International Journal of Research Studies in Computing
2012 October, Volume 1 Number 2, 21-26

Strong fault-tolerant conflict-free coloring (Strong FTCF Coloring) for intervals

ISSN: 2243-772X
Online ISSN: 2243-7797

OPEN ACCESS

Singhal, Vinay Kumar ✉
Department of MCA, Meerut Institute of Engineering & Technology, Meerut, Uttar Pradesh, India
(singhalmca04@gmail.com)
Rastogi, Akanksha
Department of MCA, Meerut Institute of Engineering & Technology, Meerut, Uttar Pradesh, India
(akanksharastogi.mca@gmail.com)

Received: 10 June 2012 **Revised**: 20 June 2012 **Accepted**: 23 June 2012
Available Online: 24 June 2012 **DOI:** 10.5861/ijrsc.2012.137

Abstract

Given a set of points P, a conflict-free coloring of P is an assignment of colors to points of P, such that there exist a point p in any subset of P whose color is distinct from all other points in that subset of P. This notion is motivated by frequency assignment in wireless cellular networks: one would like to minimize the number of frequencies (colors) assigned to base stations (points), such that within any range, there is no interference. Also base stations in cellular networks are often not reliable. Some base station may fail to function properly or get faulted. This leads us to study fault-tolerant CF-colorings: colorings that remain conflict-free even after some objects are deleted from P. We provide a strong fault-tolerant conflict-free coloring (Strong-FTCF coloring) framework for intervals of points such that if some colors gets faulted, the whole system will remain working. We can say that the system is working properly if some of the colors are deleted. Our algorithm uses O(log n) colors with high probability.

Keywords: frequency-assignment problem; cellular network; intervals; conflict-free coloring; strong fault-tolerant

© The Authors / Attribution-NonCommercial-NoDerivs CC BY-NC-ND

Strong fault-tolerant conflict-free coloring (Strong FTCF Coloring) for intervals

1. Introduction

Consider a cellular network consisting of a set of base stations, where the signal from a given base station can be received by clients within a certain distance from the base station. In general, the regions covered by the base stations may overlap. For a client, this may lead to interference of the signals. Client cannot get the signals that work for him. Thus one would like to assign frequencies to the base stations such that for any client within reach of at least one base station, there is a base station within reach with a unique frequency (among all the ones within reach). The goal is to minimize the number of assigned frequencies because spectrum is limited and costly.

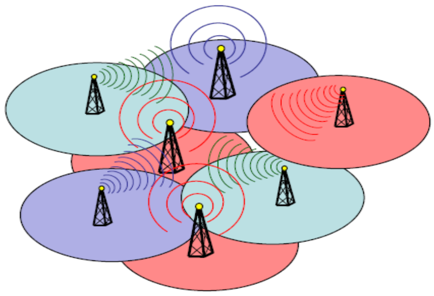

Figure 1. Assignment of frequencies for a set of cellular networks

Traditionally, this problem was considered as an application of usual graph coloring. The base stations are the nodes of a graph and two nodes are connected by an edge if the ranges of their associated base stations intersect. If each node (base-station) is assigned a color (frequency) such that no two neighboring nodes are colored with the same color, certainly every client can be served by some base station. However in, it was shown that this is not the right way to look at the problem since even an optimal coloring may use too many colors (Even, Lotker, Ron, & Smorodinsky, 2003). Instead, we have to color the base station's regions such that each point in the plane is covered only once by at least one color. If the sending areas of all base stations are disks of the same radius, it was shown that log (n) colors (n is the number of base stations) always suffice to find such a **conflict-free** coloring. However, there are configurations where the algorithm of Even et al. (2003), needs log (n) colors although a small constant of colors would be enough.

In this paper, we start by defining the detailed definition of conflict-free coloring in *section II*. In *section III*, we define the advantages and concept of conflict free coloring with respect to intervals. A randomized Strong-FTCF coloring algorithm for intervals is presented in *Section IV*. Finally, *Section V* contains conclusion and *Section VI* contains references.

2. Conflict-free Coloring

The study of the frequency assignment problem was initiated in Even et al. (2003), and continued in a series of papers Pach and Toth (2003), Fiat et al. (2005), Har-Peled and Smorodinsky (2005), Chen (2006), Smorodinsky (2006), and Chen, Kaplan, and Sharir (2009). A formal and more general definition of CF-coloring is given as follows:

Definition 1 (Range Space): *Range Space is defined by a pair (X, R) where X is a set of vertices, and R is a family of subsets of X. The subsets in R are called ranges.*

Definition 2 (CF-coloring): *Let (X, R) be a range space. A coloring* $\chi: R \rightarrow N$*, which maps R to the set of natural numbers N, is conflict-free if for every vertex* $x \in X$*, there is an* $r \in R$ *such that* $x \in r$ *and* $\chi(r) \neq \chi(r')$ *for all other ranges r' containing x.*

In the literature, the frequency allocation problem was also modeled as a dual of the CF-coloring problem. The dual problem treats the base stations as vertices and ranges as the subsets of base stations within some geometric regions. The goal of the dual problem is to find a conflict-free coloring of the vertices such that for any range, there is always a vertex among the vertices in the range with a unique color.

Definition 3 (Dual CF-Coloring): *Let* (X, R) *be a range space. A coloring* $\varphi: X \rightarrow N$ *is conflict-free if for every range* $r \in R$*, there exists a vertex* $x \in r$ *such that* $\varphi(x) \neq \varphi(x')$ *for all other* $x' \in R$.

It can be formally described as follows. Let $P \subseteq \mathbb{R}^2$ be a set of points and R be a set of ranges (e.g. the set of all discs or rectangles in the plane). A *conflict-free* coloring (or CF-coloring) of P with respect to the range R is an assignment of a color to each point $p \in P$ such that for any range $T \in R$ with $T \cap P \neq 0$, the set $T \cap P$ contains a point of unique color. Naturally, the goal is to assign a conflict-free coloring to the points of P with the *smallest* number of colors possible.

Related work: The CF-coloring problem for points with respect to disks was studied by Even et al. (2003). They showed that for this setting one can always find a CF-coloring using O(log n) colors, which is tight in the worst case. They also studies CF-colorings for points with respect to disks. Har-Peled and Smorodinsky (2005) extended those results by considering other range spaces. In particular, they gave sufficient conditions for a range space to allow a CF-coloring with few colors. Recently, Smorodinsky (2006) improved several results from Even et al. (2003) by providing deterministic coloring algorithms. Chen et al. (2009) and Bar-Noy et al. (2006a) studied various CF-coloring problems in an on-line setting. Here the objects are given one by one, and each object has to be colored when it arrives, in such a way that the coloring remains conflict-free at all times. While, Ali Abam et al. (2008) studied k-fault-tolerant CF-coloring (k-FTCF-coloring) i.e. a coloring that remains conflict-free after an arbitrary collection of k objects is deleted from the set. Thus a k-FTCF-coloring for k = 0 is simply a standard CF-coloring. Such colorings for points with respect to disks were also studied by Abellanas et al. (2005) who showed that any set of n points admits a k-FTCF coloring with respect to disks that uses O(k log n) colors. More importantly, Ali Abam et al. (2008) provide an algorithm that consists of two phases:

Phase 1: They process the intervals one by one, as follows. To process I_j, they check if every point in I_j is contained in at least $k + 1$ intervals from the current set $\mathcal{I}$. If this is the case, they assign color 0 to I_j and remove I_j from $\mathcal{I}$, otherwise I_j stays in $\mathcal{I}$ and does not get a color yet. They claim that after Phase 1, every point q is contained in at most $2k + 2$ intervals. Indeed, the $k + 1$ intervals containing q and extending the farthest to the left, and the $k + 1$ intervals containing q and extending the farthest to the right, must cover every other interval containing q. Hence, any such other interval will be removed in Phase 1.

Phase 2: Now they color the remaining intervals, only using colors from the set $S = \{1, \ldots, \lceil (3k + 3)=2 \rceil + 1\}$. To this end, they sweep from left to right. When the sweep arrives at the left endpoint of an interval I, they assign a color to I, as follows. Let S_I be a set of forbidden colors for I in the sense that if they assign one of them to I, then the collection of intervals colored so far is not a k-FTCF anymore. They take an arbitrary color from $S \backslash S_I$ and assign it to I.

3. Conflict-free Coloring for Intervals

Chen et al. (2009) considered the special case of the problem where the hypergraph is defined as follows: Vertices are identified by points that lie on a line and E consists of all subsets of V defined by intervals intersecting at least one vertex. Conflict-free coloring for intervals is important because it can model assignment of frequencies in networks where the agents' movement is approximately one-dimensional (unidimensional), e.g., the cellular network that covers a single long road and has to serve agents that move along this road. Also, conflict-free coloring for intervals plays a role in the study of conflict free coloring for more complicated range spaces [please see Even et al. (2003) for more details]. The static version of the problem, where the n points are to be colored simultaneously, is solved optimally in Even et al. (2003) with $\lfloor \lg n \rfloor + 1$ colors.

The problem becomes more interesting when the vertices are given online by an adversary. Namely, at every given time step $t \in \{1, \ldots, n\}$, a new vertex $v_t \in V$ is given and the algorithm must assign v_t a color such that the coloring is a conflict-free coloring of the hypergraph that is induced by the vertices $V_t = \{v_1, \ldots, v_t\}$. Once v_t is assigned a color, that color cannot be changed in the future. This is an online setting, in which the algorithm has no knowledge of how vertices will be requested in the future. For this version of the problem, in the case of intervals, the best known deterministic algorithm is from Fiat et al. (2005) and uses $O(\log^2 n)$ colors in the worst case. That algorithm requires $\Omega(\log^2 n)$ colors on some inputs. Recently, randomized algorithms that use $O(\log n)$ colors with high probability have been obtained (Fiat et al., 2005; Bar-Noy et al., 2006a). All of these algorithms assume the slightly weaker *oblivious* adversary model, in which the adversary has to commit on a specific input sequence before revealing the first vertex to the algorithm without knowing the random bits that the algorithm is going to use and the expected number of colors is analyzed. The randomized model can be seen as a relaxation of the strict deterministic model: some power is taken from the adversary, or equivalently given to the algorithm, in order to use just a logarithmic number of colors.

Another such relaxation, introduced in Bar-Noy et al. (2006b), is to give extra information to the algorithm about where each requested point will end up in the final coloring (the 'absolute positions' model). Furthermore, in the absolute positions model; the best algorithm in Bar-Noy et al. (2006b) uses $3\lceil\log_3 n\rceil \approx 1.89 \lg n$ colors. Other such relaxations are given in Bar-Noy et al. (2006b) (coloring with respect to rays) and Schiermeyer, Tuza, and Voigt (2000) (online ranking of paths). In this paper they introduce yet another relaxation, the recoloring model, in which the algorithm is allowed to recolor some of the points. An interesting question is to come up with $O(\log n)$ algorithms that rely as little as possible on their extra power (as few random bits as possible, as few recolorings as possible). Towards that goal, in a unified framework, they provide the best known results: randomized algorithms that expectedly use a logarithmic number of random bits, and a recoloring algorithm that recolors at most one point per request.

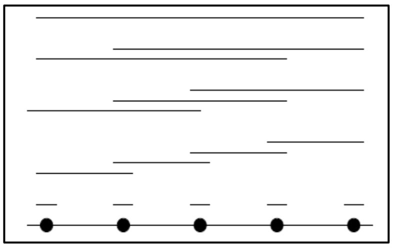

Figure 2. Sets of possible intervals for a given number of points

To understand the concept of intervals, let us consider an example given below:

For any given set of points $P = \{a_1, a_2, a_3, a_4\}$ we have following intervals-

$I_1 = \{(a_1)\ (a_2)\ (a_3)\ (a_4)\}$
$I_2 = \{(a_1, a_2)\ (a_2, a_3)\ (a_3, a_4)\}$
$I_3 = \{(a_1, a_2, a_3)\ (a_2, a_3, a_4)\}$
$I_4 = \{(a_1, a_2, a_3, a_4)\}$
For all intervals we have to maintain the conflict-free coloring property.

There are various research studies being conducted on this topic. Fiat et al. (2005) shows that a natural and simple (deterministic) approach may perform rather poorly, requiring $\Omega(\sqrt{n})$ colors in the worst case. They derive two efficient variants of this simple algorithm. The first is deterministic and uses $O(\log^2 n)$ colors, and the second is randomized and uses $O(\log n)$ colors with high probability.

4. Strong Fault-Tolerant CF coloring for Intervals

Base stations in cellular networks are often not completely reliable: every now and then some base station may (temporarily or permanently) fail to function properly or get faulted. This leads us to study fault-tolerant

CF-colorings: colorings that remain conflict-free even after some objects are deleted from the set of given points P. More precisely, a strong-fault-tolerant CF-coloring (Strong-FTCF-coloring) is a coloring that remains conflict-free after an arbitrary collection of many objects is deleted from P. Such colorings for points with respect to disks were studied by Abellanas et al. (2005), who showed that any set of n points admits a k-FTCF coloring with respect to disks that uses O(k log n) colors. We studied S-FTCF coloring with respect to intervals and obtain upper and lower bounds on the worst-case number of colors needed in fault-tolerant colorings.

Lemma 1: A CF coloring is said to be good fault-tolerant CF coloring if it contains at least two unique colors.

Proof: The correctness of the above lemma can be proofed very easily. Consider we have a CF coloring that has only one unique color. If, for example, that unique color is faulted then the whole system is corrupted. That's why we need at least two unique colors so that if one color faulted the whole system does not corrupt.

Lemma 2: Again a CF coloring is said to be good fault-tolerant CF coloring if a color is repeated when two different color are in between them.

Proof: Consider we have a CF coloring in which a color is repeated when only one color is in between them. If, for example, that middle color is faulted then the whole system is corrupted. That's why we need at least two different colors when a color is repeated so that if one middle color faulted the whole system does not corrupt.

4.1 Strong FTCF Coloring Algorithm

Now, we present a randomized Strong FTCF coloring algorithm that uses at least $\lceil \log_2 n + 1 \rceil$ colors with high probability.

Initialize: n = number of points to be colored. i = 1, j=3

1. Color first point of n with 1.

2. Color next point with j and put the value of j in i.

3. Decrease the value of i with 1 and Color next points with i until i = 1

4. Increase the value of j by 1.

5. Repeat Steps 2 to 4 until all the points are colored.

Figure 3. Strong FTCF coloring algorithm

For example: Let, we have n = 6 number of points to be colored, i = 1, j = 3.

1. i = 1, CFC = 1.
2. j = 3, CFC = 13 and i = 3
3. i = 2, CFC = 1321
4. j = 3+1=4
5. j = 4, CFC = 13214
6. i = 3, CFC = 132143
7. Fault Tolerance CFC = 132143

Here if any color is get faulted e.g. 4, the whole system work normally as 13213 holds the conflict free property i.e. there is at least one unique color for all intervals.

Proposition 3: The above algorithm gives a strong CF coloring in which the probability of fault occurrence is very low.

Theorem 4: For any set of n points on the real line, there is a Strong Fault Tolerant CF coloring with respect to points for its intervals that uses at least $\lceil \log_2 n + 1 \rceil$ colors and the complexity of the above algorithm is O(log n).

Proof: To prove this lower bound, consider the algorithm described above. For a given set of n points, we have n intervals. For Strong FTCF coloring, we require at least two unique colors so that if one fails, second color maintains the uniqueness in the coloring. We know that the coloring of n points with respect to intervals uses at least $\log_2 n$ colors (Bar-Noy, 2006a). This means the above algorithm uses one color more that $\log_2 n$ i.e. it uses at least $\lceil \log_2 n + 1 \rceil$ color in tight cases.

This Strong FTCF coloring algorithm may fails only in two conditions given below: a) Both unique colors get faulted at one time, and b) Two neighbor colors get faulted. The probability of such occurrence is very low otherwise algorithm works well in all other conditions.

5. Conclusion and Future Work

Base stations play a very important role in today's world of mobile communication. If some base stations get faulted, due to inference, the communication or client will suffer. So as to avoid such situation our algorithm is very much capable. It uses only O(log n) colors in the worst cases. According to our algorithm the probability of failure of the system is very low and reliability is high. One can find an algorithm that gives conflict-free coloring such that it remains conflict-free even any number of colors are deleted from the set with few colors.

6. References:

Abellanas, M., Bose, P., Garca, J., Hurtado, F., Nicolas, M., & Ramos, P. A. (2005). On properties of higher-order Delaunay graphs with applications. *Abstract of the 21st European Workshop on Computational Geometry*, 119-122.

Ali Abam, M., de Berg, M., & Poon, S. H. (2008, August 13-15). *Fault-tolerant conflict-free colorings*. Paper presented in the CCCG 2008, Montreal, Quebec.

Bar-Noy, A., Cheilaris, P., & Smorodinsky, S. (2006a). Conflict-free coloring for intervals: From offline to online. *Proceeding of the 18th Annual ACM Symposium on Parallelism Algorithms Architectures (SPAA)*, 128-137. <http://dx.doi.org/10.1145/1148109.1148133>

Bar-Noy, A., Cheilaris, P., Olonetsky, S., & Smorodinsky, S. (2006b). Randomized online conflict-free coloring for hypergraphs. *Journal of Combinatorics, Probability and Computing, 19*(4), 493-516. <http://dx.doi.org/10.1017/S0963548309990587>

Chen, K. (2006). How to play a coloring game against a color-blind adversary. *Proceedings of the 22nd Annual Symposium on Computational Geometry (SCG '06)*, 44–51.

Chen, K., Kaplan, H., & Sharir, M. (2009). Online CF coloring for halfplanes, congruent disks, and axis-parallel rectangles. *ACM Transactions on Algorithms, 5*(2). <http://dx.doi.org/10.1145/1497290.1497292>

Even, G., Lotker, Z., Ron, D., & Smorodinsky, S. (2003). Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. *Siam Journal of Computing, 33*, 94-136. <http://dx.doi.org/10.1137/S0097539702431840>

Fiat, A., Levy, M., Matousek, J., Mossel, E., Pach, J., Sharir, M., Smorodinsky, S., Wagner, U., & Welzl, E. (2005). Online conflict-free coloring for intervals. *Proceeding of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, 545-554.

Har-Peled., S., & Smorodinsky, S. (2005). Conflict-free coloring of points and simple regions in the plane. *Discrete and Computational Geometry, 34*, 47–70. <http://dx.doi.org/10.1007/s00454-005-1162-6>

Irani, S. (1994). Coloring inductive graphs on-line. *Algorithmica, 11*(1), 53-72. <http://dx.doi.org/10.1007/BF01294263>

Pach, J., & Toth, G. (2003). Conflict free colorings. In *Discrete and computational geometry* (pp. 665-671), Springer Verlag. <http://dx.doi.org/10.1007/978-3-642-55566-4_30>

Schiermeyer, I., Tuza, Z., & Voigt, M. (2000). On-line rankings of graphs. *Discrete Mathematics, 212*(1–2), 141–147. <http://dx.doi.org/10.1016/S0012-365X(99)00215-0>

Smorodinsky, S. (2006). On the chromatic number of some geometric hypergraphs. *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA '06)*, 316–323.