A Decentralize Algorithm for Perturbation Minimization in 5G D2D Communication

Subhankar Ghosal and Sasthi C. Ghosh

Advanced Computing & Microelectronics Unit
Indian Statistical Institute,
203 B. T. Road, Kolkata 700108, India
Emails: subhoghosal7@gmail.com and sasthi@isical.ac.in

***Abstract*—In 5G device to device (D2D) communication, two users residing in close proximity can directly communicate among themselves without the need of a base station. Such a communicating pair of users forms a D2D link which operates on a common channel. Suppose Y dedicated channels are available to support the D2D communications. Since channels are limited resources, we may have to reuse the same channel to multiple links in order to enhance the capacity. But if two links are interfering to each other, they cannot use the same channel to avoid co-channel interference. The interference constraints at a particular time instant t can be modeled in terms of a graph $g(t)$ where each vertex represents a link and two vertices have an edge between them if their corresponding links are interfering to each other. At any time instant, a vertex must be assigned with a channel, otherwise the corresponding user pair can not communicate. Due to the movement of the users, a color/channel assigned at time $t-1$ may not be a valid coloring at time t. Hence we may have to recolor some vertices to satisfy the interference constraints. Our objective is to minimize the total number of such recoloring/perturbations $A(t)$ at time t. To minimize $A(t)$, we propose a decentralize differential coloring (DDC) algorithm and calculate the asymptotic bound on $A(t)$ produced by DDC. We also compare the value of $A(t)$ produced by DDC with other existing approaches. Results show that DDC produces less perturbations than the existing approaches. Theoretical results have also been validated through simulations.**

I. Introduction

In 5G device to device (D2D) communication, two proximity users residing within some *transmission range* of each other can communicate directly without involving the base station [11], [12], [13], [14], [10]. Each D2D communicating pair forms a link and communicate through a common channel. We assume that some Y channels are available for providing the D2D communications [19], [26], [27] which are different from those being used by the cellular networks. Since channels are limited resources, we have to reuse the same channel to multiple links. We assume a simple wireless network model that uses only omni-directional antenna. Hence if receiver of a link is residing within the *interference range* of the transmitter of another link, they cannot communicate through the same channel to avoid interference [31]. As discussed in several studies [19], [20], [21], [9], [15], the interference relationship among the active links at a particular time instant t can be modeled as an *interference graph* $g(t) = (V(g(t)), E(g(t)))$, where each vertex represents a link and two vertices have an edge between them if their corresponding links are interfering to each other. At a particular time instant, a link may or may not be active. Moreover, as interference range is typically small and users are moving, edges may appear or disappear over time. Since both $V(g(t)$ and $E(g(t))$ are time variant, $g(t)$ becomes a *dynamic graph* [1] which evolves over time.

To have a smooth communication, at each time t, each active link must be assigned with a channel such that the interference constraints are satisfied. In other words, at each time t, we have to find a channel/color vector $C(t) = (c_i(t))$ of $g(t)$, where $c_i(t) \in \{1, 2, \cdots, Y\}$ is the channel/color assigned to link/vertex i so that no monochromatic edge exists. Two interfering links form a monochromatic edge if they are assigned with the same channel. Since $g(t)$ is a dynamic graph, $C(t-1)$ of $g(t-1)$ may not be a valid coloring of $g(t)$. If we copy the colors of the common vertices of $g(t-1)$ and $g(t)$ from $C(t-1)$ to $C(t)$, many monochromatic edges may form in $g(t)$. The graph induced by those monochromatic edges is termed as *conflict graph* $g_c(t)$ [15]. To dissolve each monochromatic edge, we have to recolor at least one endpoint of it. Thus to dissolve all monochromatic edges, we have to recolor the vertices of a vertex cover of $g_c(t)$. Let $A(t)$ be the total number of recoloring/perturbation in $C(t)$ from $C(t-1)$. That is, $A(t) = \sum_i I_i(t)$ where $I_i(t) = 1$, if $c_i(t) \neq c_i(t-1)$, and 0, otherwise. In a practical point of view, recoloring a link essentially implies switching the channel of that link and hence it incurs a delay. To minimize such switching delays, we must minimize $A(t)$. Given Y, $C(t-1)$ and $g(t)$, our problem is to find $C(t)$ such that no monochromatic edge exists in $g(t)$ and $A(t)$ gets minimized. It is evident that the problem is equivalent to find the minimum vertex cover of $g_c(t)$ and hence NP-complete.

There could be two different approaches to solve this problem. First is the centralized approach, where using the position information of the D2D communicating pairs we could explicitly build $g_c(t)$ and then find its vertex cover to minimize $A(t)$. Second one is the decentralized approach. Here each D2D communicating pair will decide its coloring themselves. For such a decision to be made it has to exchange some control messages with its neighboring devices. We assume that these control message communications will be held using a separate channel and scheduling of such communications

ISBN 978-3-903176-13-3

would be done by the assistance of the base station. Note that for decentralized approach, we take help of the base station only for the control message interactions. This kind of help of base station had widely been taken in different distributed algorithms in the context of D2D communications [25], [26]. In fact, availability of base station is the fundamental difference of 5G D2D communication network from the Ad hoc network. Since the computation of channel allocation will be done by the D2D communicating pairs themselves, significant amount of overhead on the base station can be reduced.

A. Related Literatures

Coloring $g(t)$ with Y colors is equivalent to traditional graph coloring problem, which is known to be NP-complete [24], [23]. Graph coloring in both centralized and distributed set up is a well studied subject. Several centralized [2], [3] and decentralized [4], [5] graph coloring techniques are available in the literature.

Coloring $g(t)$ with Y colors with minimum perturbations is a less studied subject. It is evident that if $Y \geq |V(g(t))|$, $\forall t$ then the best solution is to apply a new color to each newly appeared vertex and retain that color throughout. According to [1] and also to our best of knowledge, only little works had been done to solve the problem of perturbation minimization. The problem had mostly been tackled before in terms of centralized heuristics and experimental results [6], [7], [8], [15]. In [1] authors proposed a big bucket and a small bucket algorithm, in which per newly appeared vertex or monochromatic edge they try to minimize the total number of recoloring/perturbations.

In [9], [15] authors deal with a slightly different problem. They assumed $Y(t)$ is not known and minimized a combined cost function $f(t) = Y(t) + \alpha A(t)$ of number of colors and perturbations where α is a constant representing relative weights of $Y(t)$ and $A(t)$. In [9], two approaches namely SNAP and SMASH are proposed to minimize $f(t)$. SNAP considers each graph separately whereas SMASH considers a union of k graphs. In [15], a differential coloring (DC) technique is proposed. DC first copies the colors of the common vertices of $g(t)$ and $g(t-1)$ from $C(t-1)$ to $C(t)$ and then constructs the conflict graph $g_c(t)$ induced by the monochromatic edges created by such copying of colors. Next DC finds a minimum vertex cover $v_c(t)$ of $g_c(t)$ using a 2-approximation vertex cover finding algorithm [29], [28] and recolors the vertices of $v_c(t)$. DC is a centralizes algorithm which needs channel state information (CSI) of each device to build $g_c(t)$. Note that several decentralized algorithms exist for finding a minimum vertex cover [16], [17], [18] of a given graph. But all these algorithms need a graph as input to find a minimum vertex cover of it. In other words, we must have to construct the conflict graph $g_c(t)$ first to apply those algorithms to find a minimum vertex cover of $g_c(t)$. It is important to note that building $g_c(t)$ in a decentralized manner itself is a challenging task. To build $g_c(t)$ in a decentralized way, each node must know the information of the entire graph $g(t)$, which involves huge amount of control message overhead. In this paper, we attempt to find a minimum vertex cover $v_c(t)$ of the conflict graph *without explicitly building the conflict graph*. Next, we recolor the vertices of $v_c(t)$ in a decentralize fashion so that $A(t)$ get minimized.

B. Our contributions

Our contributions are summarized as follows:

- A decentralized differential coloring (DDC) algorithm is proposed for perturbation minimization problem. DDC finds a minimal vertex cover $v_c(t)$ of the conflict graph $g_c(t)$ and recolors all the vertices in $v_c(t)$ to keep the perturbations at minimum level. The salient feature of DDC is that it does not explicitly generates $g_c(t)$ to find $v_c(t)$ and hence DDC reduces the computation time as well as control message overhead.
- Considering $g(t)$ as random graph, we have computed the expected number of perturbations $\mathbb{E}[A(t)]$ produced by DDC theoretically.
- We compared DDC theoretically as well as through simulations with existing centralized big and small bucket algorithms. We have shown that DDC performs better than those algorithms in terms of expected number of perturbations.
- The theoretical results have also been validated through simulations.

Rest of the paper is organized as follows. In section II, we present DDC formally. In section III, we present asymptotic bounds on $\mathbb{E}[A(t)]$ produced by DDC. We compare DDC with small and big bucket algorithms in section IV. We have validated our bounds through rigorous simulations in section V. The paper is concluded in section VI.

II. Decentralized differential coloring (DDC)

In this section we present our decentralized differential coloring (DDC) algorithm to minimize $A(t)$. Initially all links which were active in $g(t-1)$ and are still active in $g(t)$ will retain their previous colors. Links which become newly active in $g(t)$ will be assigned with random colors. This may cause some monochromatic edges in $g(t)$. We assume that the user-pair forming a D2D link can identify whether its channel is being used by other user-pair in its proximity by observing the channel state information (CSI). When user-pair of a link identify that some other link is interfering with it, then the concerned user-pair goes in channel switching mode. In this mode, the concerned user-pair will be in communication with each other and decide together a different channel for communication so that the monochromatic edge is dissolved. When user-pair forming link i finds that its channel is being used by another link, it broadcasts its channel/color information to all its neighbors. Here a vertex is a neighbor of another vertex if it falls withing the interference range of that vertex. Communication power of a vertex will be set to transmit upto its interference range while the concerned vertex is in channel switching mode. Communications in this mode will be performed by taking the assistance of the base station. Each receiving user of that signal then reply back their channel

ISBN 978-3-903176-13-3

information to the requesting user. By this way the user-pair forming link i will get the total channel information of its neighbors and construct a set $M_i(t)$ containing all links which are active with the same channel as that of link i. If $M_i(t)$ is not empty, link i randomly chooses link $j \in M_i(t)$ and send a pairing request to j. If i and j both send pairing request to each other then they form a pair and send that information to all their neighbors. The said pair (i, j) will then choose two random colors from $\{1, 2, \cdots, Y\}$ such that for each link the color of that link is not being used by their neighbors. After forming the pair and successfully acquiring the channel, both i and j will get released from channel switching mode. If link i is not paired, upon receiving pairing requests from other neighbors, it removes those neighbors from $M_i(t)$. If $M_i(t)$ is still not empty, i repeat the process to pair up with another link. Finally when $M_i(t)$ become empty, then no monochromatic edge exists whose one endpoint is the color used by link i. In that case link i will get released from channel switching mode. This algorithm is presented formally in Algorithm 1 which will be run by the user-pair forming link i.

Algorithm 1: *Decentralized differential coloring (DDC)*

Input: Set of colors $\{1, 2, \cdots, Y\}$, $c_i(t-1)$
Output: $c_i(t)$

For each vertex common in $V(g(t))$ and $V(g(t-1))$ do $c_i(t) = c_(t-1)$;
For each newly appeared vertex set $c_i(t)$ to a random color from $\{1, 2, \cdots, Y\}$;
Check whether $c_i(t)$ is used by any other link by seeing CSI;
if $c_i(t)$ *is used by another link in its neighborhood* **then**
    Enter into channel switching mode;
    Send current color $c_i(t)$ to all neighbors and request for their colors;
    Receive colors from all neighbors;
    Construct $M_i(t)$ as the set of all neighbors whose color is same as $c_i(t)$;
    while $M_i(t) \neq \emptyset$ **do**
        Choose an user j randomly from $M_i(t)$;
        Send pairing request to j;
        Receive pairing request from all neighbors;
        if j *sent a pairing request to* i **then**
            Pair with j;
            Share the pairing status with all neighbors;
            Choose $c_i(t)$ and $c_j(t)$ randomly from $\{1, 2, \cdots, Y\}$ such that links i and j get colors which are not being used by their respective neighbors;
            Exit from channel switching mode;
        else
            Receive pairing information from all neighbors;
            Remove all already paired links from $M_i(t)$;
            Exit channel switching mode;
else if i *receives coloring request from any of its neighbors* **then**
    Send current color $c_i(t)$ to all such requesting neighbors;

Note that Algorithm 1 essentially recolors all endpoints of a *random maximal matching* of the conflict graph $g_c(t)$ without explicitly constructing the conflict graph $g_c(t)$. Since any minimum vertex cover must include at least one endpoint of each edge of a maximal matching, we could bound the number of perturbations as stated in Theorem 1.

Theorem 1. *If optimum perturbation in $g(t)$ from $C(t-1)$ is $A_{opt}(t)$ then DDC essentially creates number of perturbations $A(t) \leq 2A_{opt}(t)$.*

Proof. It is evident that we must have to recolor each element of a minimum vartex cover of $g_c(t)$ to dissolve all monochromatic edges. In other words, $A_{opt}(t)$ is the size of a minimum vertex cover of $g_c(t)$. Note that Algorithm 1 recolors both endpoints of each edge of a random maximal matching of $g_c(t)$. Each vertex cover includes at least one endpoint of each edge of a maximal matching. Also the set constructed by both endpoints of each edge of a maximal matching is a vertex cover itself. Hence Algorithm 1 essentially a 2-approximation algorithm to find a minimum vertex cover of $g_c(t)$. Hence $A(t) \leq 2A_{opt}(t)$. □

Remark 1. *It is evident that if we build $g_c(t)$ explicitly and then find the conflict graph, each link requires information of the whole graph. That is, in that case, control message overhead for each link would be $O(|E(g(t))|)$. In contrast to that in DDC algorithm each vertex needs to know the information of its neighbors only. Hence in DDC the control message overhead for each link i will be $O(deg(i, g(t)))$, where $deg(i, g(t))$ is the degree of vertex i in $g(t)$. Thus the total control message overhead for $g(t)$ corresponding to the above mentioned two cases will be $O(|V(g(t))| \times |E(g(t))|)$ and $O(\sum_i deg(i, g(t))) = O(2 \times |E(g(t))|)$ respectively. Hence DDC improved the control message overhead by a factor of $\frac{|V(g(t))|}{2}$.*

Complexity 1. *While in switching mode, the user-pair forming link i has to send and receive color information from all its neighbors in $O(1)$ time. Let $\Delta(g_c(t))$ be the maximum degree of $g_c(t)$. To form a pair, the user-pair forming link i has to send pairing request to all its neighbors in $O(1)$ time. There could be at most $O(\Delta(g_c(t)))$ neighbors. Hence probability that i will be able to form a pair is*

$$\sum_{j=1}^{\Delta(g_c(t))} O((\frac{1}{\Delta(g_c(t))})^2) = O(\frac{1}{\Delta(g_c(t))}).$$

Thus expected time required to form a pair is $O(\Delta(g_c(t)))$. Hence the total expected time complexity of Algorithm 1 is $O(\Delta(g_c(t)))$. Also each user-pair forming a link has to store the information received from all its neighbor. Thus total space complexity is $O(\Delta(g_c(t)))$ per link.

III. Expected number of perturbations

In this section, we calculate bounds on the expected number of perturbations generated by DDC assuming $g(t)$ as a random graph. Let us denote $G_{N,p}$ as a Erdos-Reyni (ER) random graph where N is the number of vertices and each edge is generated independently with probability p. It is well known that the average degree of $G_{N,p}$ is Np. We assume that some n links are present in a large coverage area. We also

assume that at a particular point of time a link will be active with probability p_v. As links are independent to each other, $\mathbb{E}[|V(g(t))|] = np_v$ and $g(t)$ will be G_{np_p,p_e}, where p_e is the edge generation probability in $g(t)$. It is evident that as $n \to \infty$, the user density over the entire region tends to some constant value. In other words, when $n \to \infty$, the average degree of $g(t)$ tends to some constant value. That is, $np_vp_e = O(1)$ when $n \to \infty$. Using this fact we derive expected number of perturbations in Theorem 2. Before going to our main result as stated in Theorem 2, we first state a short result in Lemma 1 using which we will prove the theorem. We now introduce a notation which will be used in the proofs of both Lemma 1 and Theorem 2. We denote $A(n)$ is asymptotically equals to $B(n)$ as $A(n) \sim_n B(n)$. Here $A(n) \sim_n B(n)$ means

$$\lim_{n\to\infty} \frac{A(n)}{B(n)} = 1.$$

Lemma 1. *Size of minimum vertex cover of Erdos-Reyni random graph $G_{N,p}$ is $O(N^2p)$, when $Np = O(1)$, $Np < 1$ and $N \to \infty$.*

Proof. It is well-known that the size of the minimum vertex cover of $G_{N,p}$ is $\sim_N N(1 - \frac{1}{\chi(N,p)})$ [30], where $\chi(N,p)$ is the expected chromatic number of $G_{N,p}$. We also know from [30] that when $N \to \infty$, $Np = O(1)$ and $Np \geq 1$ then $\chi(N,p) = O(Np) = O(1)$. Hence the size of minimum vertex cover of $G_{N,p}$ is $O(N)$ when $Np \geq 1$.

We now calculate the size of minimum vertex cover of $G_{N,p}$ when $Np < 1$. Let X_i be an indicator variable where $X_i = 1$ and $X_i = 0$ represent that vertex i is non-isolated and isolated in $G_{N,p}$ respectively. Note that $P(X_i = 1) = Np$ when $Np < 1$. Hence the total number of non-isolated vertices in $G_{N,p}$ is $\mathbb{E}[\sum_i X_i] = \sum_i \mathbb{E}[X_i] = N \times P(X_i = 1) = N \times Np$. It is evident that the size of minimum vertex cover of $G_{N,p}$ essentially is the size of minimum vertex cover of the graph induced by its non-isolated vertices only. This implies if $Np < 1$, the size of minimum vertex cover of $G_{N,p}$ will be equivalent to the size of minimum vertex cover of a $G_{N^2p,p'}$ where each of N^2p vertices is a non-isolated vertex and $N^2pp' = O(1)$. Since each vertex is a non-isolated vertex in such a $G_{N^2p,p'}$, we get $N^2pp' \geq 1$. Hence using the above mentioned well-known result, we get the size of minimum vertex cover of $G_{N,p}$ as $O(N^2p)$ when $Np < 1$. Hence the proof. □

Theorem 2. *Expected number of perturbations in $g(t)$ is $\mathbb{E}[A(t)] = O(\frac{n^2p_v^4p_e}{Y})$.*

Proof. Since we are using Y colors to color $g(t)$, $\mathbb{E}[Y(t)] \leq Y$. We apply 2-approximation algorithm to find the minimum vertex cover of the conflict graph and size of it represents the total number of perturbations. Again in conflict graph each vertex is independent and identical. Note that to be a member of conflict graph $g_c(t)$ each vertex must be a member of both $g(t)$ and $g(t-1)$. Hence conflict graph is also an ER graph $G_{np_v^2,p'_e}$, where p'_e is the edge generation probability in the conflict graph.

We now calculate p'_e. It is evident that an edge will be monochromatic in $g(t)$ only if it is present in $g(t)$, absent in $g(t-1)$ and both endpoints of it were colored with the same color in $g(t-1)$. Hence

$$\begin{aligned} p'_e = \; & P(ij \in E(g(t)) \;\&\; ij \notin E(g(t-1)) \\ & \times P(c_i(t-1) = c_j(t-1)). \end{aligned} \tag{1}$$

It is evident that $P(ij \in E(g(t)) \;\&\; ij \notin E(g(t-1)) = p_e(1-p_e)$. It is also evident that $P(c_i(t-1) = c_j(t-1)) = \sum_{c=1}^{Y} (P(c_i(t-1) = c))^2$. So Equation (1) can be rewritten as

$$p'_e = \; p_e(1-p_e) \times \sum_{c=1}^{Y} (P(c_i(t-1) = c))^2. \tag{2}$$

As we don't put any bias on the colors in DDC algorithm and colors are chosen randomly and independently from $\{1, 2, \cdots, Y\}$, we get $P(c_i(t-1) = c) = \frac{1}{Y}$ for all $c \in \{1, 2, \cdots, Y\}$. So Equation (2) can be rewritten as

$$\begin{aligned} p'_e = \; & p_e(1-p_e) \times \sum_{c=1}^{Y} \frac{1}{Y^2} \\ = \; & \frac{p_e(1-p_e)}{Y}. \end{aligned} \tag{3}$$

Hence $g_c(t)$ is $G_{np_v^2, \frac{p_e(1-p_e)}{Y}}$ whose average degree is $\frac{np_v^2p_e(1-p_e)}{Y}$. It is evident that when $n \to \infty$ and $np_e = O(1)$ then $\frac{np_v^2p_e}{Y}(1-p_e) \sim_n \frac{np_v^2p_e}{Y}$. Now $\frac{np_v^2p_e}{Y} = O(1)$ as $np_e = O(1)$, and p_v, Y are constants. Hence average degree of $G_{np_v^2, \frac{p_e(1-p_e)}{Y}}$ is $\sim_n \frac{np_v^2p_e}{Y} = O(1)$ when $n \to \infty$.

We get from [30] that np_vp_e is the expected chromatic number of $g(t)$ and hence $np_vp_e \leq Y$. Thus $\frac{np_vp_e}{Y} \times p_v < 1$.

Since the average degree of $G_{np_v^2, \frac{p_e(1-p_e)}{Y}}$ is $O(1)$ at $n \to \infty$ and $\frac{np_v^2p_e}{Y} < 1$, we get from Lemma 1 that the size of minimum vertex cover of $g_c(t)$ is $O(\frac{n^2p_v^4p_e}{Y})$. Hence the proof. □

IV. Comparison with other approaches

In [1] authors have proposed two centralized algorithms, namely small bucket and big bucket algorithms, for maintaining coloring of a dynamic graph. In their algorithms d levels are considered. Each level $i \in \{0, 1, \cdots, d-1\}$ consists of s buckets plus an reset bucket. A bucket at level i consists of s^i vertices. Each bucket uses Y_{max} colors and the colors allocated for each bucket are distinct from the colors of other buckets. They assumed a coloring algorithm which can color the graph with Y_{max} colors. If an monochromatic edge appears one of its endpoint is removed from the corresponding bucket and considered as a newly appeared vertex. Once a new vertex appears their algorithm puts that vertex in an empty bucket at level $i = 0$. If $s-1$ buckets are already filled in a level it accumulates all vertices at that level and puts them in an empty bucket of the immediate upper level and recolor the subgraph induced by them with the assumed coloring algorithm. They

ISBN 978-3-903176-13-3

showed that each bucket could be colored in Y_{max} colors and hence using $O(dn^{\frac{1}{d}}Y_{max})$ colors they can color the graph with $O(d)$ recoloring for each update. Here an update means appearance of a monochromatic edge or insertion of a new vertex. Varying the bucket size they conclude $O(dn^{\frac{1}{d}}Y_{max})$ color requirements with $O(d)$ recoloring per update in small bucket algorithm and $O(dY_{max})$ color requirements with $O(dn^{\frac{1}{d}})$ recoloring per update in big bucket algorithm.

Note that in our problem scenario at each time instant t we have to recolor the vertices of a minimum vertex cover of the conflict graph. It is evident that DDC makes 2 recoloring per update as DDC is a 2-approximation algorithm for finding a minimum vertex cover of the conflict graph. In contrast to this, small bucket algorithm requires $O(d)$ recoloring and big bucket algorithm requires $O(dn^{\frac{1}{d}})$ recoloring per update. Hence if we set the same Y as the color requirements of all these algorithms, then DDC produces less perturbations than big bucket algorithm and if $d > 2$, DDC also produces less perturbations than small bucket algorithm. This improvement is quite expected because of the fact that DDC recolors only the vertices of a minimum vertex cover of the conflict graph instead of recoloring both endpoints of each monochromatic edges.

V. Simulation

In this section, we would actually generate random sequences of $g(t)$s and observe the behavior of DDC on those random graphs. At each time instance, there are n links of which each link gets active with probability p_v. Considering $g(t)$ as G_{np_v,p_e}, we set the average degree of the $g(t)$ as $np_vp_e = 10p_v$. We observe the number of perturbation $A(t)$ obtained by DDC for different values of n, p_v, p_e and Y. For a fixed value of n, p_v, p_e and Y we run DDC on 10000 many randomly generated graphs and report the average $A(t)$ produced by DDC. Note that the expression of $\mathbb{E}[A(t)]$ in Theorem 2 is asymptotic in nature. For finite n, we now observe the behavior of $\mathbb{E}[A(t)]$ through simulations and compare the obtained results with the asymptotic results.

In Figure 1 we fix $Y = 800$, $p_v = 0.5$ and increase n from 500 to 1000 with a step of 25. We observe that $\mathbb{E}[A(t)]$ is almost linearly increasing with n. Note that from Theorem 2 we get that the expression of $\mathbb{E}[A(t)]$ as $O(\frac{n^2p_v^4p_e}{Y}) = O(n \times \frac{np_v^4p_e}{Y}) = O(n)$, where $np_e = O(1)$ and p_v and Y are all constants. Hence the expression says that if other parameters remains constant $\mathbb{E}[A(t)] = O(n)$. Hence this behavior tallies with theoretical findings of Theorem 2.

In Figure 2 we fix $n = 750$, $Y = 500$ and increase p_v from 0 to 1 with a step of 0.05. We observe that $\mathbb{E}[A(t)]$ is increasing with p_v like p_v^4. This observed behavior again tallies with our theoretical expression of Theorem 2, which is $\mathbb{E}[A(t)] = O(\frac{n^2p_v^4p_e}{Y}) = O(p_v^4)$, as $np_e = O(1)$ and n and Y are all constants.

In Figure 3 we fix $n = 750$, $p_v = 0.5$ and increase Y from 500 to 750 with a step of 5. We observe that $\mathbb{E}[A(t)]$ is decreasing hyperbolically with Y. This behavior

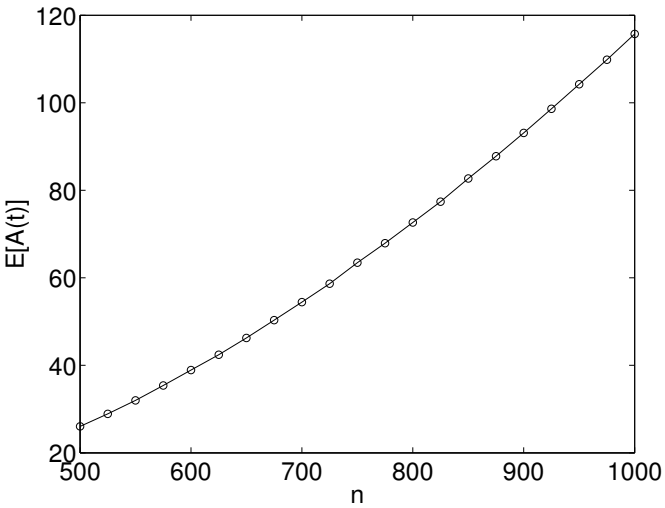

Fig. 1: Behavior of $\mathbb{E}[A(t)]$ for DDC with increasing n where $Y = 800$ and $p_v = 0.5$

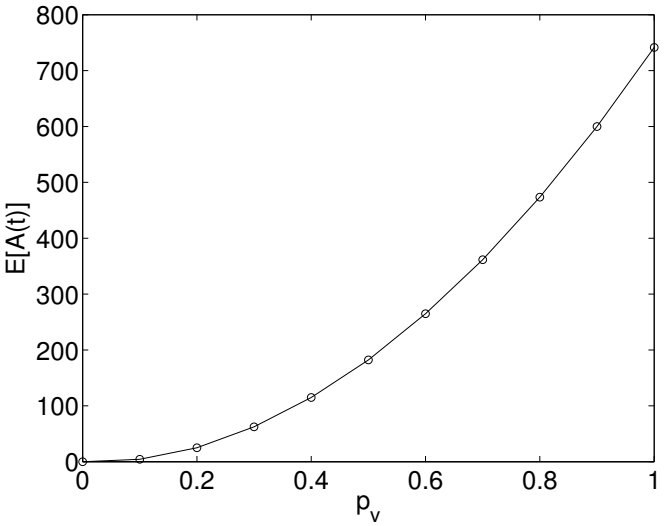

Fig. 2: Behavior of $\mathbb{E}[A(t)]$ for DDC with increasing p_v where $n = 750$ and $Y = 500$

also tallies with the theoretical expression of Theorem 2, which is $\mathbb{E}[A(t)] = O(\frac{n^2p_v^4p_e}{Y}) = O(\frac{1}{Y})$, as $np_e = O(1)$ and n and p_v are all constants.

In Figure 4 we fix $n = 750$, $p_v = 0.5$ and $Y = 500$ and vary p_e from 0 to 1 with a step of $\frac{1}{30}$. We observe that $\mathbb{E}[A(t)]$ is first increasing and after reaching a point it started decreasing down to 0. When p_e is small $g(t)$ has smaller number of edges, hence probability that a monochromatic edge will form is very small because the probability of edge formation itself is small. With the increase of p_e probability of monochromatic edge formation gradually increases and hence $\mathbb{E}[A(t)]$ increases. But with increasing number of edges, the chromatic number of the graph itself will become very high. It is evident that with low chromatic numbered graph random assignment get more scope to put same color to multiple vertices. But as the

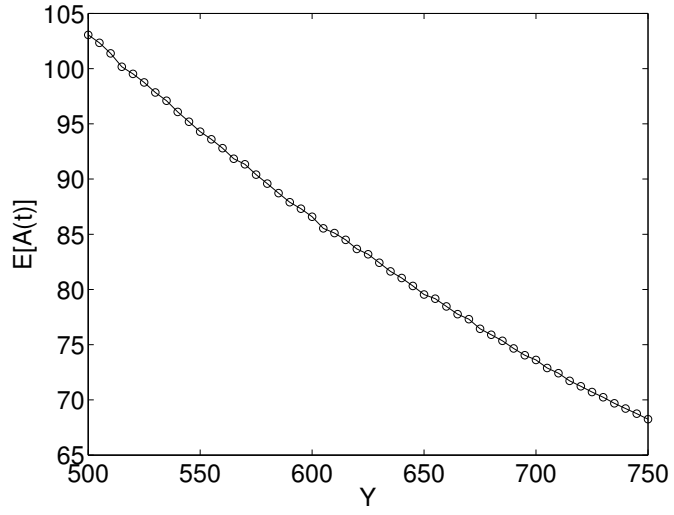

Fig. 3: Behavior of $\mathbb{E}[A(t)]$ for DDC with increasing Y where $n = 750$ and $p_v = 0.5$

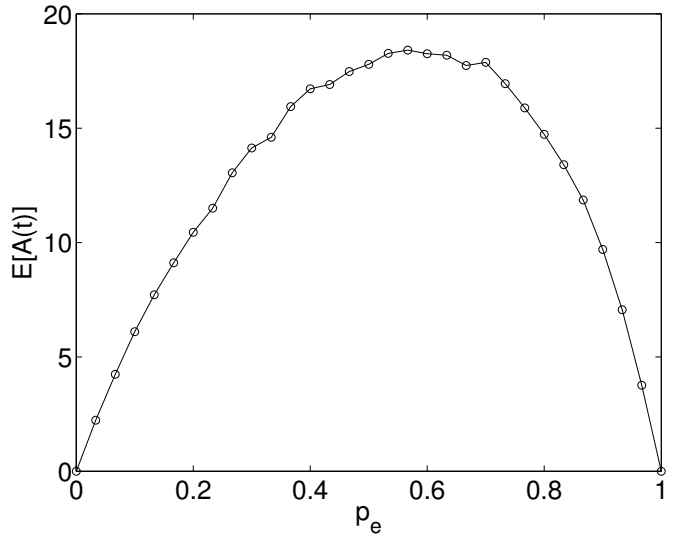

Fig. 4: Behavior of $\mathbb{E}[A(t)]$ for DDC with increasing p_e where $Y = 500$, $n = 750$ and $p_v = 0.5$

p_e cross some threshold $g(t)$'s chromatic number become so high that each vertex get eventually different color, and hence with slight variation of the graph the new graph does not need any new recoloring. Thats why after a point $\mathbb{E}[A(t)]$ started to decline till it reaches to 0 when the graph become a complete graph by itself.

We now compare DDC with an existing bucket based algorithm [1]. We generate ER graphs with different n, p_v and p_e. By varying s, d for a given n and Y, we generate $\mathbb{E}[A(t)]$ for bucket based algorithm. Also, for the same n and Y, we calculate $\mathbb{E}[A(t)]$ as produced by DDC algorithm. The results are reported in Table I. We observe that for a given value of n, p_v, p_e and Y DDC produces much smaller $\mathbb{E}[A(t)]$ than the bucket based algorithm. This result validates the theoretical finding. Here A_b and A_d in Table I represent $\mathbb{E}_b[A(t)]$ produced by the bucket based algorithm and DDC respectively.

n	p_v	p_e	s	d	Y	A_b	A_d
100	0.5	$10/n$	2	5	65	20.59	0.80
200	0.5	$10/n$	2	6	75	41.68	1.97
300	0.5	$10/n$	2	7	85	61.31	2.39
400	0.5	$10/n$	2	7	85	81.93	2.61
500	0.5	$10/n$	2	7	85	105.72	3.89
100	0.5	$10/n$	3	3	65	18.37	0.80
200	0.5	$10/n$	3	3	65	38.91	1.50
300	0.5	$10/n$	3	4	80	59.47	2.43
400	0.5	$10/n$	3	5	110	85.16	2.15
500	0.5	$10/n$	3	5	110	105.4	2.58

TABLE I: Bucket based algorithm vs DDC

VI. Conclusion

In this paper, we have proposed a distributed differential coloring algorithm to minimize perturbations for a given number of colors. We have calculated the expected perturbations produced by DDC and compared with bucket based algorithm. Finally theoretical findings are also verified through simulations.

References

[1] L. Barba, J. Cardinal, M. Korman, S. Langerman, A. V. Renssen, M. Roeloffzen, S. Verdonschot, "Dynamic Graph Coloring" *In Workshop on Algorithms and Data Structures (Springer, Cham)*, pp. 97-108, 2017

[2] P. Galinier, and A. Hertz. "A survey of local search methods for graph coloring." *Computers & Operations Research*, Vol 33, No. 9, pp.2547-2562, 2006.

[3] P.M. Pardalos, T. Mavridou, and J. Xue. "The graph coloring problem: A bibliographic survey." *Handbook of combinatorial optimization. Springer*, Boston, MA, pp. 1077-1141, 1998.

[4] G. De Marco, and A. Pelc. "Fast distributed graph coloring with O(Δ) colors." *In proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pp. 630-635, 2001.

[5] L. Barenboim, M. Elkin (2013). "Distributed graph coloring: Fundamentals and recent developments." *Synthesis Lectures on Distributed Computing Theory*, Vol. 4, No. 1, pp. 1-171, 2013.

[6] P. Borowiecki, E. Sidorowicz, "Dynamic coloring of graphs" *In procceding of Fundamenta Informaticae* Vol. 114(2), pp. 105-128, 2012.

[7] L. Ouerfelli, H. Bouziri, "Greedy algorithms for dynamic graph coloring" *In procceding of International Conference on Communications, Computing and Control Applications (CCCA)*, pp. 15, 2011.

[8] D. Preuveneers, Y. Berbers. "ACODYGRA: an agent algorithm for coloring dynamic graphs" *In procceding of nternational Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 381390, 2004.

[9] F. Yu, A. Bar-Noy, P. Basu and R. Ramanathan, "Algorithms for channel assignment in mobile wireless networks using temporal coloring", *Proceedings of the 16th ACM international conference on modeling, analysis & simulation of wireless and mobile systems (ACM)*, 2013.

[10] Z. Lin, Y. Li, S. Wen, Y. Gao, X. Zhang, D. Yang "Stochastic geometry analysis of achievable transmission capacity for relay-assisted device-to-device networks.", *Proceedings of IEEE ICC*, pp. 2251-2256, 2014.

[11] A. Pratap, and R Misra. "Firefly inspired improved distributed proximity algorithm for D2D communication.", *IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW)*, 2015.

[12] H. ElSawy, E. Hossain, and M Alouini "Analytical modeling of mode selection and power control for underlay D2D communication in cellular networks.", IEEE Transactions on Communications Vol 62, No 11, pp - 4147-4161, 2014.
[13] A. H. Sakr, E. Hossain, "Cognitive and energy harvesting-based D2D communication in cellular networks: Stochastic geometry modeling and analysis." *IEEE Transactions on Communications*, Vol 63(5),pp - 1867-1880, 2015.
[14] M. N. Tehrani, M Uysal, and H Yanikomeroglu "Device-to-device communication in 5G cellular networks: challenges, solutions, and future directions." *IEEE Communications Magazine* Vol 52, No 5, pp - 86-92, 2014.
[15] S. Ghosal, S. C. Ghosh, "Channel assignment in mobile networks based on geometric prediction and random coloring", *Proceedings of the 40th IEEE Conference on Local Computer Networks (LCN)*, pp. 237-240, 2015.
[16] M. Astrand, J. Suomela (2010, June), "Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks." *In Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures* pp. 294-302, 2010.
[17] C. Koufogiannakis, N. E. Young, "Distributed and parallel algorithms for weighted vertex cover and other covering problems" *In Proceedings of the 28th ACM symposium on Principles of distributed computing* pp. 171-179, 2009.
[18] F. Grandoni, J. Knemann, A. Panconesi, "Distributed weighted vertex cover via maximal matchings." *ACM Transactions on Algorithms (TALG)*, Vol. 5, No. 1, pp. 6, 2008.
[19] L. Zhao, H. Wang, X. Zhong, (). "Interference Graph Based Channel Assignment Algorithm for D2D Cellular Networks." *IEEE Access*,Vol 6,pp. 3270-3279,2018.
[20] X. Cai, J. Zheng, and Y. Zhang, "A Graph-coloring based resource allocation algorithm for D2D communication in cellular networks," *In Proceedings of IEEE ICC*, pp. 54295434, 2015.
[21] D. Tsolkas, E. Liotou, N. Passas, and L. Merakos,"A graph-coloring secondary resource allocation for D2D communications in LTE networks," *In Proceeding of IEEE CAMAD*, pp. 5660, 2012.
[22] X. Ma and E. L. Lloyd, "An incremental algorithm for broadcast scheduling in packet radio networks", *Proceedings of the IEEE Military Communications Conference (MILCOM 98)*, Vol. 1, 1998.
[23] W. Hale, "Frequency assignment: theory and application", *Proceedings of the IEEE*, Vol. 68, No. 12, pp. 1497-1514, 1980.
[24] R.M. Karp, "Reducibility among combinatorial problems". *In Complexity of computer computations* Springer US pp. 85103, 1972.
[25] R. Yin, G. Yu, H. Zhang, Z. Zhang, G. Y. Li, "Pricing-based interference coordination for D2D communications in cellular networks." *IEEE Transactions on Wireless Communications*, Vol 14, No. 3, pp. 1519-1532, 2015.
[26] R. Yin, C., Zhong, G. Yu, Z. Zhang, K. K. Wong, X. Chen, "Joint spectrum and power allocation for D2D communications underlaying cellular networks." *IEEE Transactions on Vehicular Technology*, Vol .65, No. 4, pp. 2182-2195, 2016.
[27] Y. Jiang, Q. Liu, F. Zheng, X. Gao, X. You, "Energy-efficient joint resource allocation and power control for D2D communications." *IEEE Transactions on Vehicular Technology*, Vol. 65, No. 8, pp. 6119-6127, 2016.
[28] T.H. Cormen, E.C. Leiserson, L.R. Rivest, C. Stein, "Section 35.1: The vertex-cover problem". *Introduction to Algorithms (2nd ed.)* MIT Press and McGraw-Hill, pp. 10241027, ISBN 0-262-03293-7, 2001.
[29] C. H. Papadimitrioua and K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity", *Dover Publications*, pp. 432.
[30] G. R. Grimmett and C. J. H. McDiarmid, "On coloring random graphs," *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 77, No. 2, pp. 313-324, 1975.
[31] K. Venugopal, M. C. Valenti, R. W. Heath, "Device-to-device millimeter wave communications: Interference, coverage, rate, and finite topologies." *IEEE Transactions on Wireless Communications*, Vol. 15, No. 9, pp. 6175-6188, 2016.

ISBN 978-3-903176-13-3