on the competitive ratio of any online algorithms for multicoloring interference graphs. We consider first algorithms with recoloring distance k, and then algorithms that do not allow recoloring. We also show lower bounds on algorithms that are k-view and color class determined.

3.1 Online recoloring algorithms

We first prove a technical lemma that aids the proof of the lower bound.

Lemma 4 *Let P be a path of length ℓ, with weight n on each of its $\ell+1$ nodes. Then the minimal number of colors required to color P such that the end nodes have exactly α colors in common, is at least $2n + \frac{2\alpha}{\ell-1}$ when ℓ is odd, and at least $2n + \frac{2(n-\alpha)}{\ell}$ when ℓ is even.*

Proof: Let P, n and ℓ be as in the statement of the lemma. Let u and v be the end nodes of P, and let u' and v' be the neighbors of u and v, respectively (See Figure 5a). We first construct P' from P as follows. Node u is split into two connected nodes, u_1 and u_2, which are assigned weight α and $n-\alpha$, respectively. Similarly, node v is split into the connected nodes v_1 and v_2, with weight α and $n-\alpha$, respectively (Figure 5b). Obviously, coloring P such that u and v have exactly α colors in common is equivalent to coloring P' such that u_1 and v_1 receive the same colors, and u_2 and v_2 receive completely different colors. Next, we construct graph P'' from P' as follows. Nodes u_1 and v_1 are identified into one node uv_1, and nodes u_2 and v_2 are joined by an edge (Figure 5c). It is easy to see that any coloring of P'' is equivalent to a coloring of P' in which u_1 and u_2 receive the same colors, and u_2 and v_2 receive different colors, which in turn is equivalent to a coloring of P as required by the lemma.

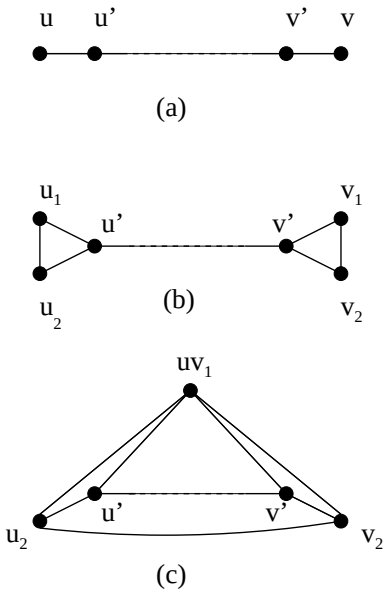

Figure 5: Multicoloring a path where the endpoints have colors in common

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

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

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

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

□

Next we show a lower bound on any online algorithm with recoloring distance k.

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

Proof: Fix $k \geq 0$. We will exhibit a strategy for the adversary which forces the algorithm to use at least $2n + \frac{n}{2(k+1)}$ colors, while the offline algorithm never needs more than $2n$ colors. The graph used by the adversary is shown in Fig 6. Let u and v be two nodes at distance 3 of each other along one of the axis of the grid. The adversary starts by raising the weight

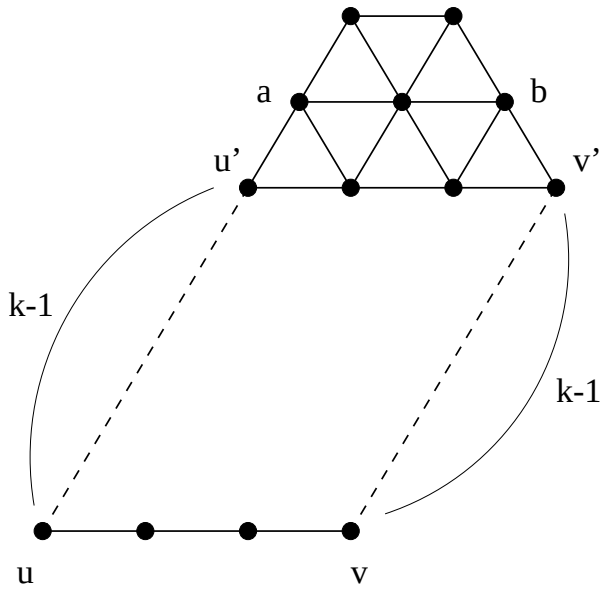

Figure 6: Graph used by adversary to show a lower bound on any online algorithm with recoloring distance k.

on u and v to n. If $k > 0$, the adversary continues to raise the weight to n on all nodes along two parallel axes of length $k-1$ which make an angle of $\frac{\pi}{3}$ with the axis uv, and which start at u and v, respectively. The algorithm may color and recolor as desired.

Let u' and v' be the last nodes of the axis growing out of u and v, respectively, on which the weight has been raised. Nodes u' and v' have distance $k-1$ from u and v, respectively. Next, the adversary raises the weight to n on two nodes, a and b, situated as follows. Node a is a neighbor of u', situated along the axis uu', at distance k from u. Node b is a neighbor of v' and lies at distance k from v, but is situated at an angle $\frac{\pi}{3}$ from the axis vv', and thus lies at distance 2 from node a (See Fig 6).

The next moves of the adversary will only involve nodes at distance greater than k from u and v, so the colors on u and v are now fixed. Let α be the number of colors that u and v have in common. The strategy of the adversary now depends on α. If $\alpha \geq n/2$, then the adversary raises the weight to n on the two nodes which lie on a path of length 3 between a and b and which have distance greater than k to both u and v. The nodes of positive weight now lie on a path of length $2k+3$. By Lemma 4 the algorithm must now use at least $2n + \frac{n}{2k+2}$ colors. If $\alpha < n/2$, the adversary raises the weight of the common neighbor of a and b to n. By Lemma 4, the algorithm will have to use at least $2n + \frac{n}{2(k+1)}$ colors. The number of colors needed by the off-line algorithm is $2n$. The above construction can be repeated as many times as desired, proving that there are infinitely long sequences on which the ratio bound of $1 + \frac{1}{4(k+1)}$ is achieved. □

For the special case $k = 0$, we can prove a better bound:

Theorem 5 *Any online algorithm with recoloring distance* 0*, acting on a triangular grid of diameter at least* 3*, has competitive ratio at least* $1 + 2/7$.

Proof: The adversary raises the weight to n on three nodes that all lie at distance 3 from each other. Such nodes exist: take every three nodes of an induced 9-cycle. The algorithm uses n colors on each of these nodes. Now let $\alpha_1 n$, $\alpha_2 n$ and $\alpha_3 n$ denote the number of colors that each pair of these three nodes have in common. If $(\alpha_1 + \alpha_2 + \alpha_3) < 12/7$, then the number of colors that the algorithm has used is at least $3n - (\alpha_1 + \alpha_2 + \alpha_3)n > 9n/7$. Since the offline algorithm could have colored all three nodes with the same n colors, triangle is n, this means that the adversary has succeeded.

If $(\alpha_1 + \alpha_2 + \alpha_3) \geq 12/7$, then one of the α_i must be at least $4/7$. Let u and v denote those two nodes that have at least $4n/7$ colors in common. Now the adversary raises the weight to n on the two nodes on the path of length 3 that connects u and v. The result is a path of length 3. The nodes u and v cannot be recolored since the recoloring distance is 0. By Lemma 4, the algorithm now must use at least $2n + \frac{2(4/7n)}{2} = 18n/7$ colors. It is easy to see that the offline algorithm could have colored the sequence with $2n$ colors, thus yielding a competitive ratio of at least $9/7$ for the algorithm. □

3.2 Online non-recoloring algorithms

In this section, we show a lower bound on the competitive ratio for any non-recoloring online algorithm. For such algorithms, the adversary can specify which colors the algorithm should drop, by which it can force the remaining colors to stay. Thus we are able to obtain stronger lower bounds for non-recoloring algorithms. We create a graph and a sequence of requests such that the offline algorithm could always color the graph using n colors, but any non-recoloring online algorithm is forced to use $2n$ colors, yielding the following theorem:

Theorem 6 *Any non-recoloring online algorithm has competitive ratio at least* 2.

Proof: The graph we use is given in Figure 7. We assume that initially, every node has weight 0. The adversary proceeds in several steps which are described below. In each step, the adversary can increase the weight at any node, as well as drop any subset of colors already at a node. We use the notation $C(v)$ to denote the set of colors currently used at node v. It is straightforward to verify that a coloring using n colors is possible at every step.

1. First the adversary creates a situation where there are two nodes with exactly same set of colors. To do this, the adversary raises the weight to n at nodes a, d, and k. It is straightforward to argue that either the algorithm has used $2n$ colors in all at this point, or at least two of the three nodes have $n/3$ colors in common. Without loss of generality, let a and d have at least $n/3$ colors in common. The adversary then drops $2n/3$ colors at a and d, keeping $n/3$ of the common colors at both nodes.

2. The adversary raises the weight to n at nodes f and i. Since at most $n/3$ colors can be re-used, the algorithm has to use at least $2n/3$ new colors at both these nodes. The adversary now drops $2n/3$ colors at f and i in such a way that the remaining colors at these nodes are disjoint sets, and additionally have no colors in common with the colors at a and d.

   At this point, the online algorithm has used n colors.

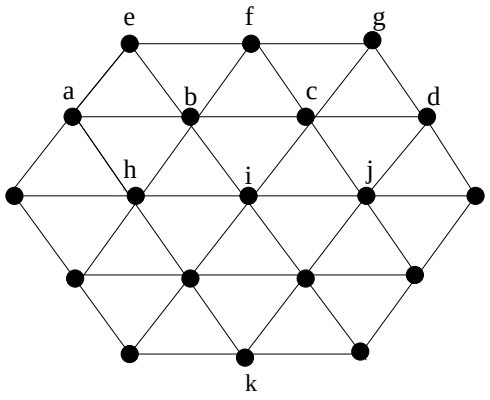

Figure 7: Graph used to prove a lower bound for non-recoloring algorithms

3. The adversary raises the weight to $2n/3$ at nodes b, g, and j. Since each of these nodes is adjacent to a node with the color set $C(a)$ and to at least one of f and i; at most $n/3$ colors can be reused, and at least $n/3$ new colors have to be used at each of the three nodes. The adversary now drops $2n/9$ colors at each of these nodes, in such as a way as to keep $n/3$ new colors in all.

   At this point, the online algorithm has used $n + n/3$ colors,

4. The adversary raises the weight to $5n/9$ at e, c, and h. Since all of the $n + n/3$ colors used by the algorithm at this point are present at one of $\{b, i, j, d, g, f\}$, the algorithm must use $5n/9$ new colors to color c.

   Let δ be the total number of new colors used by the algorithm at e and h. Then the online algorithm has used $n + 8n/9 + \delta$ colors.

5. If $\delta \geq n/9$, then the adversary has succeeded. This is because the online algorithm has used at least $2n$ colors, while the offline algorithm would have needed only n colors. If instead $\delta < n/9$, then the algorithm must have reused old colors. The adversary's strategy now is, based on the colors used by the algorithm at the nodes e and h, to drop certain colors from some subset of the nodes $\{e, h, f, i, c\}$ and raise the weight at b in such a way as to force the algorithm to use at least an additional $n/9 - \delta$ colors, completing the proof. The weights at all nodes in the graph at this point are given in Figure 8.

   Consider the set of colors used by the algorithm at e; it may have non-empty intersections with $C(g), C(j), C(i)$, and $C(c)$. Let $|C(e) \cap C(i)| = \beta_1$ and $|C(h) \cap C(f)| = \beta_2$ Without loss of generality, let $\beta_2 \geq \beta_1$. Notice that the only colors that can be reused at b are those at g or j, which have $2n/9$ colors in all. We first consider the following three cases when $|(C(e) \cup C(h)) \cap (C(g) \cup C(j))| \geq n/9$;. that is, at least $n/9$ of the colors at g and j are also present at e or h.

   Case 1: $\beta_1 \geq 2n/9$.

   The adversary chooses the following three sets:

   - a set $X \subseteq (C(e) \cup C(h)) \cap (C(g) \cup C(j))$ of size $n/9$.

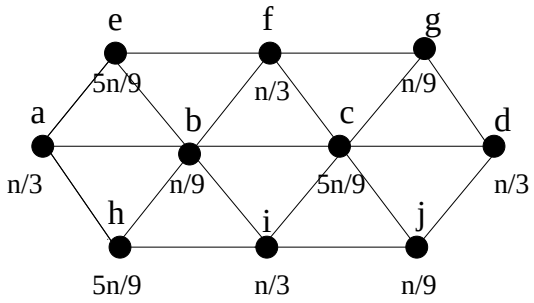

Figure 8: Weights on nodes after Step 4.

- a set $Y \subseteq (C(e) \cap C(i))$ of size $2n/9$.
- a set $Z \subseteq (C(h) \cap C(f)$ of size $2n/9$.

The adversary then

- keeps the sets X and Y at e and drops everything else.
- keeps the sets X and Z at h and drops everything else.
- drops the set Y from i and the set Z from f.
- raises the weight by $2n/9$ at b.

Since b's 1-locality still contains all the colors previously used at b, a, f, i, c, and exactly $n/9$ of the colors from $C(g) \cup C(j)$ are guaranteed to be in b's 1-locality, at most $n/9$ colors can be reused at b. Thus the algorithm has to use at least $n/9$ new colors.

Case 2: $n/9 \leq \beta_1 < 2n/9$.

In this case $|C(e) \cap C(c)| \geq n/9 - \delta$. The adversary chooses the following sets:

- a set $X \subseteq (C(e) \cup C(h)) \cap (C(g) \cup C(j))$ of size $n/9$.
- a set $Y \subseteq (C(e) \cap C(i))$ of size $n/9$.
- a set $Z \subseteq (C(h) \cap C(f)$ of size $n/9$.
- a set $U \subseteq C(e) \cap C(c)$ of size $n/9 - \delta$.

The adversary then

- keeps the new colors (at most δ of them), the sets X, Y, and U at e and drops everything else. Thus e now has at most $\delta + n/9 + n/9 + n/9 - \delta \leq n/3$ colors
- keeps the new colors (at most δ of them), the sets X and Z at h and drops everything else.
- drops the set Y from i and the set Z from f.
- drops the set U from c.
- raises the weight by $2n/9 - \delta$ at b.

As in Case 1, at most $n/9$ colors can be reused at b. Thus the algorithm has to use at least $n/9 - \delta$ new colors.

Case 3: $\beta_1 < n/9$

In this case $|C(e) \cap C(c)| \geq 2n/9 - \delta$. The adversary chooses the following sets:

- a set $X \subseteq (C(e) \cup C(h)) \cap (C(g) \cup C(j))$ of size $n/9$.
- a set $U \subseteq (C(e) \cup C(h)) \cap C(c)$ of size $2n/9 - \delta$.

The adversary then

- keeps the new colors (at most δ of them), the sets X and U at e and drops everything else. Thus e now has at most $\delta + n/9 + 2n/9 - \delta \leq n/3$ colors
- keeps the new colors (at most δ of them), the and the set X at h and drops everything else.
- drops the set U from c.
- raises the weight by $2n/9 - \delta$ at b.

As in Case 1, at most $n/9$ colors can be reused at b. Thus the algorithm has to use at least $n/9 - \delta$ new colors.

The case when $|(C(e) \cup C(h)) \cap (C(g) \cup C(j))| < n/9$ is handled in an almost identical manner, concluding the proof. □

3.3 k-view and color class determined algorithms

In this section, we consider the restricted class of k-view and color class determined algorithms. We show lower bounds on both recoloring and non-recoloring algorithms in this class.

Theorem 7 *For all $k \geq 0$, any recoloring k-view and color class determined algorithm has competitive ratio at least 3/2.*

Proof: Fix $k \geq 0$ and a k-view and color class determined algorithm. We first describe the strategy used by the adversary on the infinite lattice. The adversary simply raises the weight of all nodes along a horizontal axis to n. Since the algorithm is k-view and color class determined, and the k-views of all red node are the same, every red node must be colored by the same set of colors, say R, and similarly every blue node is colored by a set B and every green node by a set G. Furthermore, since every red node is adjacent to a blue and green node, and similarly, every blue node is adjacent to a green and a red node, the sets B, G, and R must be completely disjoint. Thus the algorithm has used $3n$ colors, where the off-line algorithm needed only $2n$ colors.

Now consider a finite induced subgraph of the infinite triangular lattice containing all nodes within distance $k+2$ of some arbitrary fixed origin. It is not difficult to see that the algorithm will color the three nodes in the 1-locality of the origin in the same way as in the

infinite case, thus forcing the algorithm to use $3n$ colors. □

For non-recoloring k-view and color class determined algorithms (for any $k > 0$), we are able to show much stronger lower bounds.

Theorem 8 *For any constant $\epsilon > 0, k \geq 0$, any non-recoloring k-view and color class determined algorithm has competitive ratio at least $3 - \epsilon$.*

Proof: For ease of exposition, we first describe the argument on an infinite triangular lattice. Consider a base 3-coloring of the infinite triangular lattice with all nodes colored red, blue, or green. Further, we can divide all the red nodes into 3 different classes such that the three red neighbors of any blue or green node belong to 3 different classes. Blue and green nodes can also be divided into 3 classes in a similar way.

Fix $k \geq 0, \epsilon > 0$ and a non-recoloring k-view and color class determined algorithm. The adversary now raises weights and drops colors in such a way that the algorithm is forced to use $(3 - \epsilon)n$ colors while it is easy to verify that the offline algorithm could perform the coloring using n colors.

In the first step, the adversary raises the weight to n at all the red nodes. Since the algorithm is k-view and color class determined, and since for any $k \geq 0$, the k-view of any red node is identical, the algorithm uses the same colors at all the red nodes. In the next step the adversary drops $2n/3$ colors at each red node in such a way that the $n/3$ colors at two red nodes of two different classes are completely distinct. This means that any blue node cannot reuse any of the n colors used by the algorithm so far. In the third step, the adversary raises the weight to $n - n/3 = 2n/3$ at each blue node. As before, the algorithm must use the same new set of $2n/3$ colors at all the blue nodes. In the next step, the adversary drops $4n/9$ colors at each blue node in such a way that blue nodes belonging to different classes have completely different colors. In the fifth step, the adversary raises the weigh to $4n/9$ at every green node. Once again, the algorithm must use the the same new set of $4n/9$ colors at all the green nodes. In the next step the adversary drops $8n/27$ colors at each green node in such a way that green nodes belonging to different classes have completely different colors.

At this point the algorithm has used at least $n + 2n/3 + 4n/9$ colors. Every red node has a weight of $n/3$, every blue node a weight of $2n/9$ and every green node a weight of $4n/27$, adding up to a total of weight $19n/27$ on any triangle. The adversary now repeats an identical strategy, raising the weight by $n - 19n/27 = 8n/27$ at all red nodes and continuing on. Thus in the next six steps of raising weights and dropping colors at red, blue and green nodes, an additional $8n/27(1 + 2/3 + 4/9)$ colors are used. After $2j$ steps, the algorithm has been forced to use $n\Sigma_{i=0}^{j}(2/3)^i$ colors.

To force the algorithm to use more than $(3 - \epsilon)n$ colors, the adversary uses a sequence of $m = 2\lceil \log_{\frac{3}{2}} \frac{2}{\epsilon} \rceil$ steps. This proves that in the infinite lattice, any k-view and color class determined algorithm has competitive ratio at least $3 - \epsilon$.

Now consider a finite induced subgraph of the infinite triangular lattice containing all nodes within distance $mk + 1$ of some arbitrary fixed origin. A straightforward inductive argument implies that after i steps, all nodes within distance $(m - i)k + 1$ of the origin behave exactly as in the infinite case. Thus after m steps, the algorithm has used the same

colors in the 1-locality of the origin as in the infinite case, proving the result. □

Theorems 7 and 8 show that FPA-Local and FA-Local are optimal for the class of view and color class determined algorithms that are allowed to recolor and disallowed from doing so respectively.

4 Conclusions and open questions

In this paper, we developed a framework for studying distributed online frequency assignment in cellular networks. We exhibited the first distributed online algorithms for this problem with proven bounds on their competitive ratios. In particular, we showed 0, 1, 2, and 4-local algorithms with competitive ratios of $3, 3/2, 17/12$, and $4/3$ respectively. In terms of lower bounds, we showed that any online algorithm with recoloring distance k has competitive ratio at least $\frac{1}{4(k+1)}$. For non-recoloring and k-view and color class determined algorithms, we were able to prove better lower bounds. Our results show that FPA-Local and *FA-Local* are optimal for the class of k-view and color class determined algorithms that are allowed to recolor and forbidden to recolor respectively. The algorithms we described are distributed algorithms, and all except FA-Local, are synchronous. FA-Local is a 0-local algorithm and requires no communication between nodes at all. All the other algorithms we described require each node to communicate at most a constant number of messages, and $O(\log max_{v,t} w_t(v))$ bits, with a constant number of other nodes in the network between steps.

While we showed tight bounds on the competitive ratio of any k-view and color class determined algorithm (recoloring and non-recoloring), we were unable to prove such tight results in the absence of restrictions on the algorithm. Even in the static case, the best known approximation algorithms for the problem have performance ratio $4/3$. While we derived a 4-local algorithm with the same competitive ratio, it would be interesting to know if the same ratio can be achieved by a 0-local or a 1-local algorithm.

Another natural requirement on online algorithms might be to restrict the *amount* of recoloring at any vertex to be proportional to the change in demand in the vertex's k-locality in every step. Among the algorithms we describe, FPA-Local meets this requirement, but the NSA-Local and NSB-Local algorithms do not. Also, in practice, the available frequency spectrum is a contiguous linear sub-interval of the radio spectrum, and frequency reuse is controlled by a sequence of non-negative integers, $c_0 \geq c_1 \ldots$, with $c_0 \geq 1$, called *distance reuse constraints*, where any two frequencies assigned to nodes that are distance i apart in the graph are required to differ by at least c_i. This paper assumes $c_0 = c_1 = 1$ and $c_i = 0$ for $i > 1$; tight bounds for the more general case would be interesting. Finally, some of our lower bounds are for deterministic algorithms. Investigating the use of randomization in the design of efficient online algorithms would be an interesting avenue of further research.

Acknowledgements

We are grateful to the anonymous referees for several comments and suggestions that improved the presentation of this paper.

References

[1] 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.

[2] S. Irani. Coloring inductive graphs online. *Algorithmica*, 11(1):53–72, 1994.

[3] J. Janssen and K. Kilakos. Adaptive multicolourings. *Combinatorica*. To Appear.

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

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

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

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

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

[10] L. Narayanan and S. Shende. Static frequency assignment in cellular networks. In *In Proceedings of SIROCCO 97*, pages 215–227. Carleton Scientific Press, 1997. To appear in *Algorithmica*.

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

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

[13] W. Wang and C. Rushforth. Local search for channel assignment in cellular mobile networks. In *Satisfiability Problem: Theory and Applications*, volume 35 of *DIMACS Series in Math. and Theoretical Comp.Sc.* American Mathematical Society, 1997.