Worst-case analysis of a dynamic channel assignment strategy [*]

Lata Narayanan and Yihui Tang
Department of Computer Science
Concordia University
Montreal, Quebec
Canada H3G 1M8
email { lata, tang } @cs.concordia.ca

August 11, 2003

Abstract

We consider the problem of channel assignment in cellular networks with arbitrary reuse distance. We show upper and lower bounds for the competitive ratio of a previously proposed and widely studied version of dynamic channel assignment, which we refer to as *the greedy algorithm.* We study two versions of this algorithm: one that performs reassignment of channels, and one that never reassigns channels to calls. For reuse distance 2, we show tight bounds on the competitive ratio of both versions of the algorithm. For reuse distance 3, we show non-trivial lower bounds for both versions of the algorithm.

1 Introduction

The demand for wireless telephony and wireless data services is expected to continue growing dramatically over the next decade. This makes the efficient use of already scarce spectrum resources of great importance. The key idea behind the *cellular* concept was that of *frequency reuse*: by dividing the service area into small coverage areas called *cells* served by low-power transmitters, one could reuse the same channels in different cells in the network, thereby greatly increasing the capacity of the network. However, in practice, reuse is limited by the phenomenon of *co-channel interference*: if the same channel is used in two different cells that are geographically close to each other, there can be radio interference, which distorts the signals. To achieve an acceptable signal to interference ratio, the same channel should therefore not be reused in two different cells in the network, unless they are separated by a minimum distance, which is called the *reuse distance.*

Cellular data and communication networks are usually modeled as graphs with each node representing a base station in 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 networks employing TDMA/FDMA technology, service involves the assignment of a frequency channel to each client call. For convenience, the set of available channels is assumed to be the set $\{1, 2, \ldots, C\}$. The graphs most often used to model cellular networks are finite portions

[*]Research supported partially by NSERC, Canada.

of the infinite triangular lattice. We refer to a finite induced subgraph of the triangular lattice as a *hexagon graph*. In the rest of this paper, the graphs we consider are always hexagon graphs. The number of calls to be served, that is, channels to be assigned, at a node v is modeled by its weight $w(v)$. Interference constraints are modeled in terms of reuse distance; it is assumed that the same channel can be assigned to two different nodes in the graph if and only if their graph distance is at least r. The objective of an algorithm for the *channel assignment problem* is to assign $w(v)$ channels to each node v in the network, such that interference constraints are respected, and the total number of channels used over all the nodes in the network is minimized. In practice, the problem is an *online* one; at each step, each node has a weight, and the algorithm must assign channels to all new calls, and perhaps adjust the previous assignment by reassigning channels to old calls.

A simple and commonly used strategy is called *fixed channel assignment (FCA)*, in which base stations can only use channels from fixed sets that are precomputed to avoid interference with neighbors [11]. For example, when the reuse distance is 2, a 3-cell pattern is used to cover the network, and the set of available channels is partitioned into three equal sized sets. Each cell uses channels from only one of the three sets based on which cell it corresponds to in the pattern. Similarly, for the case $r = 3$, a 7-cell pattern can be used. The problem with such a fixed strategy is that a lot of channels may be unused in cells with low traffic, while calls are rejected in cells with higher traffic, resulting in inefficient use of available bandwidth. In fact, this strategy can be seen to use 3 times (7 times) the minimum number of channels needed when the reuse pattern is a 3-cell pattern (7-cell pattern respectively). In contrast, *dynamic channel assignment (DCA)* [2, 3, 5, 15, 19] does not partition the channel set, and in principle, allows any channel to be used by any node in the network. In other words, each base station can use any channels that are currently unused by any neighbors with whom there might be a possibility of interference. In between are *borrowing* strategies [4, 6, 7, 12, 14], where a fixed number of channels is reserved per node, but borrowing is allowed, provided there is no conflict with neighbors. This can also be seen as providing a different ordering of channels to different nodes. An interesting alternative called *cluster partitioning* was proposed in [9], the details of which are provided in Section 5.2. While it is repeatedly claimed that DCA achieves higher capacity than FCA, and this has been demonstrated in some situations, using both empirical and analytical methods, its worst case performance in a theoretical framework has not been studied.

Katzela and Nagshineh [10] provide an extensive survey of DCA strategies. The main characteristic common to all DCA schemes is that all channels are kept in a central pool and are assigned dynamically to radio cells as new calls arrive in the system. A channel is eligible for use in any cell, provided interference constraints are met. DCA strategies vary in the criterion used to select the channel to be assigned from the set of all eligible channels. They also vary in whether they are centralized or distributed, and the synchronization mechanism used in the latter case. A wide variety of selection criteria to choose the assigned channel can be found in the literature. For example, in [19], the algorithm tries to maximize the amount of reuse: roughly, the channel that has been used the most often at distance r but least often at distances $r + 1$ and $r + 2$ is used. A number of recently proposed DCA schemes are based on measurement of signal strength from various eligible channels, and aim to increase the reusability of channels [16, 17]. A commonly used strategy [3, 5, 15], is the purely *greedy* strategy of using the minimum numbered channel among those eligible.

In this paper, we investigate in detail the worst-case performance of this greedy strategy.

Both centralized and distributed versions of the greedy strategy have been proposed and studied in the literature, and in the distributed cases, synchronization mechanisms based on message passing have been described. We are not concerned here with synchronization details, and hence, for purposes of ease of analysis, we impose a simple synchronization scheme based on rounds, that ensures that no conflicts take place owing to synchronization problems. In fact, we first analyze an offline algorithm, *i.e.* we study the performance of the algorithm on a single set of weights assigned to each node. We emphasize that we are not proposing this simplified version of the greedy strategy as a new DCA strategy to be implemented, but simply as a tool to be able to analyze easily the strategies mentioned earlier [3, 5, 15], and previously proposed in the literature. It is straightforward to see that the lower bounds we prove on this offline algorithm also hold for the online variations proposed in [3, 5, 15]. An additional motivation is to be able to provide a fair comparison in the online setting with previous borrowing strategies. Our lower bounds also apply to the signal-strength-based strategies mentioned above. We use standard yardsticks for measuring the efficacy of offline and online algorithms. An offline channel assignment algorithm is said to have *performance ratio k* if it uses at most k times the minimum number of channels required to satisfy all requests. An online channel assignment algorithm has *competitive ratio c* if uses at most c times as many channels overall as the optimal offline algorithm would.

Our Results. For any reuse distance r, we show that the offline greedy algorithm has a performance ratio of at most 6. For reuse distance 2, we show that the greedy strategy has a guaranteed performance ratio of 5/3; this is a tight bound, as it is possible to construct a hexagon graph and weight vector so that the algorithm uses 5/3 times the optimal number of channels. For the case $r = 3$, we show that the greedy strategy has a guaranteed performance ratio of 23/8. On the other hand, we show that in some cases, this strategy uses 7/3 times the optimal number of channels. Thus the performance ratio of the greedy strategy lies between 7/3 and 23/8. We also construct a weighted hexagon graph on which the greedy algorithm uses $12D/5$ channels where D is the maximum total weight on any clique in the graph, and is a lower bound on the number of channels needed. Finally, for arbitrary reuse distance, the greedy strategy has a performance ratio of at most 6, while a simple borrowing strategy is shown to have a performance ratio of at most $\frac{18r^2+6}{3r^2+21}$ which is $6 - o(1)$.

There are two straightforward online implementations of the offline greedy strategy. The first is essentially the offline algorithm repeated at every step, and has the same bounds on its performance. However, channels assigned to ongoing calls may be reassigned during successive steps, leading to intra-cell handoffs. We refer to this online implementation as G-R (greedy algorithm with reassignments), and mention it in order to provide a fair comparison with the borrowing strategies, which also perform reassignments. The second online implementation of the greedy strategy does not reassign channels, and is exactly the same as the strategies proposed in [3, 5, 15], with the synchronization details abstracted away. We call this version G-NR (greedy algorithm with no reassignments). Recently, Caragiannis *et al.* [1] showed an upper bound of 2.5 on the competitive ratio of this algorithm for reuse distance 2, for the case when calls are of infinite duration. In this paper, we show a lower bound of 2.5 for the general case. Our result thus demonstrates a tight bound on the competitive ratio of G-NR for reuse distance 2 when calls are of

Reuse distance	Best known upper bound	Upper bound for G-R	Lower bound for G-R	Best known lower bound
2	$\frac{4D}{3}$ [8, 12]	$\frac{5D}{3}$	$\frac{5D}{3}$	$\frac{9D}{8}$ [14]
3	$\frac{7D}{3}$ [4]	$\frac{23D}{8}$	$\frac{12D}{5}$	$\frac{5D}{4}$
r	$4D$ [9]	$6D$	$\frac{5D}{4}$	$\frac{5D}{4}$

Table 1: Summary of results for algorithms that perform reassignments. D is used to represent $D_r(G, w)$, the weighted clique number; G-R is the greedy algorithm that allows reassignments.

Reuse distance	Best known upper bound	Upper bound for G-NR	Lower bound for G-NR	Best known lower bound
2	$3D$ [9, 11]	$3D$	$\frac{5D}{2}$	$2D$ [8]
3	$3D$ [9]	$4D$	$3D$	$\frac{5D}{4}$
r	$4D$ [9]	$6D$	$\frac{5D}{4}$	$\frac{5D}{4}$

Table 2: Summary of results for algorithms with no reassignments. D is used to represent $D_r(G, w)$, the weighted clique number; G-NR is the greedy algorithm without reassignments. The bounds for G-NR apply to the algorithms in [2, 4, 13].

infinite duration. We also show that G-NR has competitive ratio between 3 and 4 for reuse distance 3. Both the lower bounds and the upper bounds apply to the DCA strategies from the literature that we wish to analyze. Our results are summarized in Tables 1 and 2. Note that borrowing strategies with competitive ratio 4/3 are known for reuse distance 2 [12, 14, 8]. For reuse distance 3, Feder and Shende have shown a borrowing strategy with competitive ratio 7/3 [4]. The best previously known algorithm that does not perform reassignment of channels is the cluster partitioning strategy of Jordan and Schwabe [9], which has a competitive ratio of 3 for reuse distances 2 and 3, and 4 for higher values of reuse distance.

The rest of the paper is organized as follows. The next section contains definitions and some technical facts. Section 3 and 4 give our results for reuse distances 2 and 3 respectively. Section 5 presents our results for arbitrary reuse distance. Conclusions and some discussion of future work are given in Section 6.

2 Preliminaries

Let G be a hexagon graph, *i.e.* a finite induced subgraph of the infinite triangular lattice. The parameter r is called the reuse distance. A *weighted graph* is a pair (G, w) where w is a positive integral vector indexed by the nodes of G. The component of w corresponding to node v is denoted $w(v)$ and is called the *weight* of v. The weight of a node represents the number of calls to be served at that node.

Given $G = (V, E)$, the graph $G^r = (V, E')$ is defined by $E' = E \cup E^2 \cup \ldots \cup E^{r-1}$. Thus any pair of nodes at distance $i < r$ in G is connected by an edge in G^r. The problem of channel assignment in a weighted hexagon graph (G, w) with reuse distance r is thus the same as multicoloring the graph G^r. We will assume that nodes are assigned channels from the ordered interval $[1, \Delta] = \{1, 2, \ldots, \Delta\}$, where $\Delta \geq 1$ depends on the particular graph under consideration. For instance, when a node is assigned the subinterval of channels $[i, j]$, it means that the node is assigned the channels $\{i, i+1, \ldots, j\}$.

The *unweighted clique number* of G^r is the maximum size of any clique in G^r, and is denoted $\omega(G^r)$. Similarly, the *chromatic number* of G^r, the minimum number of colors needed to color G^r, is denoted by $\chi(G^r)$. It is known that $\chi(G^r) = \omega(G^r)$ (see Section 5), and an optimal coloring can be computed in polynomial time. We assume that such an optimal coloring of the graph G^r is available; thus every node in G^r is assigned a color from the set $\{1, 2, \ldots, \chi(G^r)\}$. We define $N_r(v)$ to be all neighbors of v in G^r that have a lower color than that of v. For example, if v is a node with color 3 in G^r, then $N_r(v)$ consists of all neighbors of v in G^r that have color 1 or 2. Given a weighted graph (G, w), for any node v, we define $H_r(v)$ to be $w(v) + \Sigma_{u \in N_r(v)} w(u)$. Given a (partial) assignment of channels to $N_r(v)$, $RC_r(v)$ is defined to be the number of channels assigned to more than one node in $N_r(v)$. Thus $RC_r(v)$ is a measure of the reuse of channels within $N_r(v)$. Finally, we define $\overline{N}_r(v)$ to be all neighbors of v in G^r and $\overline{H}_r(v)$ to be $w(v) + \Sigma_{u \in N_r(v)} w(u)$.

We are now ready to define the greedy algorithm in a more precise manner. Given a weighted hexagon graph (G, w) and a reuse distance r, the algorithm is assumed to have a coloring of G^r using $\chi(G^r)$ colors available. The algorithm proceeds in $\chi(G^r)$ rounds. In round i, each node v with color i and weight $w(v)$ assigns to itself the set of $w(v)$ channels with the lowest numbers that are not used in $N_r(v)$. Notice that all nodes in $N_r(v)$ were assigned channels in previous rounds. This synchronization scheme is given only for ease of analysis; it is not hard to see that the lower bounds obtained for the above algorithm also apply to the greedy DCA algorithms in the literature.

We consider two online implementations of the greedy offline algorithm described above. The first one essentially consists of running the offline algorithm at every step on the new weight vector. We call this version *G-R*, the greedy algorithm with reassignments. At step t, every node v knows its weight $w_t(v)$. In every step, the algorithm proceeds in rounds, and the nodes with color i participate in round i. At every step, a node completely recalculates the channels to be assigned to all its nodes, and the channel assigned to an ongoing call can be changed from time to time. The second online implementation, called *G-NR*, does not perform reassignments. At step t, each node v knows the number of *new* call arrivals, say $n_t(v)$. The node v then assigns to itself the set of $n_t(v)$ channels with the lowest numbers that are not used at itself or in $\overline{N}_r(v)$. Any synchronization mechanism (including the one described above for the offline algorithm) could be used to prevent conflicts occurring during a single time step.

Given a weighted hexagon graph (G, w), we define $D_r(G, w)$ to be the maximum total weight on any clique in G^r. When solving the channel assignment problem on (G, w) with respect to reuse distance r, clearly $D_r(G, w)$ is a lower bound on the number of channels required. Let $mc(v)$ be the highest channel used by the vertex v. The following lemma is used frequently in subsequent sections.

Lemma 1 *For the offline greedy algorithm, and for any node v, $mc(v) \leq min\{H_r(v) - RC_r(v), w(v) + max_{u \in N_r(v)} mc(u)\}$.*

Proof: The number of distinct channels used by nodes in $N_r(v)$ is at most $H_r(v) - w(v) - RC_r(v)$. Therefore, v will never use a channel higher than $H_r(v) - RC_r(v)$. Also, if u is a node in $N_r(v)$ that uses the highest channel in v's neighborhood, v will never use more than the next $w(v)$ channels. □

The following technical lemma is useful in determining which nodes will be neighbors in G^r for a given r. In a hexagon graph, the vertices are all integer linear combinations $x\mathbf{p}$ $+y\mathbf{q}$ of the two vectors $\mathbf{p} = (1,0)$ and $\mathbf{q} = (\frac{1}{2}, \frac{\sqrt{3}}{2})$. Thus we may identify the vertices with the pairs (x, y) of integers, called their coordinates.

Lemma 2 *For any two nodes p and q at coordinates (x_1, y_1), (x_2, y_2) respectively, let $d(p, q)$ denote the distance between p and q.*

$$d(p,q) = \begin{cases} (x_2 - x_1) + (y_2 - y_1) & \textit{if } x_2 \geq x_1 \textit{ and } y_2 \geq y_1 & Case1 \\ y_2 - y_1 & \textit{if } x_2 \leq x_1 \textit{ and } y_2 \geq y_1 & \\ & \quad \textit{and } y_2 - y_1 \geq x_1 - x_2 & Case2 \\ x_1 - x_2 & \textit{if } x_2 \leq x_1 \textit{ and } y_2 \geq y_1 & \\ & \quad \textit{and } y_2 - y_1 \leq x_1 - x_2 & Case3 \\ (x_1 - x_2) + (y_1 - y_2) & \textit{if } x_2 \leq x_1 \textit{ and } y_2 \leq y_1 & Case4 \\ y_1 - y_2 & \textit{if } x_2 \geq x_1 \textit{ and } y_2 \leq y_1 & \\ & \quad \textit{and } x_2 - x_1 \leq y_1 - y_2 & Case5 \\ x_2 - x_1 & \textit{if } x_2 \geq x_1 \textit{ and } y_2 \leq y_1 & \\ & \quad \textit{and } x_2 - x_1 \geq y_1 - y_2 & Case6 \end{cases}$$

Proof: Straightforward. See Figure 1 for an example of each case, where $(x_1, y_1) = (0, 0)$. □

3 Reuse distance 2

In this section, we study the behavior of the greedy strategy when the reuse distance is 2. We show a tight bound for the performance of the static version of the greedy algorithm, and a lower bound for the competitive ratio of G-NR, the online greedy algorithm with no reassignments.

3.1 Static case

For this case, the best known algorithms [12, 14] have performance ratio 4/3. We show that the greedy strategy has performance ratio 5/3. This is a tight bound, as there are weighted hexagon graphs where the greedy algorithm uses 5/3 times the optimal number of channels required.

Theorem 1 *For reuse distance 2, the performance ratio of the offline greedy algorithm is $\frac{5}{3}$.*

Proof: Let (G, w) be a weighted hexagon graph, where every node is colored red, blue, or green (corresponding to colors $1, 2$, and 3 in the base coloring of G). We denote $D_2(G, w)$ by D_2. It is easy to see that all red and blue nodes can be assigned channels using the first D_2 channels $C[1, 2, \ldots, D_2]$. In the third round of the algorithm, we assign channels to the green nodes. For any green node v with $w(v) \geq \frac{2D_2}{3}$, since $N_2(v)$ and v can be covered by three cliques(as shown in Figure 2), $H_2(v) \leq 3D_2 - 2w(v) \leq 3D_2 - \frac{4D_2}{3} = \frac{5D_2}{3}$.

Thus, by Lemma 1, $mc(v) \leq H_2(v) \leq \frac{5D_2}{3}$. If instead, $w(v) < \frac{2D_2}{3}$ since we can use the first D_2 colors $C[1, 2, \ldots, D_2]$ to color the red and blue nodes, by Lemma 1, we have $mc(v) \leq w(v) + max_{u \in N_2(v)} mc(u) \leq 2D_2/3 + D_2 = \frac{5D_2}{3}$. Since a green node can never use a channel higher than $5D_2/3$ and D_2 is a lower bound on the number of channels needed, the algorithm has performance ratio at most 5/3.

We note that $\frac{5}{3}$ is also a lower bound for the performance ratio of the greedy algorithm. For example, in Figure 3, the reader can verify that $D_2 = 3k$, and that there is an optimal assignment using $3k$ channels, but the greedy algorithm will use $5k$ channels. Thus, the greedy algorithm uses 5/3 times the optimal number of channels required. □

Corollary 1 *G-R, the online greedy algorithm with reassignments, has competitive ratio* $\frac{5}{3}$ *for reuse distance 2.*

3.2 Online case

For the online case, a trivial upper bound of 3 for the performance ratio of the greedy algorithm follows from the fact that $\overline{N}_2(v) \cup \{v\}$ can be covered with 3 cliques (see Figure 2). In [1], the authors use a more detailed analysis of $\overline{H}_2(v)$ to prove that for reuse distance 2, the competitive ratio p of the greedy algorithm satisfies $2.429 \leq p \leq \frac{5}{2}$ when calls are of infinite duration. For the general case, we construct a hexagon graph and a sequence of weight vectors to show a lower bound of 2.5 for this case.

Theorem 2 *G-NR, the online greedy algorithm with no reassignments, has competitive ratio* $\frac{5}{2}$ *for reuse distance* 2.

Proof: We provide a hexagon graph (see Figure 4) and a sequence of call arrivals and terminations on nodes in the graph so that the greedy algorithm is forced to use 20 channels, while the reader can verify that the optimal offline algorithm needs only 8 channels. Let the pair (α, i) represent:

$$\begin{cases} i \text{ new calls come into node } \alpha & \text{if } i > 0 \\ i \text{ calls finish at node } \alpha & \text{if } i < 0 \end{cases}$$

It is easy to see that the number of calls can easily be multiplied by any $k > 0$ to give arbitrarily large weight vectors. While in principle, the adversary can specify the exact calls that finish, in our example, when reducing the weight, we always reduce it to zero, that is, *all* current calls terminate. Thus there is no need to specify the exact calls that finish in a particular time step. Finally, many elements of the following sequence could be done in parallel; we do not use this optimization for clarity and ease of verification.

The adversary's objective is to make the node o use the channels $[19, 20]$. This happens only if all the channels $[1, 18]$ are currently in use in o's neighborhood. Therefore the first goal of the adversary is to make such a situation occur. It is intuitively clear that it is harder to make a neighbor of o use the *high* channels than the *low* channels. Thus we work on the harder task first. We first force the node n to use the channels $[16, 18]$. To do this, once again, we create a situation where the 6 neighbors of n use all the channels $[1, 15]$. In particular, the nodes h, o and s will have weight 2 each, and will use the higher channels $[10, 15]$ and the remaining neighbors of n have weight 3 each, and will use the channels $[1, 9]$. The value of $D_2[G]$ never exceeds 8.

First we work on the node s. The sequence $(E,2),(D,3),(E,-2),\ (t,2),\ (A,5),(t,-2),$ $(D,-3)$ leads to A using the channels $[6,10]$. Next, the sequence $(C,5),(z,3),(C,-5),$ $(A,-5)$ leads to z using the channels $[11,13]$. Now, the sequence $(x,3),(y,3),(x,-3),$ $(\ell,3),\ (r,4),\ (y,-3),(\ell,-3)$ leads to r using the channels $[7,10]$. Increasing the weight of n to 6 leads to n using the channels $[1,6]$ since n has no neighbors of positive weight. Now, increasing the weight of s to 2 forces s to use the channels $[14,15]$. The sequence $(r,-4),(z,-3)$ ensures that the only nodes with non-zero weight are s and n.

Next we work on the node o. The sequence $(w,3),(v,3),(w,-3)$ leads to v using the channels $[1,3]$. Next, the sequence $(k,3),(p,5),(k,-3),\ (v,-3),\ (o,2),(p,-5)$ leads to the node o using the channels $[12,13]$. At this point, o,s, and n are the only nodes with non-zero weight. Next, to make node h use the channels $[10,11]$, we employ the sequence $(a,3),(b,3),(a,-3),\ (d,3),(c,3),(b,-3),\ (d,-3),(h,2)$. Next, the sequence $(n,-6),(c,-3)$ ensures that the only nodes with positive weight are h, o, and s, and they are using the channels $[10,15]$.

Finally we make the other neighbors of n use the channels $[1,9]$. The sequence $(y,3),(r,3),$ $(y,-3),(g,3),\ (m,3),(g,-3),(r,-3)$ makes m use the channels $[7,9]$. The sequence $(i,3),(A,3),$ $(t,3),(A,-3)$ makes i use the channels $[1,3]$ and t use the channels $[4,6]$. Then the sequence $(n,3),(h,-2),\ (o,-2),\ (s,-2),(m,-3),(i,-3),(t,-3)$ leads to n using the channels $[16,18]$ and being the only node of positive weight in the graph.

Recall that the final goal is to make o use the channels $[19,20]$. At this point, we work on the other neighbors of o. First we make the node j use the channels $[13,15]$. The sequence $(w,4),(q,4),(w,-4),\ (f,4),(k,4),(f,-4),(q,-4)$ leads to k using the channels $[9,12]$. Next, $(d,5),(e,3),\ (d,-5),(o,5),\ (j,3),\ (k,-4),(e,-3),(o,-5)$ achieves the purpose of j using the channels $[13,15]$. Now, the sequence $(k,2),(q,2),(B,3),\ (v,2),(p,3),(k,-2),$ $(q,-2),(v,-2)$ leads to p getting the channels $[7,9]$. At this point, $(z,3),(t,3),(z,-3),$ $(i,3),(u,3),(B,-3),(o,2)$, leads to t getting the channels $[4,6]$, i getting the channels $[1,3]$, u getting the channels $[10,12]$, and finally, o getting the channels $[19,20]$ as desired.

The reader can verify that the optimal offline algorithm can perform the assignment with 8 channels. In particular, we can break up the sequence of calls into two parts: at the end of the first part, node n has weight 2, and its neighbors alternately have weight 2 or 3. In the second part of the call sequence, first all neighbors of node n lose all their calls, and then the other neighbors of node o get 3 calls each, culminating with node o getting two calls. By looking ahead a few steps, the optimal assignment can ensure that node n ends up with channels $[1,3]$ at the end of the first part, and that the alternating neighbors of node n have channel sets $[4,6]$ or $[7,8]$. Similarly, in the second part of the sequence, the neighbors of node o can be assigned either $[1,3]$ or $[4,6]$, so that node o can finally be assigned channels $[7,8]$. □

4 Reuse distance 3

In this section, we study the case when the reuse distance $r = 3$. We show that for the static case, the greedy strategy has performance ratio at least that of the borrowing strategy given in [4]. This shows that for the static case, the greedy strategy is no better than the borrowing strategy in the worst case. We also show that G-NR, the online greedy algorithm with no reassignments, has competitive ratio at least that of the cluster partitioning strategy of [9]. This shows that for the online case, the greedy strategy is no better than the cluster partitioning strategy in the worst case.

4.1 Static case

Feder and Shende have given a borrowing strategy for this case that has performance ratio 7/3 [4]. In particular, for any weighted hexagon graph (G, w), their algorithm uses at most $7D_3(G, w)/3$ channels. We show in this section that there are situations when the greedy algorithm performs worse than this. Also, we show an upper bound on the performance ratio of the greedy algorithm; however, we were unable to prove a tight bound for this case.

For reuse distance 3, the underlying unweighted graph G^3 can be colored with 7 colors. Let G be a weighted hexagon graph and let D_3 denote $D_3(G, w)$. For convenience, we refer to a node with color 1 as a 1-node, a node with color 2 as a 2-node and so on. According to the algorithm, when a node i is being assigned channels, the only nodes which already have assigned channels, and may therefore affect its assignment are nodes with lower colors. For example, when a 4-node is being assigned channels, the only nodes that may affect the assignment are the 1, 2 and 3-nodes in its neighborhood. Thus, one can obtain an upper bound on the largest channel used by a 4-node by knowing bounds on the largest channel used by 1, 2, and 3-nodes. We now prove a succession of lemmas, showing upper bounds on the largest channel used by nodes of colors from 1 to 7. Clearly, upper bounds on the largest channel used by nodes of colors 1 to 7 provides an upper bound on the performance ratio of the greedy algorithm.

Lemma 3 *All the 1-nodes and 2-nodes can be assigned using the first D_3 channels $C[1, 2, \ldots, D_3]$.*

Proof: Straightforward. □

Lemma 4 *For any* 3*-node v, $mc(v) \leq min\{D_3 + w(v), 3D_3 - 2w(v)\} \leq \frac{5D_3}{3}$.*

Proof: For any 3-node v, the set $\{v\} \cup N_3(v)$ can be covered by three cliques. and also, by Lemma 3, no element of $N_3(v)$ can use a channel higher than D_3. It then follows from an argument similar to the proof for green nodes in Theorem 1 that $mc(v) \leq \min\{D_3 + w(v), 3D_3 - 2w(v)\} \leq \frac{5D_3}{3}$. □

In fact, this is a tight bound, as shown by the example in Figure 5. The reader can verify that $D_3 = 3k$, but the greedy algorithm will use $5k$ channels.

We prove the following general fact that will be useful later.

Fact 1 *Let p be a 3-node in $N_3(v)$ where v is a 4-node, and let $w(p) = D_3 - w(v) - i$. Then $H_3(p) \leq D_3 + 2i$.*

Proof: Since p is a 3-node, $N_3(p)$ can be covered by three cliques, each consisting of a pair of 1- and 2-nodes. Since two of these cliques are also sub-graphs of a clique containing both v and p, their total weights are at most i each (see Figure 6 for one such position of p with respect to v, the other positions can be verified by the reader). The third clique in $N_3(p)$ has weight at most $w(v) + i$. Thus, $H_3(p) \leq D_3 + 2i$. □

Lemma 5 *For any 4-node v, $mc(v) \leq \frac{13D_3}{7}$.*

Proof: For any 4-node v, since $\{v\} \cup N_3(v)$ can be covered by three cliques, $H_3(v) \leq 3D_3 - 2w(v)$. So if $w(v) \geq \frac{4D_3}{7}$, by Lemma 1, $mc(v) \leq H_3(v) \leq 3D_3 - 8D_3/7 = \frac{13D_3}{7}$. On the other hand, by Lemma 4, none of the nodes in $N_3(v)$ can use channels higher than $\frac{5D_3}{3}$. Therefore, if $w(v) \leq \frac{4D_3}{21}$, $mc(v) \leq \frac{5D_3}{3} + \frac{4D_3}{21} = \frac{13D_3}{7}$.

It remains to show that $mc(v) \leq \frac{13D_3}{7}$ when $\frac{4D_3}{21} < w(v) < \frac{4D_3}{7}$. Let $w(v) = \frac{4D_3}{21} + k$, where $0 < k < \frac{8D_3}{21}$. Then for any 3-node p in $N_3(v)$, $w(p) \leq \frac{17D_3}{21} - k$. Let $w(p) = \frac{17D_3}{21}$ $-k - i$. We claim that $mc(p) \leq \frac{5D_3}{3} - k$.

If $i < \frac{D_3}{7}$, by Fact 1, we have $H_3(p) \leq D_3 + 2i \leq 9D_3/7 \leq \frac{5D_3}{3} - k$, and therefore by Lemma 1, $mc(p) \leq \frac{5D_3}{3} - k$. If instead $i \geq \frac{D_3}{7}$, we have $w(p) \leq \frac{2D}{3} - k$. Since none of the 1- and 2-nodes in $N_3(p)$ will use colors higher than D_3, by Lemma 1, $mc(p) \leq D_3 + w(p) \leq \frac{5D_3}{3} - k$.

Since no 3-node in $N_3(v)$ uses a channel higher than $\frac{5D_3}{3} - k$, and any 1- or 2-node in $N_3(v)$ uses channels numbered at most $D_3 < \frac{5D_3}{3} - k$, by Lemma 1, $mc(v) \leq w(v) + \frac{5D_3}{3} - k \leq \frac{13D_3}{7}$. □

The bound in Lemma 5 is a tight bound, as shown by the example in Figure 7. The reader can verify that $D_3 = 7k$ but the greedy algorithm uses $13k = 13D_3/7$ channels in this case.

Lemma 6 *For any 5-node v, $mc(v) \leq \frac{11D_3}{5}$.*

Proof: For any 5-node v, by Lemmas 3 to 5, since none of its neighbors will use channels higher than $\frac{13D_3}{7}$, if $w(v) \leq \frac{12D_3}{35}$, by Lemma 1, $mc(v) \leq \frac{13D_3}{7} + w(v) = \frac{11D_3}{5}$.

Consider two of the 1-nodes 1_a and 1_b in v's neighborhood (see Figure 8). If $w(1_a) \leq w(1_b)$, all of 1_a's channels will also be used by 1_b, and thus $RC_3(v) \geq w(1_a)$. It is easy to verify that $H_3(v) \leq 3D_3 + w(1_a) - 2w(v)$ and therefore by Lemma 1, $mc(v) \leq H_3(v) - RC_3(v) \leq 3D_3 - 2w(v) \leq \frac{11D_3}{5}$ if $w(v) \geq \frac{2D_3}{5}$. Similarly, if $w(1_b) \leq w(1_a)$, we can also show that $mc(v) \leq 3D_3(v) - 2w(v) \leq \frac{11D_3}{5}$.

It remains to show that $mc(v) \leq \frac{11D_3}{5}$ when $\frac{12D_3}{35} < w(v) < \frac{2D_3}{5}$. Let $w(v) = \frac{12D_3}{35} + k$, where $0 < k < \frac{2D_3}{35}$. Then for any 4-node u in v's neighborhood, $w(u) \leq \frac{23D_3}{35} - k$.

Claim 6.1 shows that $mc(u) \leq \frac{13D_3}{7} - k$. Indeed, if u is 1-, 2, or 3-node in v's neighborhood, $mc(u) \leq \frac{5D_3}{3} \leq \frac{13D_3}{7} - k$, for all values of k in the range $0 < k < \frac{2D_3}{35}$. It follows as a consequence of Lemma 1 that $mc(v) \leq w(v) + \frac{13D_3}{7} - k = \frac{12D_3}{35} + k + \frac{13D_3}{7} - k = \frac{11D_3}{5}$. □

Claim 6.1 *$mc(u) \leq \frac{13D_3}{7} - k$.*

Proof: Let $w(u) = \frac{23D_3}{35} - k - i$. Since $N_3(u)$ consists of 3 cliques, and all the nodes in one of them are also neighbors of v, therefore $H_3(u) \leq \frac{23D_3}{35} - k - i + 2(\frac{12D_3}{35} + k + i) + i = \frac{47D_3}{35} + k + 2i$. If $w(u) \geq \frac{2D_3}{5}$, we have $k + i < \frac{9D_3}{35}$ and $H_3(u) < \frac{13D_3}{7} - k$, therefore by Lemma 1, $mc(u) \leq \frac{13D_3}{7} - k$. If instead $w(u) \leq \frac{4D_3}{21} - k$, by Lemmas 1, 3, and 4, $mc(u) \leq \frac{5D_3}{3} + w(u) = \frac{13D_3}{7} - k$.

Thus we only need to consider $\frac{4D_3}{21} - k < w(u) < \frac{2D_3}{5}$. Suppose $w(u) = \frac{4D_3}{21} - k + \ell$, where $\ell < \frac{22D_3}{105} + k$. Then for any 3-node p in $N_3(u)$, $w(p) \leq \frac{17D_3}{21} + k - \ell$. Let $w(p) = D_3 - w(v) - j = \frac{17D_3}{21} + k - \ell - j$. If $j - k \geq \frac{D_3}{7}$, then $w(p) \leq \frac{2D_3}{3} - \ell$ and by Lemma 4, $mc(p) \leq \frac{5D_3}{3} - \ell$. It then follows from Lemma 1 that $mc(u) \leq \frac{5D_3}{3} - \ell + \frac{4D_3}{21} - k + \ell = \frac{13D_3}{7} - k$ as claimed.

Otherwise if $j - k < \frac{D_3}{7}$, by Fact 1, $H_3(p) \leq D_3 + 2j < \frac{9D_3}{7} + 2k$. Thus, by Lemma 1, $mc(p) \leq \frac{9D_3}{7} + 2k$. It then follows from Lemma 1 that $mc(u) \leq \frac{9D_3}{7} + 2k + \frac{4D_3}{21} - k$ $+\ell = \frac{31D_3}{21} + k + \ell < \frac{59D_3}{35} + 2k < \frac{13D_3}{7} - k$, as required. □

The above bound is tight, as there exists a hexagon graph for which $D_3 = 5k$ but the greedy algorithm uses $11k = 11D_3/5$ channels (see Figure 8).

Lemma 7 *For any 6-node v, $mc(v) \leq \frac{5D_3}{2}$.*

Proof: Let v be a 6-node, and let u be the node in $N_3(v)$ using the maximum-numbered channel. If u is a 1- or 2-node, then by Lemmas 3 and 1, $mc(v) \leq D_3 + w(v) \leq 2D_3$. Otherwise, if u is a 3-node, then $w(v) \leq D_3 - w(u)$ and by Lemmas 4 and 1, $mc(v) \leq D_3 + w(u) + w(v) \leq 2D_3$. If u is a 4-node, then $mc(v) \leq min\{13D_3/7 + w(v), 3D_3(v) - 2w(v)\}$ $\leq 47D_3/21 \leq 5D_3/2$ as claimed. Claim 7.1 below gives three upper bounds on $mc(u)$ for the case when u is a 5-node, and Claim 7.2 establishes a minimum value for the functions obtained in Claim 7.1. The two claims together establish the lemma. □

Claim 7.1 *Let $w(v) = x$ and $w(u) = D_3 - x - y$ where u is a 5-node.*

1. $mc(v) \leq 3D_3 - 2x + y$.
2. $mc(v) \leq D_3 + 2x + 2y$.
3. $mc(v) \leq (19D_3 + x - 6y)/7$.

Proof: Notice that there are three 5-nodes in $N_3(v)$. In other words, the node u can be in three different positions relative to the node v. One such position is shown in Figures 9 and Figure 10. The case when $w(1_a) \leq w(1_b)$ is shown in Figure 9. It can be seen that $N_3(v) \cup \{v\}$ can be covered by four cliques and the node 1_a. Two of these four cliques also include the node u. Furthermore, since $w(1_a) \leq w(1_b)$, all channels used at 1_a are also reused at node 1_b, thus $RC_3(v) \geq w(1_a)$. Therefore, by Lemma 1, $mc(v) \leq H_3(v) - RC_3(v) \leq 4D_3 + w(1_a) - 3x - (D_3 - x - y) - RC_3(v) \leq 3D_3 - 2x + y$, as claimed. A similar argument applies when $w(1_b) \leq w(1_a)$; see Figure 10. Finally, we can show the identical result for the remaining two positions of u. This finishes the proof of (1).

To see (2), observe that $N_3(u)$ consists of 3 cliques of 1-, 2-, 3-, and 4-nodes, in addition to a 1-node, which can always be chosen to be the smaller of nodes 1_a and 1_b in $N_3(u)$ (see Figure 8). Further, one of these cliques is such that all nodes in it are neighbors of v, and thus has weight at most y, while the other two cliques have weight at most $x + y$. Thus, using an argument similar to the one in the previous paragraph, we can show that $H_3(u) \leq D_3 + x + 2y$, and thus by Lemma 1, $mc(u) \leq D_3 + x + 2y$. This implies in turn that $mc(v) \leq D_3 + 2x + 2y$.

To show (3), we need to analyze more carefully the maximum channels used by 3 and 4-nodes. In particular, if x_3 is a 3-node with a 4-node neighbor x_4, then $H_3(x_3) \leq 3D_3 - 2w(x_3) - 2w(x_4)$. Therefore, $mc(x_3) \leq 5D_3/3 - 2x_4/3$. We take this into account when analyzing a 4-node p with a 5-node neighbor u. In particular, the maximum channel used by any neighbor of p is at most $5D_3/3 - 2w(p)/3$. Also, $H_3(p) \leq 3D_3 - 2w(p) - w(u)$. It follows from Lemma 1 that $mc(p) \leq 13D_3/7 - w(u)/7$. Recalling that $w(u) = D_3 - x - y$ in this case, we obtain $mc(u) \leq mc(p) + w(u) \leq (19D_3 - 6x - 6y)/7$, and thus, by Lemma 1, $mc(v) \leq (19D_3 + x - 6y)/7$ as claimed. □

Claim 7.2 $min(3D_3 - 2x + y, D_3 + 2x + 2y, (19D_3 + x - 6y)/7) \leq \frac{5D_3}{2}$.

Proof: Assume the claim is not true at $x = a$, $y = b$, that is, all three functions evaluate to greater than $\frac{5D_3}{2}$ for these values. This implies:

$$\begin{aligned} 3D_3 - 2a + b &> 5D_3/2 && (1)\\ D_3 + 2a + 2b &> 5D_3/2 && (2)\\ (19D_3 + a - 6b)/7 &> 5D_3/2 && (3) \end{aligned}$$

Then (1) + (2) $\Rightarrow b > \frac{D_3}{3}$, while (1) + (3) $* 14 \Rightarrow b < \frac{7D_3}{22}$, which yields a contradiction.
□

Lemma 8 *For any 7-node v, $mc(v) \leq \frac{23D_3}{8}$.*

Proof: For any 7-node v, $N_3(v) \cup \{v\}$ can be covered by four cliques, and so $H_3(v) \leq 4D_3 - 3w(v)$. If $w(v) \geq \frac{3D_3}{8}$, by Lemma 1, $mc(v) \leq H_3(v) \leq 4D_3 - 3w(v) \leq \frac{23D_3}{8}$. On the other hand, by Lemmas 3 to 7, none of the nodes in $N_3(v)$ can use channels higher than $\frac{5D_3}{2}$. So if $w(v) \leq \frac{3D_3}{8}$, by Lemma 1, $mc(v) \leq \frac{5D_3}{2} + w(v) \leq \frac{23D_3}{8}$. □

It is possible to construct a weighted hexagon graph where the greedy algorithm uses $7D_3/3$ channels, while an optimal assignment using D_3 channels exists. Thus, the performance ratio of the greedy algorithm is at least 7/3. However, there is a weighted hexagon graph (G, w) where $D_3(G, w) = 5k$ and the greedy algorithm uses $12k = 12D_3/5$ channels, as shown in Figure 11[1]. The reader can verify the assignment given by the greedy algorithm.

The following theorem is a consequence of Lemmas 3 to 8 and the discussion in the paragraph above:

Theorem 3 *For reuse distance 3:*

1. *The offline greedy algorithm has performance ratio p where $\frac{7}{3} \leq p \leq \frac{23}{8}$.*
2. *There is a weighted hexagon graph (G, w) such that the offline greedy algorithm uses $12D_3(G, w)/5$ channels.*

Corollary 2 *G-R, the online greedy algorithm with reassignments, has competitive ratio p where $\frac{7}{3} \leq p \leq \frac{23}{8}$ for reuse distance 3.*

4.2 The online case

For this case, the cluster partitioning strategy given in [9] can easily be shown to have competitive ratio 3, as mentioned in Section 5. We show here that the greedy algorithm has competitive ratio between 3 and 4.

Theorem 4 *G-NR, the online greedy algorithm with no reassignments, has competitive ratio p where $3 \leq p \leq 4$ for reuse distance 3.*

[1]However, this graph does not appear to have an assignment with fewer than $6k$ channels.

Proof: The upper bound on p follows simply from the fact that for any node v, $\overline{N}_3(v)$ is covered by 4 cliques, each with weight at most D, which implies that $mc(v) \leq \overline{H}_3(v) \leq 4D$. To show the lower bound, we construct a hexagon graph and a series of call arrivals and departures so that the algorithm is forced to use 9 channels while the optimal offline algorithm needs only 3 channels. The proof is similar to that of Theorem 2, but the sequence of call arrivals and terminations as well as the graph used are much larger. In what follows, we use $(x, w, [c_1, c_2])$ to denote the arrival of w new calls at node x, which increases the weight of node x to w from 0. This in turn forces the greedy algorithm to use the contiguous channels c_1 to c_2 at node x. The notation $\not{x}$ denotes the removal of all calls at node x, thus reducing the weight to 0, and causing the release of all channels currently being used at node x. We note that many of the steps in this sequence can be performed in parallel, thus shortening the sequence, as well as the number of steps required. We give this longer sequence here for clarity and ease of verification.

Since the number of nodes involved is quite large, we break up the sequence into three sequences, given in Tables 3, 4 and 5. The first sequence, given in Table 3, is intended to make the node o use the channel 8. The corresponding graph is given in Figure 12. In order to make this happen, we have to first make the neighbors of o use all channels $[1, 7]$. These goals are listed in the first column on Table 3. These in turn lead to other sub-goals listed in the second column. Similarly, Table 4 in conjunction with Figure 12 describes the sequence to force node C to use channel 7 given that node o is already using channel 8. Finally, Table 5 in conjunction with Figure 13 gives the sequence to force node p to use channel 9 given that nodes o and C are using channels 8 and 7. The reader can also verify that the optimal offline algorithm can perform the assignment with 3 channels. □

5 Arbitrary reuse distance

In this section, we consider the problem of channel assignment with respect to an arbitrary reuse distance r. We show a lower bound on the performance of any algorithm, and analyze the worst-case performance of the greedy strategy as well as a straightforward borrowing strategy.

Lemma 9 *For any reuse distance $r \geq 3$, there are hexagon graphs G such that G^r contains a 5-cycle as an induced subgraph.*

Proof: If r is odd, we choose $v_1, v_2, \ldots, v_5$, at coordinates $(0,0), (r-1, 0), (r-1, r-1)$, $(0, \frac{3r-3}{2})$, and $(-\frac{r-1}{2}, r-1)$ respectively. Otherwise, we choose them at coordinates $(0,0), (r-1, 0), (r-1, r-1), (0, \frac{3r}{2} - 2)$, and $(-\frac{r}{2}, r-1)$ respectively. It follows from Lemma 2 that for any two nodes (v_i, v_j), $d(v_i, v_j) \leq r$ if and only if $i = j \bmod 5 + 1$. For example, in Figure 14(a), given reuse distance 3, the 5-cycle consists of the five nodes at $(0,0), (2,0), (2,2), (0,3), (-1,2)$, and in Figure 14(b), given reuse distance 4, the 5-cycle consists of the five nodes at $(0,0), (3,0), (3,3), (0,4), (-2,3)$. □

Theorem 5 *For any reuse distance $r \geq 3$, there exists a weighted hexagon graph (G, w) such that any algorithm for channel assignment must use $5D_r(G, w)/4$ channels.*

Proof: Let G be a hexagon graph that contains a 5-cycle as an induced subgraph (its existence is confirmed by Lemma 9). We assign every node in the 5-cycle with weight k.

Goal	Sub-goal	Sequence
$(h, 1, [7])$		generate node h's neighborhood
	$(o, 2, [1, 2])$	$(e, 1, [1]), (f, 1, [1]), (c, 1, [2]), (o, 2, [1, 2])$
	$(i, 2, [5, 6])$	$(p, 1, [3]), (t, 1, [1]), (s, 2, [2, 3]), (l, 1, [4]), (i, 2, [5, 6])$
	$(b, 2, [3, 4])$	$(d, 1, [1]), (a, 1, [2]), (b, 2, [3, 4])$
		clean up neighborhoods of i, b, and o
		$\cancel{d}$, $\cancel{a}$, $\cancel{e}$, $\cancel{c}$, $\cancel{p}$, $\cancel{f}$, $\cancel{l}$, $\cancel{s}$, $\cancel{t}$
		raise h's weight
		$(h, 1, [7])$
		clean up h's neighborhood
		$\cancel{i}$, $\cancel{b}$, $\cancel{o}$
$(u, 1, [6])$		generate node u's neighborhood
	$(A, 1, [5])$	$(z, 2, [1, 2]), (H, 1, [3])$, $\cancel{z}$, $(v, 1, [1])$
		$(N, 2, [1, 2]), (I, 1, [4]), (F, 1, [1]), (B, 2, [2, 3]), (A, 1, [5])$
	$(q, 2, [3, 4])$	$(d, 1, [1]), (j, 1, [2]), (q, 2, [3, 4])$
	$(o, 2, [1, 2])$	$(o, 2, [1, 2)$
		clean up neighborhoods of A, q, and o
		$\cancel{d}$, $\cancel{j}$, $\cancel{v}$, $\cancel{H}$, $\cancel{I}$, $\cancel{N}$, $\cancel{B}$, $\cancel{F}$
		raise u's weight
		$(u, 1, [6])$
		clean up u's neighborhood
		$\cancel{A}$, $\cancel{q}$, $\cancel{o}$
$(k, 1, [5])$		generate k's neighborhood
	$(q, 2, [3, 4)$	$(v, 1, [1]), (d, 1, [1]), (j, 1, [2]), (q, 2, [3, 4])$
	$(o, 2, [1, 2])$	$(o, 2, [1, 2])$
		clean up neighborhoods of q and o
		$\cancel{d}$, $\cancel{v}$, $\cancel{j}$
		raise k's weight
		$(k, 1, [5])$
		clean up k's neighborhood
		$\cancel{q}$, $\cancel{o}$
$(r, 1, [4])$		$(D, 2, [1, 2]), (x, 1, [3]), (p, 2, [1, 2]), (r, 1, [4])$
		$\cancel{D}$, $\cancel{x}$, $\cancel{p}$
$(n, 1, [3])$		$(m, 2, [1, 2]), (n, 1, [3]), \cancel{m}$
$(p, 1, [2])$		$(o, 1, [1]), (p, 1, [2])$, $\cancel{o}$
$(g, 1, [1])$		$(g, 1, [1])$
$(o, 1, [8])$		$(o, 1, [8])$
		$\cancel{h}$, $\cancel{u}$, $\cancel{k}$, $\cancel{r}$, $\cancel{n}$, $\cancel{p}$, $\cancel{g}$

Table 3: Sequence of calls as a result of which node o uses channel 8. See Figure 12.

Goal	Sub-goal	Sequence
$(J,2,[5,6])$	$(M,1,[4])$ $(D,1,[3])$ $(P,1,[2])$ $(F,1,[1])$	$(R,2,[1,2]),(Q,1,[3]),(G,2,[1,2]),(M,1,[4])$ $(D,1,[3])$ $(O,1,[1]),(P,1,[2])$ $(F,1,[1])$ $R\!\!\mid,\ O\!\!\mid,\ Q\!\!\mid,\ G\!\!\mid$ $(J,2,[5,6])$ $M\!\!\mid,\ D\!\!\mid,\ P\!\!\mid,\ F\!\!\mid$
$(E,2,[3,4])$	$(L,1,[2])$ $(w,1,[1])$	$(O,1,[1]),(L,1,[2])$ $(w,1,[1])$ $(E,2,[3,4])$ $O\!\!\mid,\ w\!\!\mid,\ L\!\!\mid$
$(y,2,[1,2])$		$(y,2,[1,2])$
$(C,1,[7])$		$(C,1,[7])$ $J\!\!\mid, E\!\!\mid,\ y\!\!\mid$

Table 4: Sequence of calls as a result of which node C uses channel 7, given that node o is already using channel 8. See Figure 12.

Goal	Sub-goal	Sequence
$(i,1,[6])$	$(j,1,[4])$ $(b,1,[2])$ $(c,1,[5])$ $(u,1,[3])$ $(p,1,[1])$	$(e,2,[1,2]),(d,1,[3]),(s,2,[1,2]),(j,1,[4])$ $(a,1,[1]),(b,1,[2])$ $(c,1,[5])$ $(y,2,[1,2]),(u,1,[3])$ $(p,1,[1])$ $(i,1,[6])$ $e\!\!\mid,\ y\!\!\mid,\ j\!\!\mid,\ b\!\!\mid,\ c\!\!\mid,\ u\!\!\mid,\ p\!\!\mid,\ a\!\!\mid,\ d\!\!\mid,\ s\!\!\mid$
$(t,1,[5])$	$(m,1,[2])$ $(v,1,[3])$ $(x,1,[4])$ $(p,1,[1])$	$(h,1,[1]),(m,1,[2]),\ h\!\!\mid$ $(A,1,[1]),(v,1,[3])$ $(D,1,[2]),(x,1,[4]),\ A\!\!\mid,\ D\!\!\mid$ $(p,1,[1])$ $(t,1,[5])$ $m\!\!\mid,\ v\!\!\mid,\ x\!\!\mid,\ p\!\!\mid$
$(n,1,[4])$	$(f,1,[1])$ $(l,1,[2])$ $(v,1,[3])$	$(f,1,[1])$ $(l,1,[2])$ $(z,1,[1]),(v,1,[3]),z\!\!\mid$ $(n,1,[4])$ $v\!\!\mid,\ l\!\!\mid,\ f\!\!\mid$
$(q,1,[3])$ $(w,1,[2])$ $(g,1,[1])$		$(r,2,[1,2]),(q,1,[3]),r\!\!\mid$ $(B,1,[1]),(w,1,[2]),\ B\!\!\mid$ $(g,1,[1])$
$(p,1,[9])$		$(p,1,[9])$

Table 5: Sequence of calls as a result of which node p uses channel 9, given that nodes o and C are already using channels 8 and 7 respectively. See Figure 13.

Then $D_r(G,w) = 2k$. Further, since $5k$ calls need to be served, and each channel can be used at most 2 nodes, the optimal number of channels required is $\frac{5k}{2} = 5D_r(G,w)/4$. $\square$

5.1 Static case

In this section, we analyze the performance of a simple borrowing strategy, along the lines of the algorithm for reuse distance 3 in [4]. The following lemma was derived independently in [13] and [18]. We give a simple construction below for convenience.

Lemma 10 ([13, 18]) *Given G a hexagon graph, and reuse distance r,*

$$\omega(G^r) = \chi(G^r) = \begin{cases} \frac{3r^2}{4} & \textit{if } r \textit{ is even} \\ \frac{3r^2+1}{4} & \textit{otherwise} \end{cases}$$

Let G be the infinite triangular lattice. We construct the maximum-sized clique in G^r by finding nodes in G that are at distance at most $r-1$ from each other. If r is even, we build *layers* moving outwards from a triangle. The first layer consists of 3 nodes, the second layer consists of 9 nodes, and the $\frac{r}{2}^{th}$ layer, which is the outermost layer, consists of $3 + 6(\frac{r}{2} - 1) = 3r - 3$ nodes. It is easy to verify that this set of nodes comprises a clique, and in fact, a maximum-sized clique. The total number of nodes in the clique is $3 + 9 + ... + (3 + 6(\frac{r}{2} - 1)) = \frac{3r^2}{4}$. For example, Figure 6(a) shows that $\omega(G^2) = 3$, and Figure 6(b) shows that $\omega(G_4) = 12$. It is also easy to construct a tiling of the infinite triangular lattice using these cliques, thereby showing that the chromatic number equals the clique number.

If instead r is odd, in building the maximum-sized clique we move outwards from a single node. The first layer consists of 1 node, the second layer consists of 6 nodes, the third layer consists of 12 nodes, and the $\frac{r+1}{2}^{th}$ layer consists of $3r-3$ nodes. Therefore the total number of nodes in the clique is $\frac{3r^2+1}{4}$. For example, Figure 6 shows that $\omega(G^3) = 7$, and Figure 6 shows that $\omega(G_5) = 19$. Once again, these cliques can be used to tile the triangular lattice, showing that the chromatic number equals the clique number.

Theorem 6 *For any weighted hexagon graph (G,w) and any reuse distance $r > 1$, there is an algorithm that has performance ratio $\frac{18r^2}{3r^2+20}$ if r is even, and $\frac{18r^2+6}{3r^2+21}$ when r is odd.*

Proof: The algorithm uses $\ell\chi(G^r)$ channels (where ℓ will be specified later), and partitions them into $\chi(G^r)$ sets of ℓ channels each. A node v is called *heavy* if $w(v) > \ell$ and *light* otherwise. Each node v of color i in G^r assigns the smallest $min\{w(v), \ell\}$ channels from channel set i. At this point, all light nodes have received enough channels and drop out. Any remaining heavy node now borrows any channel that is unused among its neighbors in G^r.

For any heavy node v, it is easy to see that $H_r(v) \leq w(v) + 6(D_r(G,w) - w(v)) = 6D_r(G,w) - 5w(v)$. For $\ell = \frac{6D_r(G,w)}{\chi(G^r)+5}$, since $w(v) > \ell$, $H_r(v) \leq 6D_r(G,w) - 5\ell = \chi(G^r)\ell$. This means that v has sufficient channels to borrow and to complete its assignment.

From Lemma 10, we know the $\chi(G^r)$ is $\frac{3r^2}{4}$ for even r and $\frac{3r^2+1}{4}$ for odd r, so $H_r(v) \leq \frac{18r^2}{3r^2+20}D_r(G,w)$ for even r and $\frac{18r^2+6}{3r^2+21}D_r(G,w)$ for odd r. It follows from Lemma 1 that $mc(v) \leq H_r(v)$ and thus the performance ratio is at most $\frac{18r^2}{3r^2+20}$ for even r and $\frac{18r^2+6}{3r^2+21}$ for odd r. $\square$

We note that this ratio is always lower than 6 for any bounded value of r. For the particular reuse distances of 2, 3, and 4, this algorithm gives us performance ratios of 2.25, 3.5, and approximately 4.24 respectively. The cluster partitioning algorithm described in the next section has a better performance ratio for all values of reuse distance greater than 2, in addition to having the advantage of not requiring reassignment of calls in the online case.

5.2 Online algorithms

The best known performance ratio for channel assignment for reuse distance $r > 3$ is achieved by an algorithm called *cluster partitioning* in [9]. The key idea is to partition the graph into clusters which are maximal cliques, as with FA. However, unlike FA, where identical sets of channels are assigned to corresponding cells in different clusters, here, sets of D channels are assigned to entire clusters, in such a way that any pair of clusters containing cells that are within distance $r - 1$ are assigned different sets. Calls arising in any cluster are assigned channels from its nominal set of channels. Furthermore, it turns out that it is possible to color the clusters with 4 colors such that any two clusters that have nodes within distance $r - 1$ of each other get different colors. Thus four sets of channels suffice, which implies a performance ratio of 4 for the algorithm.

However, we observe that for reuse distance $r = 2, 3$, the clusters can be 3-colored (see Figure 17). Thus, $3D$ channels suffice, and a performance ratio of 3 for the algorithm follows. The existence of a 3-coloring for other values of reuse distance is unclear.

It is easy to show an upper bound on the performance of the greedy algorithm for arbitrary reuse distance.

Theorem 7 *For any weighted hexagon graph (G, w) and any reuse distance $r > 1$, the greedy algorithms G-R and G-NR have performance ratio at most 6.*

Proof: Since for any node v, the set $\{v\} \cup N_r(v)$ can be covered by 6 cliques, each with weight at most $D_r(G, w)$, we know that $H_r(v) \leq 6D_r(G, w)$. It follows from Lemma 1 that $mc(v) \leq 6D_r(G, w)$. Since $D_r(G, w)$ is a lower bound on the number of channels required, the algorithm has performance ratio at most 6. □

6 Discussion

We have shown that the competitive ratio of the greedy DCA algorithms proposed in [3, 5, 15] are between 2.5 and 3 for reuse distance 2 and lie between 3 and 4 for reuse distance 3. Thus the worst-case performance of the greedy strategy with no reassignments is no worse than the worst-case performance than FA for both values of reuse distance. For reuse distance 2, there is no known algorithm that has better worst-case performance than the greedy algorithm or FA among algorithms that do not perform reassignment of channels. For reuse distance 3, the cluster partitioning algorithm of Jordan and Schwabe [9] can be seen to have competitive ratio 3, which is at least as good as that of the greedy algorithm.

In the static setting, or alternatively while considering online algorithms where reassignment of channels is permitted, the greedy algorithm appears to have poor performance.

We showed that G-R, the greedy algorithm with reassignments, is outperformed by the borrowing strategies in [4, 6, 8] for reuse distance 2 as well as 3.

A number of open problems remain. The most interesting problem is to find an optimal online algorithm for channel assignment that does not perform reassignments. The best known lower bound on the competitive ratio of such algorithms is 2, while the greedy algorithm has a competitive ratio of at least 2.5. Another problem is to find tight bounds on the competitive ratio of both versions of the greedy algorithm for reuse distance $r \geq 3$. In the offline setting, for reuse distance 3, we use a particular base coloring of the underlying graph for synchronization purposes. This coloring is used to determine in what order the nodes get channels in the offline greedy algorithm. Changing this ordering could, in principle, make a difference to the number of channels used. Perhaps a particular ordering may yield the best result. Finally, it is clear that hexagon graphs with reuse distance 3 contain graphs other than odd cycles as induced sub-graphs, for which the weighted chromatic number is greater than the weighted clique number. An investigation of such induced sub-graphs may yield better lower bound results than those listed in Tables 1 and 2.

References

[1] I. Caragiannis, C. Kaklamanis, and E. Papaionnou. Efficient online communication in cellular networks. In *Symposium on Parallel Algorithms and Architecture*, 2000.

[2] J. C.-I. Chuang. Performance issues and algorithms for dynamic channel assignment. *IEEE Journal on Selected Areas in Communications*, 11(6):955–963, 1993.

[3] D. C. Cox and D. O. Reudink. Dynamic channel assignment in two dimension large-scale mobile radio systems. *Bell Sys. Tech. J.*, 51:1611–28, 1972.

[4] T. Feder and S. M. Shende. Online channel allocation in FDMA networks with reuse constraints. *Inform. Process. Lett.*, 67(6):295–302, 1998.

[5] C. L. I and P. H. Chao. Local packing-distributed dynamic channel allocation at cellular base station. *Proceedings of GLOBECOM*, 1993.

[6] J. Janssen and K. Kilakos. Adaptive multicolourings. *Combinatorica*, 20(1):87–102, 2000.

[7] 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.

[8] J. Janssen, D. Krizanc, L. Narayanan, and S. Shende. Distributed online frequency assignment in cellular networks. *J. of Algorithms*, 36:119–151, 2000.

[9] S. Jordan and E. J. Schwabe. Worst-case performance of cellular channel assignment policies. *Wireless Networks*, 2:265–275, 1996.

[10] I. Katzela and S. Naghshineh. Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey. *IEEE Personal Communications*, pages 10–31, 1996.

[11] V. H. MacDonald. Advanced mobile phone service: The cellular concept. *Bell Systems Technical Journal*, 58(1), 1979.

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

[13] C. McDiarmid and B. Reed. Colouring proximity graphs in the plane. *Discrete Mathematics*, 199:123–137, 1999.

[14] L. Narayanan and S. Shende. Static frequency assignment in cellular networks. *Algorithmica*, 29:396–409, 2001.

[15] R. Prakash, N. Shivaratri, and M. Singhal. Distributed dynamic channel allocation for mobile computing. In *Principles of Distributed Computing*, pages 47–56, 1995.

[16] J. B. Punt and D. Sparreboom. Mathematical models for the analysis of dynamic channel selection for indoor mobile wireless communications systems. *PIMRC*, E6.5:1081–5, 1994.

[17] M. Serizawa and D. Goodman. Instability and deadlock of distributed dynamic channel allocation. *Proc. 43rd IEEE VTC*, pages 528–31, 1993.

[18] J. van den Heuvel, R. A. Leese, and M. A. Shepherd. Graph labeling and radio channel assignment. *Journal of Graph Theory*, 29:263–283, 1999.

[19] M. Zhang and T.-S. P. Yum. Comparisons of channel assignment strategies in cellular mobile telephone systems. *IEEE Transactions in Vehicular Technology*, 38:211–215, 1989.

Mailing address of contact author:

Lata Narayanan
Department of Computer Science
Concordia University
1455 de Maisonneuve Blvd. West
Montreal, Quebec H3G 1M8
Canada
Phone: (514) 848-2424 x3029
Email: lata@cs.concordia.ca

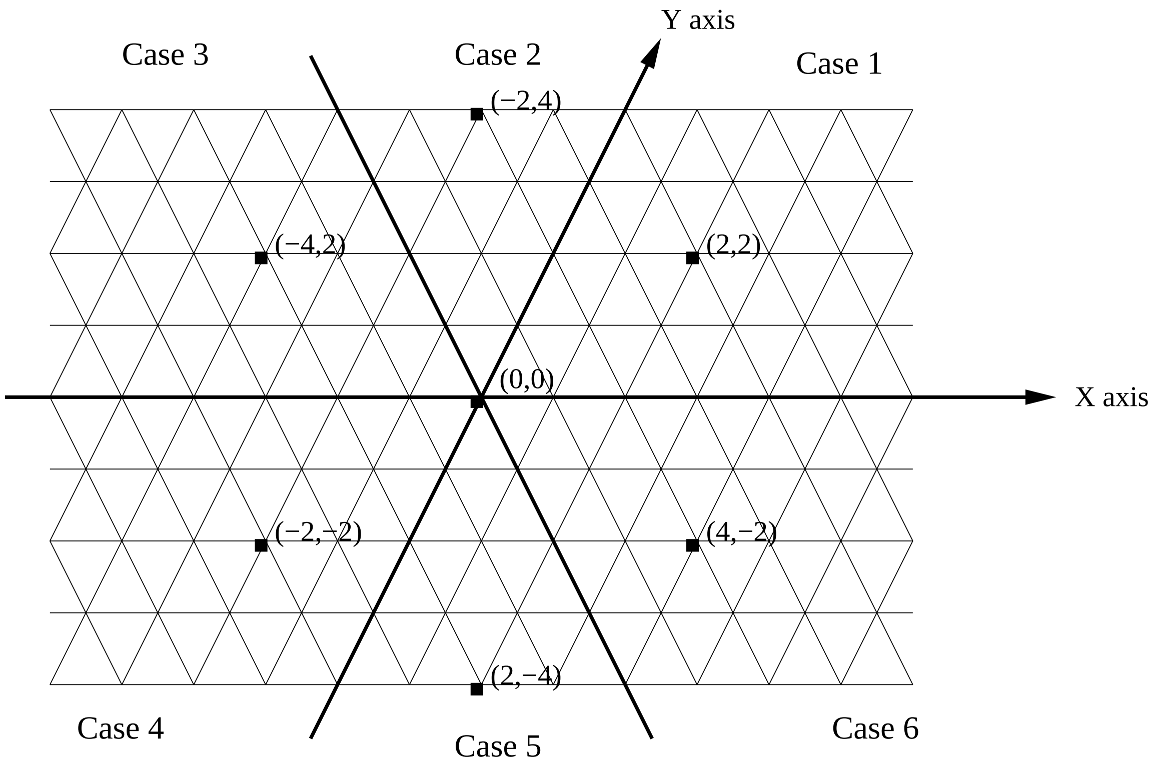


Figure 1: Distances between nodes

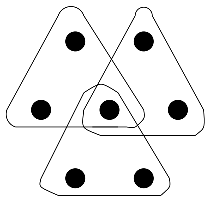


Figure 2: $N_2(v) \cup \{v\}$ is covered by 3 cliques.

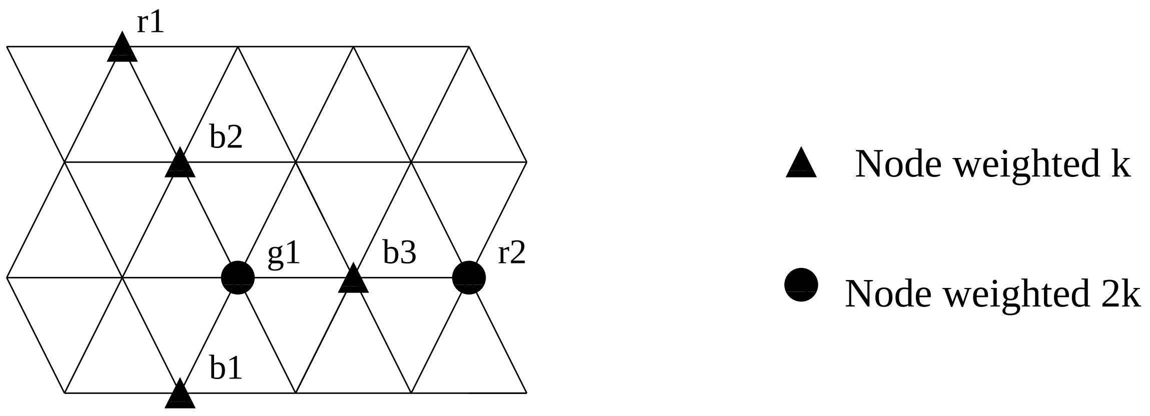


Figure 3: An example where the greedy algorithm uses $5D_2(G, w)/3$ channels.

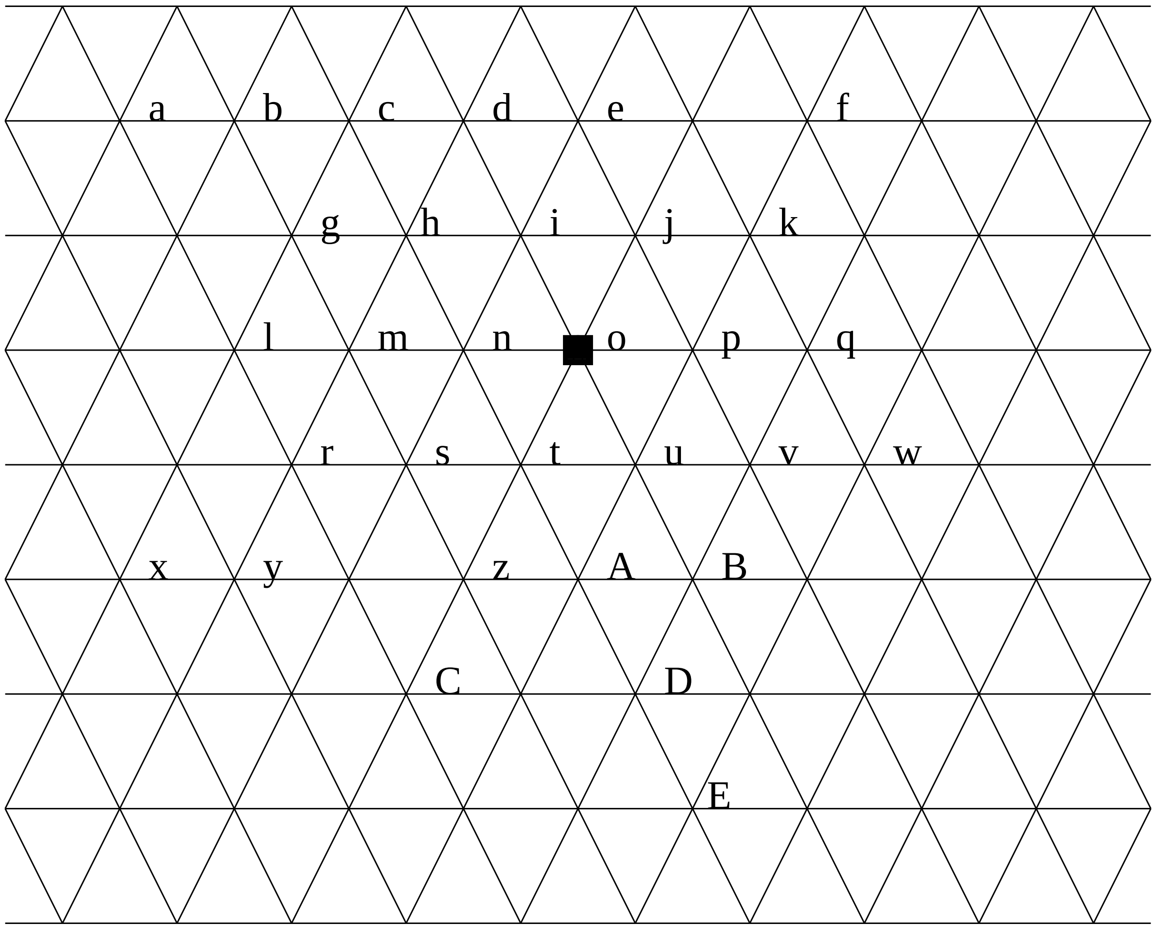


Figure 4: Worst case for G-NR for reuse distance 2

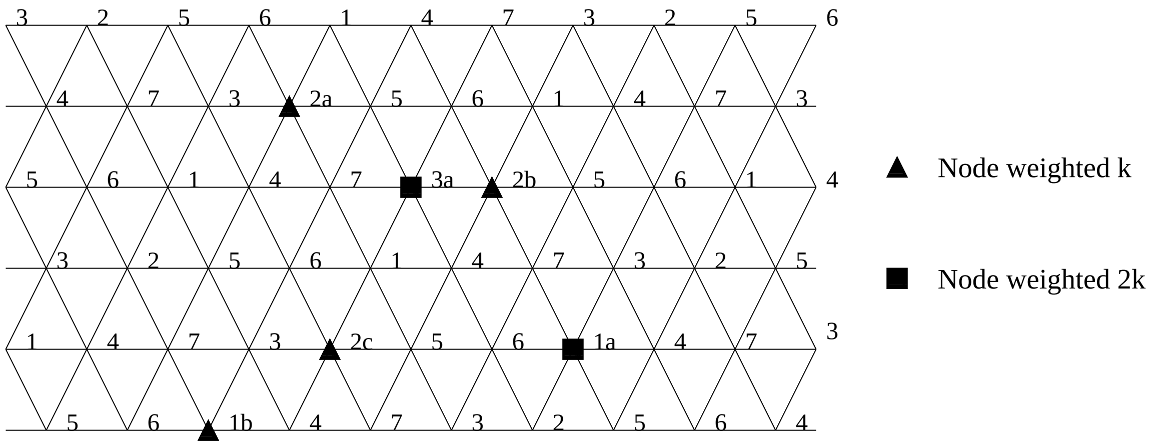


Figure 5: An example showing that a 3-node can use $5D_3/3$ channels.

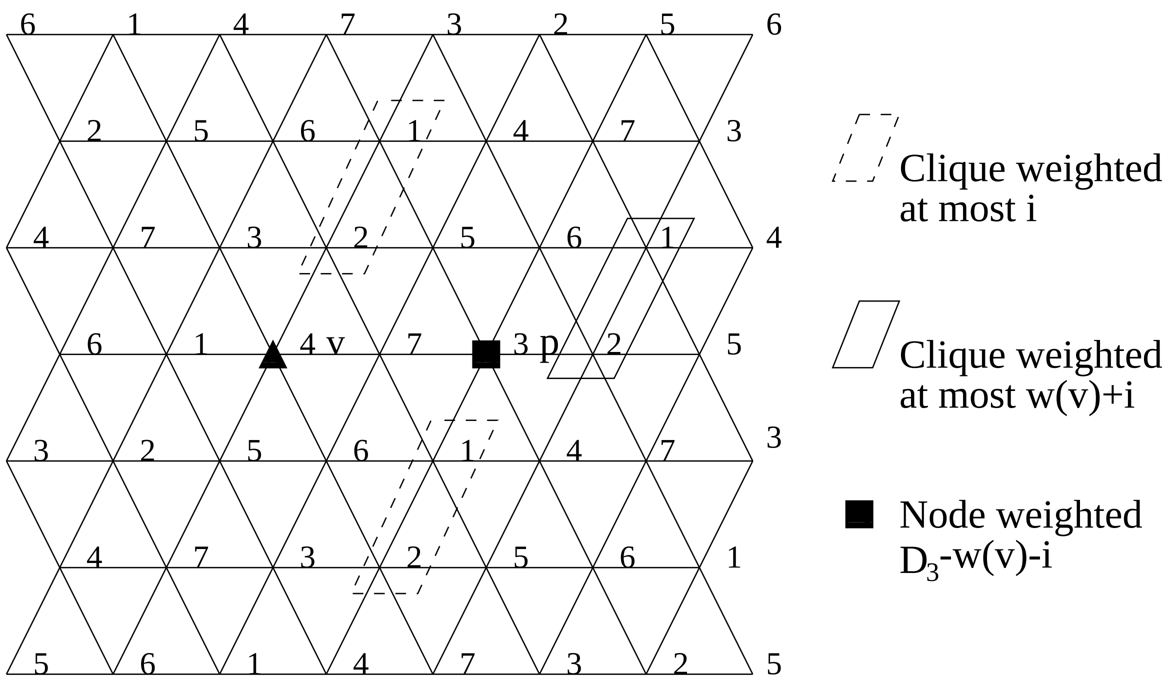


Figure 6: Deriving the bound on $H_3(p)$.

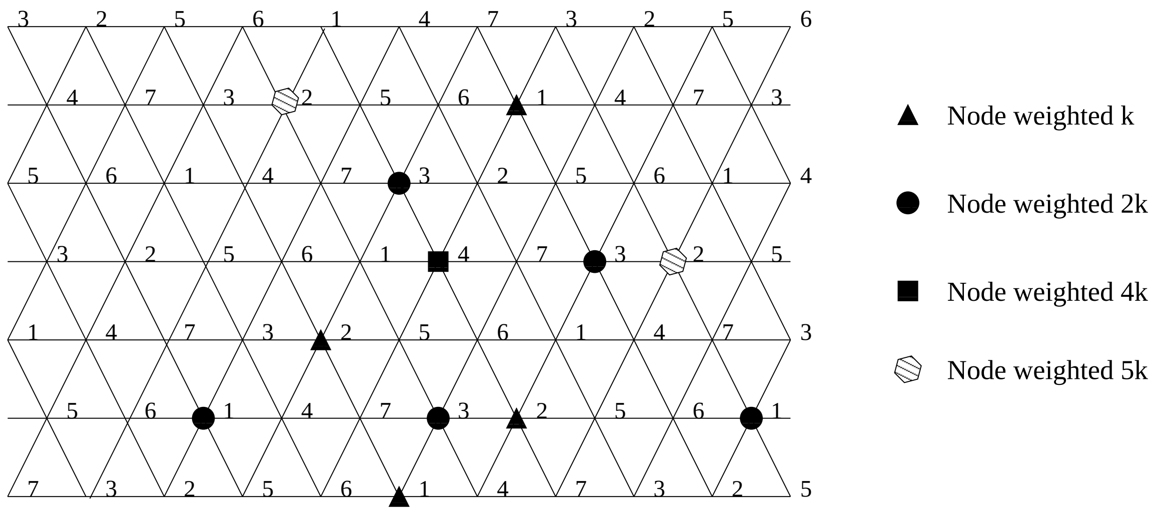


Figure 7: An example showing that a 4-node can use $13D_3/7$ channels.

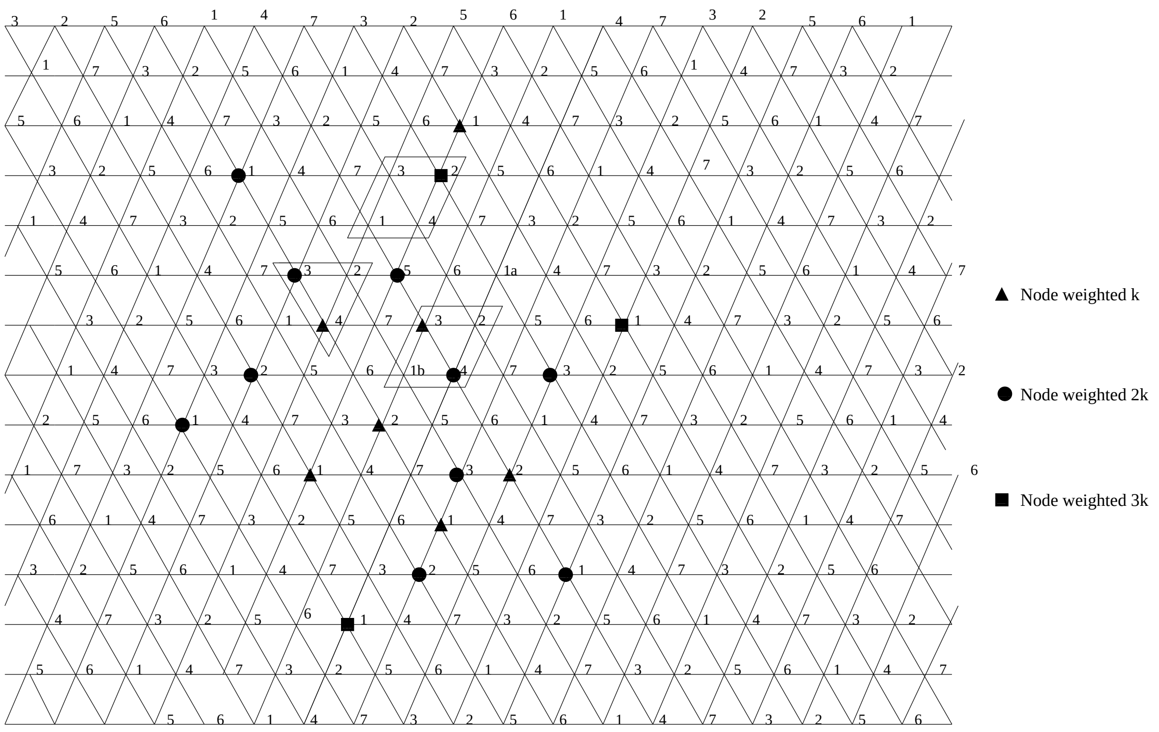


Figure 8: An example showing that a 5-node can use $11D_3/5$ channels. Notice that $N_3(v)$ can be covered by 3 cliques and node 1_a. A similar cover using 3 cliques and node 1_b can also be constructed.

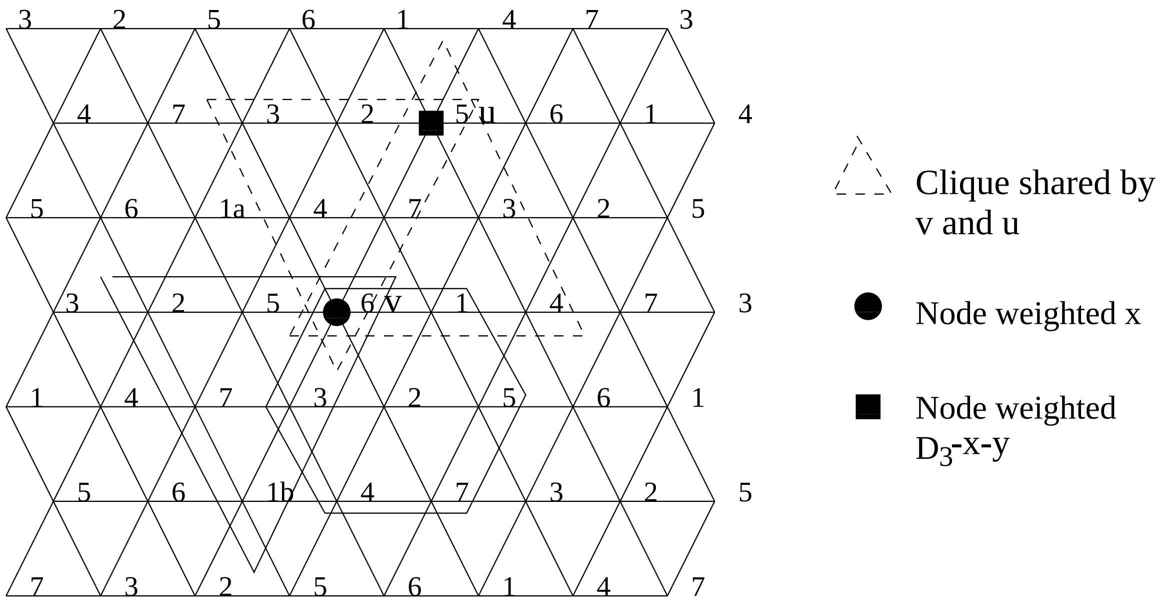


Figure 9: Deriving the bound on $H_3(u)$ when $w(1_a) \leq w(1_b)$.

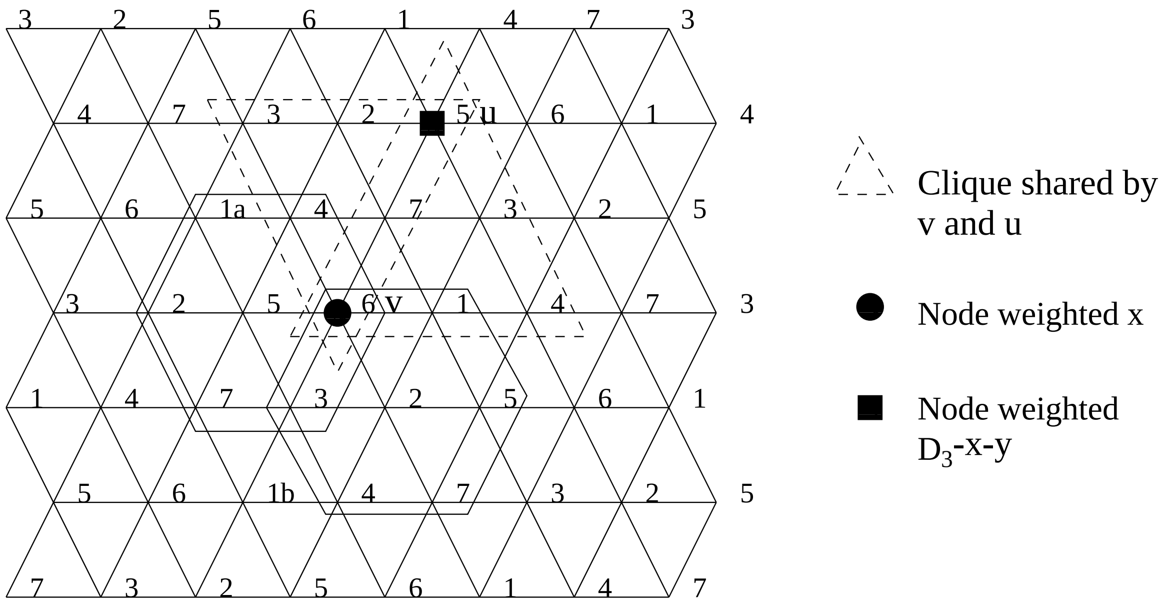


Figure 10: Deriving the bound on $H_3(u)$ when $w(1_a) \geq w(1_b)$.

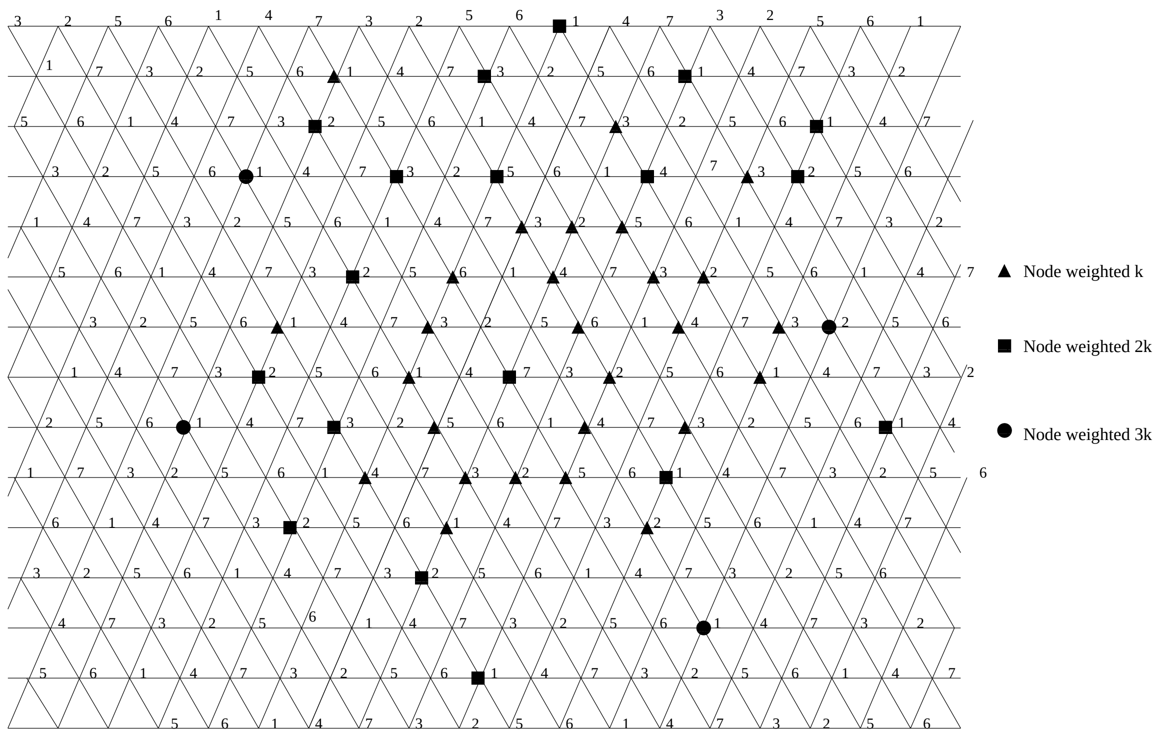


Figure 11: An example showing that a 7-node can use $12D_3/5$ channels.

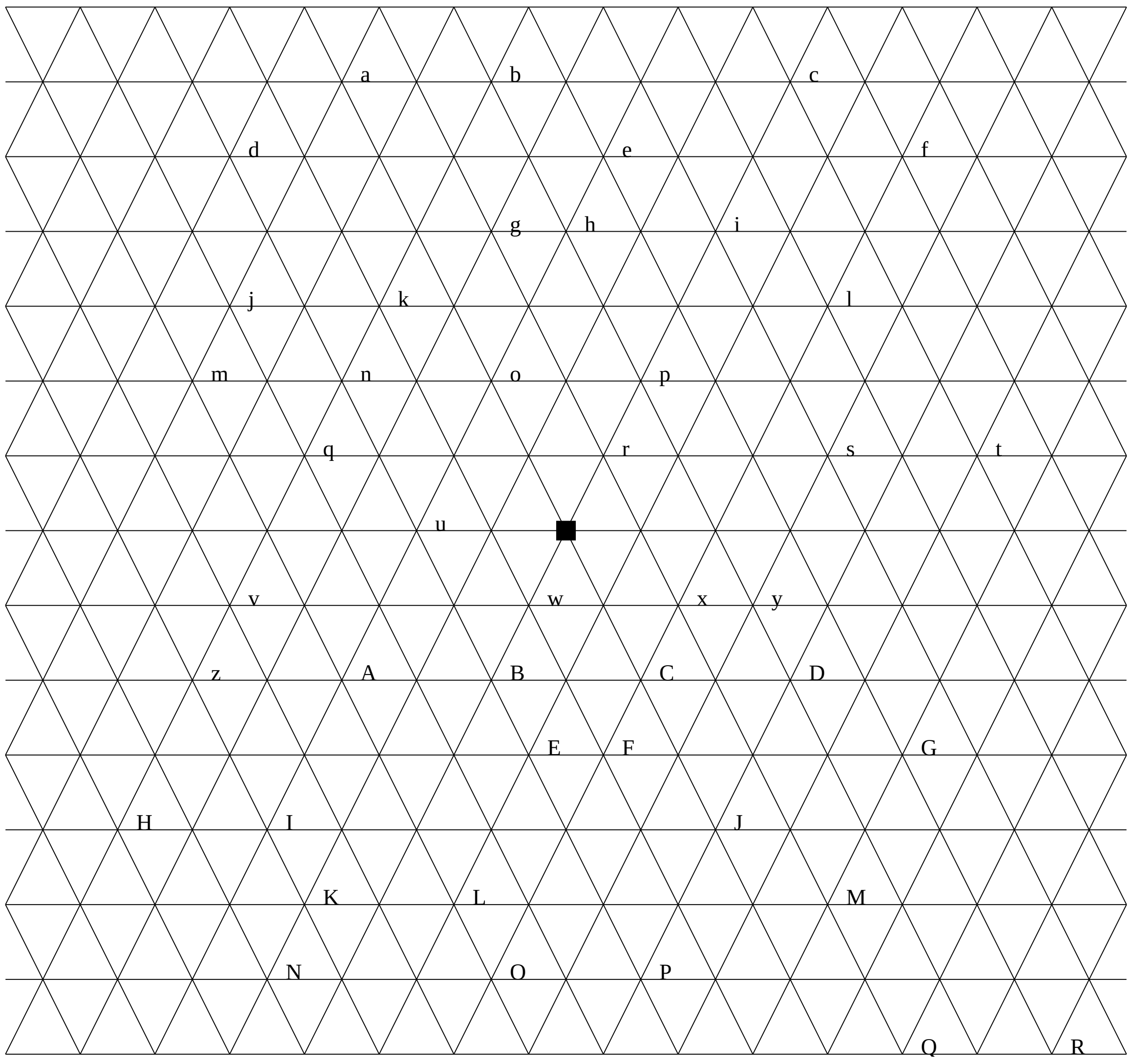


Figure 12: Graph in which nodes C and o are forced to use channels 7 and 8 respectively. See Tables 3 and 4.

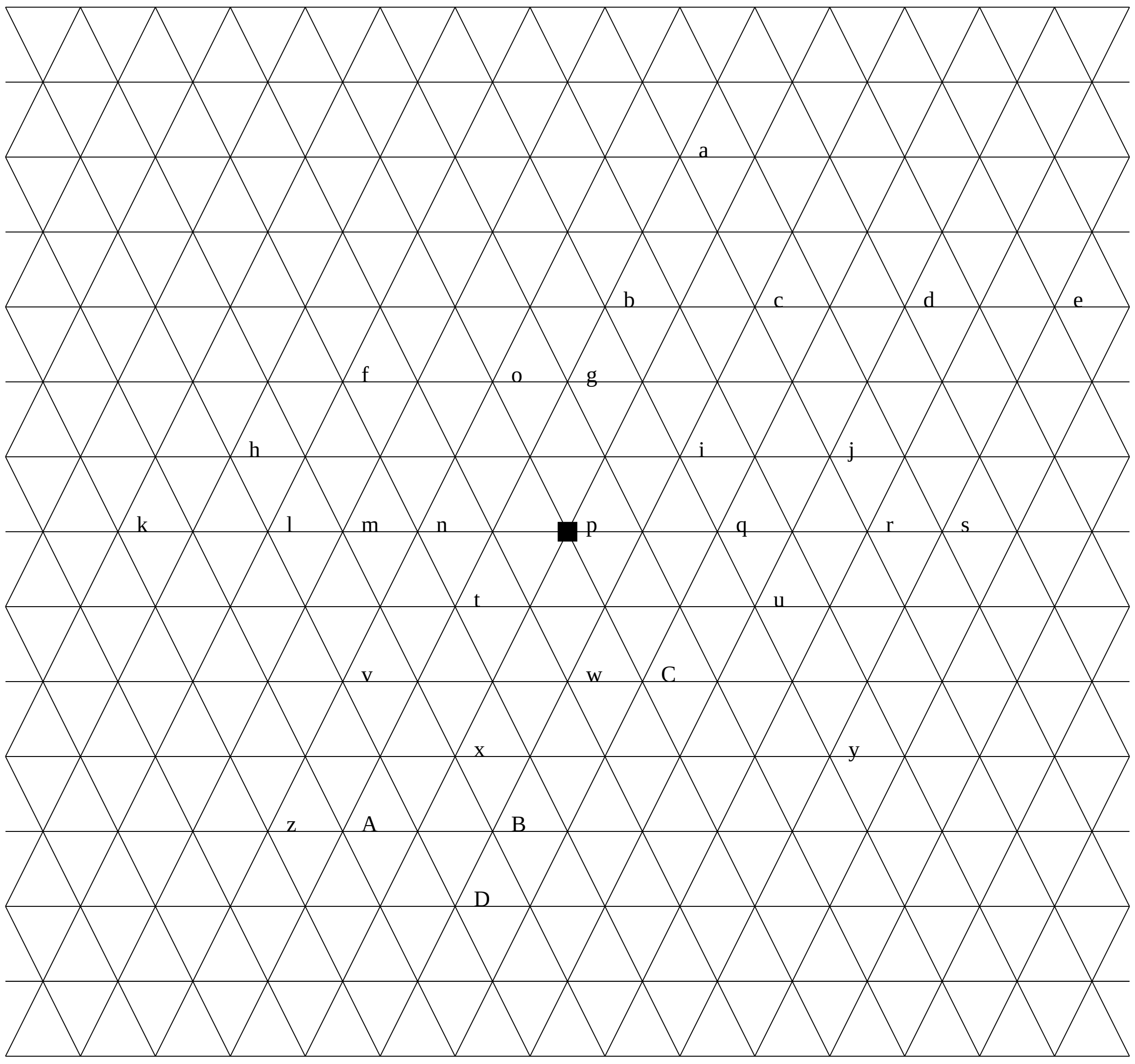
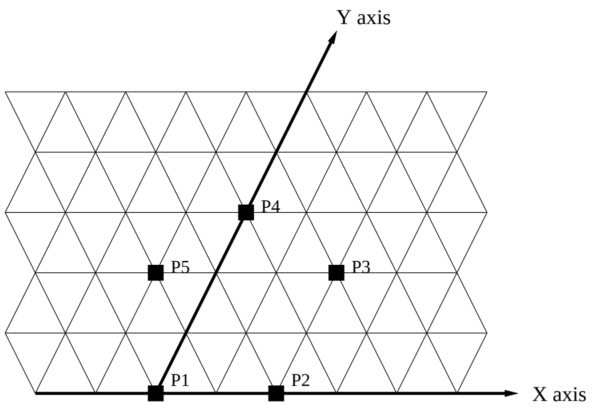


Figure 13: Graph in which node p is forced to use channel 9, assuming that nodes o and C are using channels 8 and 7 respectively. See Table 5.



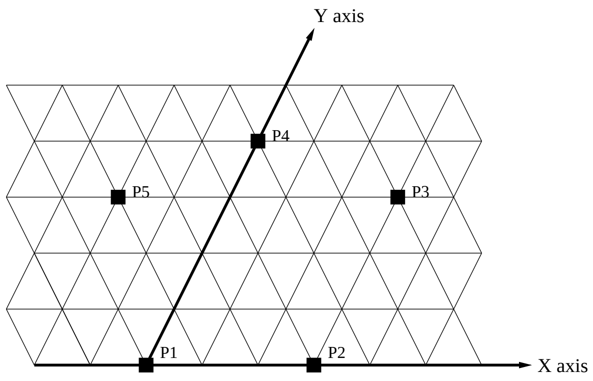


Figure 14: (a) 5-cycle for reuse distance 3, and (b) 5-cycle for reuse distance 4.

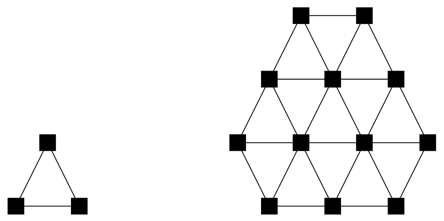


Figure 15: Maximum-sized cliques of hexagon graph with reuse distance (a) 2 and (b) 4.

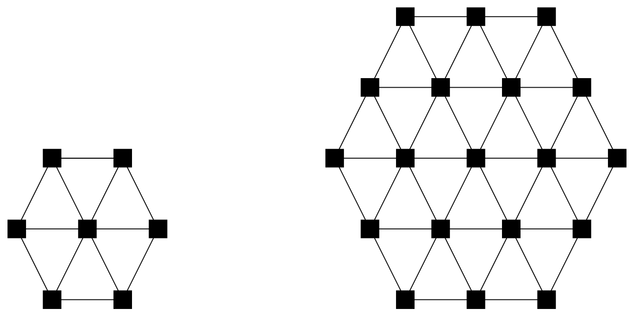


Figure 16: Maximum cliques of hexagon graph with reuse distance (a) 3 and (b)5.

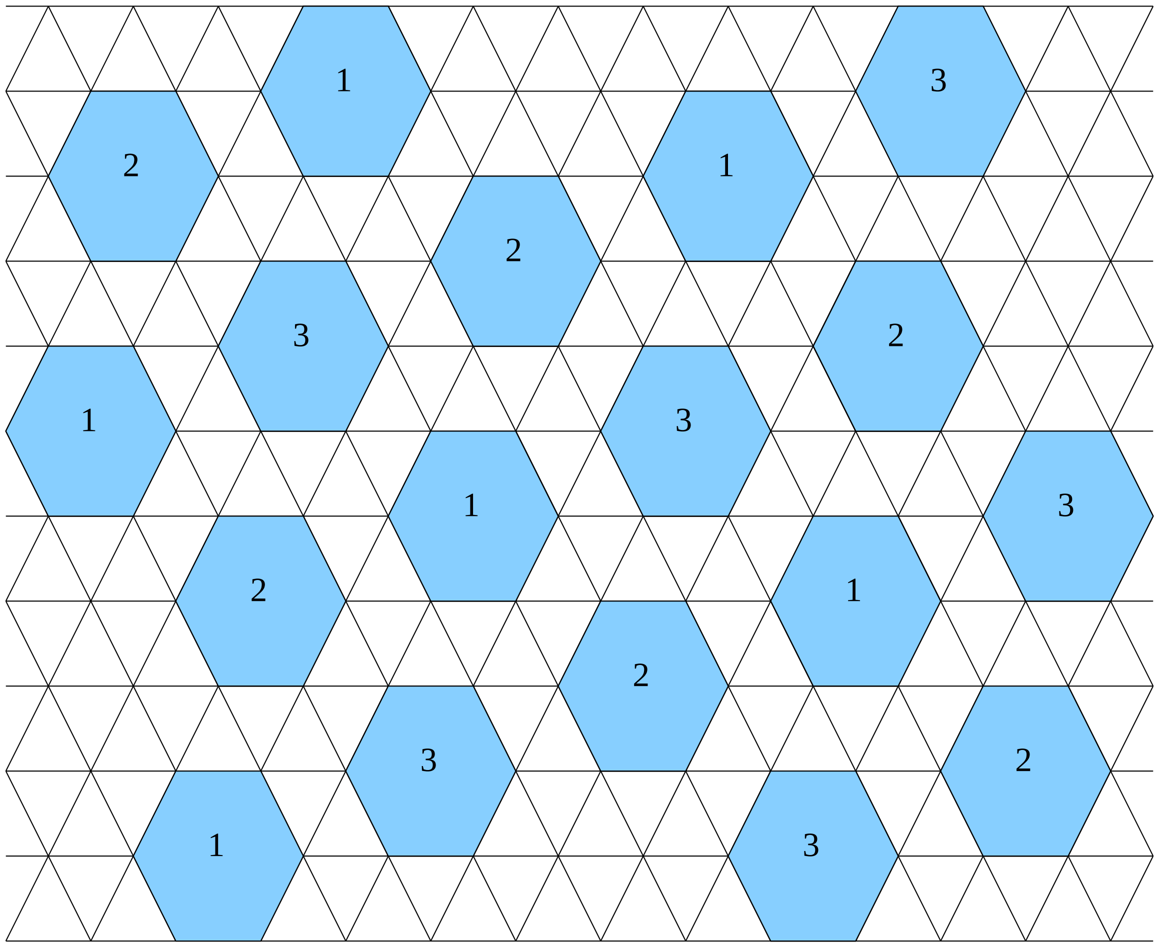


Figure 17: Clusters can be 3-colored for odd values of reuse distance (reuse distance 3 is shown here).