Randomized algorithms for resource allocation in device to device communication

Thesis submitted in fulfillment of the requirements for the degree

of

Doctor of Philosophy in Computer Science

by

Subhankar Ghosal

Under the supervision of

Prof. Sasthi Charan Ghosh

Submitted on December 2021

ADVANCED COMPUTING AND MICROELECTRONICS UNIT

INDIAN STATISTICAL INSTITUTE

DECLARATION

Thesis Title	Randomized algorithms for resource allocation in device to device communication
Author	*Subhankar Ghosal*
Supervisor	Prof. Sasthi Charan Ghosh

I declare that this thesis entitled *Randomized algorithms for resource allocation in device to device communication* is the result of my own work except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Subhankar Ghosal

Subhankar Ghosal

Advanced Computing and Microelectronics Unit
Indian Statistical Institute

Date: January 13, 2022

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my research supervisor, *Prof. Sasthi Charan Ghosh*, for kindly giving me the opportunity to do research under him and providing invaluable guidance throughout this work. His dynamism, vision, sincerity and motivation have deeply inspired me. He has taught me the methodology to carry out the work and to present the works as clearly as possible. It was a great privilege and honor to work under his guidance.

I am greatly indebted to my honorable teachers of Advanced Computing and Microelectronics Unit of Indian Statistical Institute who taught me different techniques and enlighten me about recent developments in Computer Science.

I am extremely thankful to my parents for their unconditional love, endless prayers, caring and immense sacrifices for educating and preparing me for my future. I would like to say thanks to my friends and relatives for their kind support and care.

Finally, I would like to thank all the people who have supported me to complete these research work directly or indirectly.

Subhankar Ghosal

Subhankar Ghosal

Advanced Computing and Microelectronics Unit
Indian Statistical Institute

Date: January 13, 2022

ABSTRACT

In device to device (D2D) communication, two users residing in close proximity can directly communicate between themselves, through a common channel, without the need of a base station. A pair of D2D users forms a link and a channel needs to be allocated to it. The interference relationship among the active links at time t is modelled in terms of a interference graph. To establish interference-free communication, we have to assign a channel and some power to each link such that the required signal to interference plus noise ratio (SINR) criteria is satisfied for each link. Since channels are costly resources we have to minimize the maximum channel used at time t. Due to the movement of the D2D users, a channel allocated at time $(t-1)$ may create interference at time t. Hence a link may require to switch its channel to maintain its SINR. Channel switching produces delay and hence an additional overhead. Hence, to maintain quality of service, the total number of channel switches or perturbations at time t should also be minimized. Since each transmitter has a limited battery power, minimizing the total power allocated at time t is also an objective of minimization. Note that if we allocate each link a different channel then each link has no interference from other links. In that case, total perturbations is zero and each link can operate with the minimum power, resulting total power requirement to be the minimum. But the maximum channel used is huge in that case.Thus the maximum channel used, the total perturbations and the total power requirement have natural trade-offs among them. Due to the hardness, optimizing maximum channel used, total perturbations and total power requirement, owing to their respective trade-offs is a challenging task. In this thesis, we have developed a randomized algorithm as well as its parallel version which can minimize the maximum channel (color) used, in expected polynomial time. To minimize the total perturbations, we developed a centralized and a decentralized differential coloring technique as well as a random coloring technique. We calculated expected perturbations produced by each of them. To minimize a cost defined as a linear combination of the maximum channel used and the total perturbations, we proposed a geometric prediction based and a graph union based approach and calculated the expected cost produced by them. Finally to minimize a cost defined as a linear combination of the maximum channel used and the total power requirement, we proposed a randomized joint channel and power allocation technique and calculated the expected cost and energy efficiency produced by it. For each problem, we theoretically analyse the performance of our algorithms, compare with existing state-of-the-art solutions and verify the analysis through simulation.

LIST OF PUBLICATIONS FROM THE CONTENT OF THE THESIS

Accepted Journals

- Subhankar Ghosal and Sasthi C. Ghosh: "A Randomized Algorithm for Joint Power and Channel Allocation in 5G D2D Communication", Computer Communications (Elsevier) [Accepted for publication on Jul 21, 2021].

Published in Refereed Conference Proceedings:

1. Subhankar Ghosal and Sasthi C. Ghosh: "An Incremental Search Heuristic for Coloring Vertices of a Graph". Graphs and Combinatorial Optimization from Theory to Applications, Proc. of the 18th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2021), pp. 39-52, 2021.

2. Subhankar Ghosal and Sasthi C. Ghosh: "A Randomized Algorithm for Joint Power and Channel Allocation in 5G D2D Communication". Proc. of the 18th IEEE International Symposium on Network Computing and Applications (IEEE NCA 2019), Cambridge, MA, USA, September 26-28, pp. 1-5, 2019.

3. Subhankar Ghosal and Sasthi C. Ghosh: "A Decentralize Algorithm for Perturbation Minimization in 5G D2D Communication". Proc. of the 15th Wireless On-demand Network systems and Services (IEEE/IFTP WONS 2019), Wengen, Switzerland, January 22-24, pp. 72-78, 2019.

4. Subhankar Ghosal and Sasthi C. Ghosh: "Channel Assignment in Mobile Networks Based on Geometric Prediction and Random Coloring". Proc. of

the 40th IEEE Conference on Local Computer Networks (IEEE LCN 2015), Florida, USA, October 26-29, pp. 237-240, 2015.

Submitted/Under Review at Journals:

1. Subhankar Ghosal and Sasthi C. Ghosh: "Expected Polynomial-time Randomized Algorithm for Graph Coloring Problem", Under review at Discrete Applied Mathematics (Elsevier) [Manuscript ID: DA12997].

2. Subhankar Ghosal and Sasthi C. Ghosh: "An Expected Polynomial-time Algorithm for Perturbation Sensitive Sparse Temporal Graph Coloring". Submitted to Discrete Applied Mathematics (Elsevier) [Manuscript ID: DA13503].

NOTATIONS

Notation	Meaning
$C(t) = (c_i(t))$	Color/channel vector at time t, where $c_i(t)$ is the color/channel allocated to vertex/link i at that time.
$X(t) = (x_i(t))$	Power vector at time t, where $x_i(t)$ is the power allocated to vertex/link i at that time.
n	Total number of vertices/links.
$I_i(C(t), C(t-1))$	1 if $c_i(t) \neq c_i(t-1)$ otherwise 0
$Y(t) = \max_{i=1}^{n} c_i(t)$	Maximum color/channel allocated at time t
$A(t) = \sum_{i=1}^{n} I_i(C(t), C(t-1))$	Total number of perturbations at time t
$P(t) = \sum_{i=1}^{n} x_i(t)$	Total power of all links at time t
α	Relative weight between $Y(t)$ and $A(t)$ or $Y(t)$ and $P(t)$
$g(t)$	Interference graph at time t
$\Delta(g(t))$	Maximum degree of $g(t)$
$\delta(i, t)$	Degree of vertex i at time t
$\overline{Y}$	Average Y

$\overline{P}$	Average P
$\overline{f}$	Average f
$\overline{EE}$	Average EE
$\overline{Throughput}$	Average $Throughput$
$G(n, \lambda)$	Random graph with n vertices and expected average degree λ
$SINR_i(t)$	Signal to noise plus interference ratio (SINR) required at link i at time t
$G_{ij}(t)$	Gain from transmitter of links j to receiver of link i at time t
$h_{ij}^{fast}(t)$	Fast fading from transmitter of links j to receiver of link i at time t
$h_{ij}^{slow}(t)$	Slow fading from transmitter of links j to receiver of link i at time t
S_{DL}	Set of D2D links
S_{CL}	Set of cellular links
DL	D2D link
CL	Cellular link
$\eta_i(t)$	Residual power at link i at time t
π	A permutation of pseudo-vertices
$L(\pi)$	Set of all orders that could be generated by permuting vertices within a pseudo-vertex, while keeping the order of pseudo-vertices intact as in π.
$\sim_n$	$x(n) \sim_n y(n) \implies \lim_{n\to\infty} \frac{x(n)}{y(n)} = 1$
$\lesssim_n$	$x(n) \lesssim_n y(n) \implies \lim_{n\to\infty} \frac{x(n)}{y(n)} \leq 1$

$\gtrsim_n$	$x(n) \gtrsim_n y(n) \implies \lim_{n\to\infty} \frac{x(n)}{y(n)} \geq 1$
$h(n)$	$y(n) = h(x(n)) \iff x(n) \lesssim_n y(n)$
$H(x)$	$\lim_{x\to 0} \frac{H(x)}{x} = 0$ and $\lim_{x\to\infty} \frac{H(x)}{x} = 0$
$O(x)$	$f(x) = O(g(x))$ if $\exists x_0, c_0 \in [0, \infty]$ s.t., $\forall x \geq x_0\, f(x) \leq c_0 g(x)$
$\Omega(x)$	$f(x) = \Omega(g(x))$ if $\exists x_0, c_0 \in [0, \infty]$ s.t., $\forall x \geq x_0\, f(x) \geq c_0 g(x)$
$\Theta(x)$	$f(x) = \Theta(g(x))$ if $f(x) = \Omega(g(x))$ & $f(x) = O(g(x))$
$o(x)$	$f(x) = o(g(x))$ if $\lim_{x\to\infty} \frac{f(x)}{g(x)} = 0$

Table 1: Summary of Notations

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid advancement of mobile communication technology has inspired more and more mobile users to demand for varieties of applications over wireless networks. Media-rich mobile applications involving streaming of video and multimedia files require sufficiently high data rates. Nowadays users are using different devices to communicate and hence total number of wireless devices are increasing day by day. Each device requires a channel and a power to communicate. In traditional cellular network, devices communicate via the base station (BS) through a channel. Due to explosion of number of users under a BS and scarcity of channels, BSs are not been able to provide a channel to each users under it. Device to device (D2D) communication is a paradigm shift in cellular networks which helps reducing the load of BS and also the outage of mobile devices at the cell edge, through spectrum reuse. In D2D communication, a pair of proximity users can communicate directly among themselves with or without the help of a BS which are referred to as operator-controlled (OC) and device-controlled (DC) respectively [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. The architecture that supports D2D communication is primarily based on two tiers, namely macro-cell and device tiers [5]. The macro-cell tier consists of the conventional cellular communication where the BS communicates with a mobile device or a mobile device communicates with the BS by forming a cellular link

(CL). In device tier, two proximity users communicate by forming a direct device to device link (DL) between them. Depending upon the frequency reuse factor, D2D communications can be classified as underlay or overlay. In underlay scenario, DLs and CLs share the same set of channels whereas, in overlay scenario, DLs and CLs use different set of dedicated channels. In this thesis, we consider both operator-controlled and device-controlled scenarios and primarily deal with overlay type of communications in the device tier architecture of the network.

A pair of proximity D2D users forms a link and communicates directly over a channel. Each link has a transmitter and receiver and operates on a channel. Transmitter of each link allocates a power subject to the maximum available power at that time, to transmit signal to its receiver. Receiver of a link gets interference from the transmitter of every other link running on the same channel. Each link has a data rate requirement. To fulfill that data rate, signal to noise plus interference ratio (SINR) must be more than a predefined threshold.

Since channels are costly we have to minimize the maximum channel used at time t. Since power of transmitter of a link is a limited resource, we also have to minimize the total power used. Since users are moving and the network topology changes over time, channel and power allocated to a link may need to be changed over time to maintain the required SINR. Thus a channel assigned to a link at time $(t-1)$ may not remain as an interference-free channel anymore at time t and hence we have to do channel switching which invites switching delay and degrades quality of service (QoS). Channel switch also increases packet loss probability. To maintain QoS, we also have to minimize the total number of channel switches or perturbation. It is evident that if we can afford to allocate different channel to each link then no perturbation will be needed and also each link will be able to operate with minimum power. But in that case, maximum channel used will be large. This implies that there is a natural trade-off between maximum channel used, total perturbation and total power requirement. Owing to these trade-offs, we choose our minimization objective as 1) the maximum channel used, 2) the total perturbation 3) a linear combination of the maximum channel used and the total perturbation and 4) a linear combination of the maximum channel used and the total power requirement respectively.

For first three objectives we simplify the problem as follows. Note that here we are not considering total power requirement as a part of the minimization objective and leave its allocation to the base station. Since the channel gain is a decreasing function of distance, we can neglect the interference between two links if the distance between transmitter of one link and receiver of the other link is more than some threshold distance called the *interference range*. Thus we essentially considering two links are interfering if the transmitter of one link and the receiver of the other link are within the interference range. Hence the interference relationship of the active links can be considered as an *interference graph*, where each link is a vertex and two vertices have an edge between them if corresponding links are interfering with each other. Since two users forming a link reside in close proximity, a link can be represented by the point, the midpoint of the line joining the transmitter and receiver of the link, with respect to the large geometric region. Since both of the endpoints of a link are moving, the corresponding midpoint will also move with certain velocity. We term the velocity with which the midpoint of a link is moving, as the velocity of that link. We also assume that for the considered transmission time both users of a link are residing within the transmission range of each other, i.e., the link remains active for that time period. For a large geometric region, due to law of large number and central limit theorem, the average number of links per unit area will converge to a finite constant. Hence interference graph can be considered as a random graph whose average degree is a finite constant. Since interface graph evolves over time, the vertices of it remains the same because links are active for the considered transmission time. Note that minimizing the maximum channel (color) is equivalent to solving graph coloring problem which is a well-known NP-complete problem [11]. It is also known that providing $n^{1/\epsilon}$ approximation $\forall\epsilon > 0$ is NP-hard [12]. On the other hand, if we retain the previously allocated channels of the links at the current time instant, some monochromatic edges (*an edge whose both endpoints are colored with the same color*) may appear in the interference graph. We term the graph induced by these monochromatic edges as a conflict graph. Note that to produce minimum perturbation, we essentially have to recolor each vertex in the minimum vertex cover of the conflict graph. Since finding the minimum vertex cover of a graph is NP-complete [11], minimizing the total perturbation is also a NP-hard problem. It is also known that providing $2-\epsilon$ approximation $\forall\epsilon > 0$ is also NP-hard [13]. For the last objective, we consider the general problem of the allocation of both channel and power to the active links. More specifically, we allocate both channel and power to each of the active links such that the required SINR is satisfied for each link.

This thesis mainly deals with the resource allocation problem in D2D communication. It focuses on developing algorithms for channel and power allocation in presence of interference so that the subsequent allocations require minimum maximum channel, power and channel switching. It is evident that for each of these problems, either providing an approximation algorithm with approximation ratio better than a constant factor, or providing any approximation ratio at all, is also NP-hard [12, 13]. Hence authors developed different heuristic approaches to solve these problems. These heuristic approaches mainly performed a sophisticated random search to find the optimum, but they did not provide a theoretical guarantee of hitting the optimum. On contrary, this thesis focuses on developing expected polynomial time randomized algorithms to find the optimum. We essentially proposed centralized, distributed as well as parallel algorithms to solve these problems. For each problem we have theoretically analysed the performance our algorithms. These theoretical results are also validated through extensive simulations. Finally we compared our results with state-of-the-art algorithms, both theoretically and through simulations.

1.1 Related Literature

Several authors [14, 15, 16, 17, 18] have studied the problem of resource allocation in D2D communication. A nice survey of various resource allocation schemes in D2D communication can be found in [8]. In [19] authors propose a joint mode selection and resource allocation solution based on branch-and-bound method. They developed low-complexity algorithms according to the network load. Here one CL can only be paired with one DL. In [20] authors discussed an analytical model of resource allocation. They use Shannon's capacity based approach for mode selection, channel and power allocation to maximize the data rate. They considered that one CL can be paired up with only one DL. In [15, 16, 21], energy efficient mode selection techniques were discussed. In [22] authors discussed a energy saving coding design. In [23], a power minimization solution for the joint channel allocation and mode selection problem in D2D communication is proposed. In [24] a resource allocation and data transmission procedure is described for operator controlled D2D communication. In [25] different strategies to minimize power were discussed. In [26] authors optimizes resource allocation and power control between the cellular and D2D connections that share the same resources subject to maximum transmit power or energy limitation. They apply greedy algorithm for channel and power allocation. In their set up one CL can be paired with only one DL. In [27] the close relationship between power allocation and channel allocation had been shown. In [28, 29, 30] authors propose various channel assignment algorithms where both cellular and D2D users share channels. In this underlaid scenario, cellular and D2D users may interfere with each other. Here, one channel could be used by one cellular user only whereas, a single channel might be used by multiple D2D users. In [31, 32, 33] authors developed channel assignment algorithms for D2D communication through graph coloring based approach. In this thesis, we adopt the graph coloring based approach and develop various schemes for minimizing maximum channel/color, total perturbation and total power.

1.1.1 Minimizing the maximum channel

The problem of minimizing the maximum color required to color the interference graph such that no monochromatic edge exists, is essentially the classical graph coloring problem, which is a well-known NP-complete problem [11]. It is also known that providing $n^{1/\epsilon}$ approximation solution, $\forall \epsilon > 0$, of the graph coloring problem is also NP-hard [12]. A good survey of heuristic algorithms for graph coloring can be found in [34, 35]. Recently a reinforcement learning based local search algorithm [36], a modified cuckoo algorithm [37], a hybrid evolutionary algorithm [38], a feasible and infeasible search based algorithm [39] and a parallel ordering based algorithm [40] are developed for graph coloring problem. Greedy coloring is the natural choice for this problem, which visits vertices of interference graph following an order, and while visiting a vertex v, it puts the minimum color that is absent in all neighbors of v. Since performance of greedy coloring is order dependent, several authors have proposed methods to find good orders such as *largest-degree-first* [41], *Kempe-order* [42] and *Dsature-order* [43]. Largest-degree-first sorts the vertices of a graph according to non-increasing order of their degrees. Authors of [41] have shown that with this ordering a graph having degree sequence $d_1 \geq d_2 \geq \cdots \geq d_n$, has maximum color $\leq 1 + \max_i \min(d_i, i-1)$. Kempe-order is generated as follows. We find a vertex with minimum degree in the graph and *push* it into a *stack* and *remove* it from the graph. Recursively apply this step on the *residual graph* till the graph become empty. Then *pop* the elements from the stack and this will give the Kempe-order. Dsature-order is generated as follows. Choose a random vertex and *enqueue* it into a *queue*. Calculate the *saturation degree* of all the vertices, not in queue, as defined bellow. The saturation degree of a vertex is the number of it's neighbors in the queue. Now choose the vertex with the maximum saturation degree and enqueue it into the queue. Continue this process till the queue contains all the vertices of the graph. *Dequeue* the elements from the queue and consider this order as the Dsature-order. Since finding an optimum order, among all possible $n!$ orders, is a hard problem, several authors have developed different search heuristics to find a near-optimum order. Some of them are based on simulated annealing [44], genetic algorithms [45] and tabu-search [46, 47]. Simulated annealing is a probabilistic heuristic where the probability of accepting a worse solution is decreased over time. In genetic algorithm, a set of orders is considered initially. They had been subject to mutation, cross-over, and selection to generate a set of better orders. Tabu-search minimizes the objective gradually, keeps a record of previous steps to prune the chance of repetition, and chooses a worsening move when stuck in a local optimum. It is evident that most of these heuristics may eventually reach a poorer order than the current best order during the search process. In this thesis, we developed a randomized algorithm which given the interference graph $g(t) = (V(t), E(t))$, solves the channel allocation problem via graph coloring formulation in expected $O(|V(t)| + |E(t)|)$ time and space complexities. The salient feature of our algorithm is that it never searches poorer order than the current best order .

1.1.2 Minimizing the total perturbation

The problem of assigning colors to the vertices of the interference graph such that total perturbation is minimized, is essentially equivalent to find a minimum vertex cover of the conflict graph, which is also an NP-complete [11] problem. It is also known that providing $2 - \epsilon$ factor algorithm $\forall \epsilon > 0$ for minimum vertex cover is also NP-hard [13]. A 2-approximation algorithm to find vertex cover can be found in [48]. There also exist several decentralized algorithms for finding vertex cover of a graph [49, 50, 51, 52]. As argued in [53] and to the best of our knowledge a few work had been done to maintain coloring of a graph which evolves over time with the objective of minimizing total perturbation. Authors in [53, 54, 55] have studied this problem with the property that at time t only one edge could appear or disappear. They call this incident as an update and try to maintain the coloring of the graph on a per-update basis. They fix a maximum color for the graph and upon occurrence of an update, change the color of some other vertices so that the proper coloring of the graph is maintained. To do that they partition the vertex set of interference graph into different subsets and arrange the subsets into different labels. Upon occurrence of an update, they exchange vertices between different labels. In their approaches they maintain various in-variants and use additional data structures. In [53] authors proposed a bucket based approach, which has later been improved by [54] in terms of required time per-update. In [55] authors proposed a randomized algorithm which partitions interference graph into different labels and maintains the expected number of colors of interference graph $g(t)$ in $\log(\Delta(g(t)))$ time per-update. Here $\Delta(g(t))$ is the maximum degree of $g(t)$. In [56] a dynamic-fit algorithm is developed to solve the time varying graph coloring problem for selected classes of graphs like trees and products of graphs. An agent-based algorithm [57] and greedy approach [58] are developed to color dynamic graphs. In our problem set up, at time t, some monochromatic edges may appear in $g(t)$ and we consider recoloring them together such that total perturbation $A(t)$ is minimized. To minimize $A(t)$, we propose a centralized and a decentralize differential coloring (DC and DDC) technique and a random coloring (RC) technique.

1.1.3 Minimizing both maximum channel and total perturbation

Owing to the natural trade-off between maximum color $Y(t)$ and total perturbation $A(t)$, in [59] authors first considered the cost function as $f(t) = Y(t) + \alpha A(t)$ and proposed a SNAP and a SMASH approach to solve the problem. SNAP colors $g(t)$ using *incremental coloring* (IC) [59], $\forall t$. IC colors vertices of $g(t)$ following a random order and while coloring a vertex it applies first-fit if there is no incident monochromatic edge into it. First-fit essentially puts the minimum color to a vertex which is absent in all of its neighbors. On the other hand, SMASH minimizes $A(t)$ through minimizing the size of vertex cover of the conflict graph. In $k+1$ time interval SMASH builds a SMASHed graph $g_s^k(t) = \bigcup_{\tau=0}^{k} g(t+\tau)$ and colors it applying IC, and retains that color for next k time instances. Though in this case, perturbation occurs only once in $k+1$ time instances, but to build a SMASHed graph, they require *future information*, which may not be available. In this thesis, we have proposed a perturbation sensitive greedy coloring (PGC) technique which can find minimum $f(t) = Y(t) + \alpha A(t)$ given $g(t)$ and channel allocation at time $(t-1)$, in expected $O(n)$ time and space complexities. Since coloring of future graphs are functions of coloring of current graph, to reduce expected cost $\mathbb{E}[f(t)]$ we have also proposed a graph union based approach (GU) which uses the *past information* to build a union graph and colors it with PGC. To minimize $\mathbb{E}[f(t)]$, we also propose a geometric prediction (GP) based approach which using the current position and maximum velocity of the users, builds a predicted graph and color it using DC.

1.1.4 Minimizing both maximum channel and total power

There are several algorithms which deal with the problem of optimizing both total power $P(t)$ and maximum channel $Y(t)$. Authors choose different objectives while allocating channels and powers. Most of them set the objective to maximize as spectral efficiency (SE) or energy efficiency (EE). SE [60] and EE [61, 62] are two well adopted maximization objective in power and channel allocation in D2D communication. In SE data rate per spectrum is maximized whereas, in EE, data rate per spectrum per energy unit is maximized. In [27] authors formulate the resource allocation problem as a non-convex optimization problem to minimize power. In [17] authors discussed the method to maximize the minimum weighted energy efficiency of D2D links while ensuring maximum data rate in cellular links. In [63, 64] authors propose matching based algorithms for EE maximization. They consider that each DL can pairs with at most one CL and both can operate on the same channel. They form a two-step solution of the problem where the first step differs. In the first step of the algorithm proposed in [63], a game theoretic approach is employed to analyze the interactions and correlations among the users and subsequently an iterative power allocation algorithm is developed to establish mutual preferences. Whereas, the optimal power that is required to optimize EE for each DL-CL pair is found in the first step of the algorithm proposed in [64]. In the second step, both algorithms find a stable matching of DL and CL using Gale-Shapley algorithm [65]. It is evident that Gale-Shapley algorithm provides a pareto-optimal solution of stable matching problem. Note that the performance of the second step is highly dependent on the performance of the first step and as a result, stable matching may produce highly sub-optimal solution. In [10] authors propose a mixed integer non-linear programming (MINLP) for EE maximization. In their problem setup, the number of DLs is at most the number of CLs and also one DL can be paired with at most one CL. They then solve the EE maximization problem in two-step. For a given DL-CL pairing, they propose a method to minimize the energy consumption. To efficiently form the DL-CL pairs they propose a random switch-based iterative (RSBI) algorithm. RSBI starts with a random DL-CL pairing and then taking random steps tries to minimize the energy consumption. Authors of [66] propose a graph coloring and group reforming based D2D resource allocation and power control (DRAPC) algorithm. Here multiple DL can be paired with one CL. They first run a color based grouping algorithm for channel and power allocation. Then they run different algorithms for power control, member adding and throughput rising. Since different algorithms are interdependent, performance of one algorithm depends on the other. Also there is no guarantee to hit optimum.

Since both power and channels are scarce resources, we set our minimization objective as $f(t) = Y(t) + \alpha P(t)$ and developed a randomized joint channel and power allocation (RJCPA) algorithm to find the near optimum solution. We have shown that RJCPA also minimizes EE efficiently.

1.2 Scope of the thesis

In this thesis, we formulate the resource allocation problem as a cost minimization problem where cost is defined as $Y(t)$, $A(t)$, $Y(t) + \alpha A(t)$ or $Y(t) + \alpha P(t)$. We consider $C(t) = (c_i(t))$ and $X(t) = (x_i(t))$ as the channel and power vector, where $c_i(t) \in \{1, 2, \cdots n\}$ and $x_i(t) \in [0, \eta_i(t)]$ are the channel and power allocated to link i at time t. Here $\eta_i(t)$ is the maximum power available at the transmitter of link i at time t. We first consider the problem of minimizing the maximum color $Y(t) = \max_i c_i(t)$. To minimize $Y(t)$ we essentially have to solve the graph coloring problem. Greedy coloring is the most natural solution of this problem. Greedy coloring visits the vertices of $g(t)$ following an order S and while visiting a vertex v, it puts the minimum color that is absent in all of its neighbors. Here we assume that colors are positive integers starting from 1 and denote the maximum color $Y(t)$ used in $C(t)$ as the *span*. We show that the set of orders with span $Y(t) \leq k$ can be partitioned into disjoint subsets of *equivalent orders*. Next, we propose a *selective search* (SS) algorithm, which takes ρ as an input parameter, selects $\geq \rho$ orders each from a different set of equivalent orders with high probability, applies greedy coloring on them, and returns the color vector with minimum span. We analytically show that by evaluating the same number of orders, SS performs better than greedy coloring with high probability. We further propose an *incremental search heuristic* (ISH) which ρ_1 times executes SS with parameter ρ_2 and returns the color vector with minimum span. A parallel version of ISH called PISH is also proposed, which essentially calls SS ρ_1 times in parallel. We prove that ISH solves the graph coloring problem in expected $O(|V(t)| + |E(t)|)$ time and space complexity. We have evaluated ISH and PISH on 136 challenging benchmarks and shown that they significantly outperform 10 existing state-of-the-art algorithms mentioned in [67, 68, 37, 69, 70]. Finally we have validated our theoretical findings by evaluating ISH and greedy coloring on random graphs. This work is presented in Chapter 2.

Next we consider the problem of minimizing the total number of channel switches or total perturbation $A(t) = \sum_i I_i(C(t), C(t-1))$. Here $I_i(C(t), C(t-1)) = 1$ if $c_i(t) \neq c_i(t-1)$ else 0. We first show that minimizing $A(t)$ given $g(t)$ and $C(t-1)$ is a NP-hard problem. Then to minimize $A(t)$, we propose a centralized and a decentralized differential coloring (DC and DDC) technique and a random coloring (RC) technique. DC and RC are centralized and runs at the base station, while DDC runs at the user end. All these algorithms first set $C(t) = C(t-1)$ which in effect may create some monochromatic edges in $g(t)$. Let $g_c(t)$ be the graph induced by those monochromatic edges. We then apply a 2-approximation maximal matching based technique to find a minimum vertex cover $V_c(t)$ of $g_c(t)$. DC removes the color of the vertices in $V_c(t)$ and then recolors them by applying first-fit on a random order of the vertices. First-fit puts to a vertex the minimum color that is absent in all of the neighbors of that vertex. RC does thfit instead of first-fit. Random-fit puts to a vertex a random color from $\{1, 2, \cdots, k\}$ that is absent in ae same thing as DC with the exception that it applies random-ll of the neighbors of that vertex. Here $k \geq \Delta(g(t)) + 1$, where $\Delta(g(t))$ is the maximum degree of $g(t)$. Using the channel state information from the base station, and performing message exchange among the links within $\leq r$ distance apart, DDC minimizes $A(t)$. It applies the maximal matching based algorithm on $g_c(t)$, without actually building the $g_c(t)$, in a decentralized manner. We theoretically calculate $\mathbb{E}[A(t)]$ and $\mathbb{E}[Y(t)]$ produced by DC, DDC and RC and compare with existing approaches mentioned in [53, 54, 55]. This work is presented in Chapter 3.

Next we consider the problem of minimizing the combined objective $f(t) =$

$Y(t) + \alpha A(t)$. Since $g(t)$s are evolving over time, $f(t+1)$ is a function of $C(t)$. Considering the sequence of $g(t)$s, we aim to minimize $\mathbb{E}[f(t)] = \mathbb{E}[\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} f(t)]$. With this objective we propose a geometric prediction (GP) based approach and a graph union (GU) based approach. GP uses the current position and maximum velocity of the users to *predict the future*, on the other hand, GU uses the *past information* to minimize $\mathbb{E}[f(t)]$. Considering k as an input parameter, in every $k+1$ time intervals, GP builds a predicted graph $g_p^k(t) \supseteq \bigcup_{\tau=0}^{k} g(t+\tau)$ which is a super-graph of the union of the current and all possible graphs that may appear within next k time intervals and color it using DC. Then it retains the color of this predicted graph for next k time intervals. Since perturbation can only occur once in current and next k time intervals, perturbation is reduced by $k+1$ factor, at the expense of increased color requirement. Next we propose a perturbation sensitive greedy coloring (PGC) technique which finds the minimum $f(t)$ given $g(t)$ and $C(t-1)$, in expected $O(n)$ time and space complexities, which is asymptotically minimum among all algorithms which can solve this particular problem. In contrast to GP, GU builds a union graph $g_u^k(t) = \bigcup_{\tau=0}^{k} g(t-\tau)$ and colors it using PGC. We calculate $\mathbb{E}[f(t)]$ produced by GP and GU and compare that with existing approaches mentioned in [59, 53, 54, 55]. This work is presented in Chapter 4.

Finally we consider the joint power and channel allocation problem (JPCAP) whose objective is to minimize the combined objective $f(t) = Y(t) + \alpha P(t)$, where $P(t) = \sum_i x_i(t)$ is the total power allocated at the links at time t. We reduce the problem of minimizing $f(t)$ to the classical graph coloring problem and thereby show that it is NP-hard and also providing $n^{1/\epsilon}$ approximation to JPCAP $\forall \epsilon > 0$ is NP-hard. Next we propose a mixed integer linear programming (MILP) formulation for this problem and subsequently develop a greedy channel and power allocation (GCPA) algorithm for it. GCPA works by taking an order of the links as input. We show that there exists an order of the links on which if GCPA is applied it will provide an optimal solution. Then we develop a method to search orders efficiently. We show that an order is equivalent to many orders. We develop an incremental algorithm (IA) which searches orders from different equivalent sets and thereby evaluating less number of orders, it essentially explores large number of orders. Finally, using IA, we design a randomized joint channel and power allocation (RJCPA) algorithm to find the near optimum solution. We also theoretically calculate the expected cost $\mathbb{E}[f(t)]$ and energy efficiency (EE) produced by RJCPA. Moreover, we identify some special cases where RJCPA can produce optimal result in expected polynomial time. We perform extensive simulations to show that RJCPA outperforms both the two-step approach [64] and RSBI algorithm [10] with respect to both cost and EE significantly. Finally we validate our theoretical findings through simulations. This work is presented in Chapter 5.

1.3 Organization of Thesis

In chapter 2 we consider the minimization of $Y(t)$ and for that we propose randomized algorithm, called incremental search heuristic (ISH) and its parallel version PISH, which solves it in expected polynomial time.In chapter 3 we minimize $A(t)$, for given $Y(t) \leq Y_{\max}$ and propose a centralized and a decentralized differential coloring algorithm (DC and DDC) and a random coloring (RC) algorithm to solve it. In Chapter 4 we consider the minimization objective as $Y(t) + \alpha A(t)$ and propose a geometric prediction (GP) based algorithm and a graph union (GU) based algorithm to solve it. GP uses predicted future information and GU uses past information. In chapter 5 we consider the minimization objective as $Y(t) + \alpha P(t)$ and develop a randomized joint channel and power allocation (RJCPA) algorithm to solve this problem. Finally in chapter 6 we conclude the thesis.

Chapter 2

Minimizing the maximum channel

2.1 Introduction

In this chapter, we formulate the resource allocation problem as a cost minimization problem where cost is defined as $Y(t)$. In D2D communication several D2D links are placed in a large geometric region. We are considering here D2D overlaid scenario, where a separate set of channels are available for allocating the D2D links. Each D2D link i requires a channel $c_i(t)$ to communicate at time t. Since channels are costly, we have to reuse the same channel/color to multiple links. A pair of links operating on the same channel may interfere with each other if the receiver of one link resides within the interference range of the transmitter of the other link. Hence at time t, the interference relationship among the active device to device links (DLs) can be modelled as an interference graph $g(t) = (V(t), E(t))$, where each DL is considered as a vertex $v \in V(t)$ and a pair of vertices forms an edge if their representing links are interfering to each other. So our problem is to find a channel/color vector $C(t) = (c_i(t))$ of $g(t)$ such that the span $Y(t) = \max_i c_i(t)$ gets minimized and there does not exist any monochromatic edge in $g(t)$. It is evident that the problem is equivalent to graph coloring problem which is one of the fundamentally known NP-complete [11] problem. Since for this particular problem $Y(t)$ depends only on $g(t)$, from here onward in this chapter, we will consider $g(t)$, $C(t)$, $V(t)$, $E(t)$ as G, C, V and E respectively.

Greedy coloring is the natural choice for this problem, which visits vertices of G following an order S and while visiting a vertex v, it puts the minimum color that is absent in all neighbors of v. Since performance of greedy coloring is order dependent, several authors have proposed methods to find good orders, as summarized in the related work section of the introduction. However, most of the existing heuristics may eventually reach a poorer order than the current best order during the search process. In contrast to that we propose a *selective search* (SS) algorithm which never searches poorer order than the current best order. SS starts with an order S and applies greedy coloring on it and returns a color vector with span k. It then partitions V into k independent sets having same color, called pseudo-vertices. To reduce span further, it chooses a random permutation of the k pseudo-vertices, a random order S' from it, and applies greedy coloring on S'. It can be shown that the span generated by the new order is $\leq k$. SS repeats the procedure considering S' as S. SS abort the process when ρ consecutive unsuccessful attempts to reduce span occur, where ρ is a input parameter. We analytically show that SS is better than greedy coloring when executed on the same number of orders, with very high probability. We show that, SS starting with a particular order S may not hit the optimum even if all $k!$ permutations are considered. To evaluate each order with a positive probability, we propose an incremental search heuristic (ISH) which simply calls SS with parameter ρ_2, ρ_1 times and reports the minimum span produced by them. We have shown that ISH solves the graph coloring problem in expected $O(|V| + |E|)$ time and space complexities, which is minimum among all possible randomized algorithms for this problem. For time efficient solution using multiple possessors we also developed a parallel versions of SS and ISH called PSS and PISH respectively. We simulate ISH and PISH on 136 well-known benchmarks and show that both ISH and PISH hits the best known span. Also time taken by ISH and PISH are significantly better than 10 states of the art algorithms. We validate our theoretical findings by evaluating ISH and greedy coloring on random graphs as well.

2.2 Key Ideas

Consider a graph $G(V, E)$ with n vertices $v_1, v_2, \cdots, v_n$, where V is the set of vertices and E is the set of edges. Let $S = (v_{l_1}, v_{l_2}, \cdots, v_{l_n})$ be an arbitrary order of the vertices of G, where $1 \leq l_k \leq n$. Assuming colors are positive integers starting from 1, let $c(v_{l_k})$ be the color of vertex v_{l_k} obtained by applying greedy coloring on G following order S. Recall that greedy coloring colors the vertices of a graph following a specific order of the vertices and while coloring a vertex it puts the *minimum* color that is absent in all of its neighboring vertices. Hence $c(v_{l_k})$ depends only on the colors assigned to the vertices $v_{l_1}, v_{l_2}, \cdots, v_{l_{k-1}}$ that appear before v_{l_k} in S. Moreover, greedy coloring always produces a *no-hole* coloring. A coloring of G is a no-hole coloring if it uses all colors between 1 and its maximum color. Given S, let $C = (c(v_1), c(v_2), \cdots, c(v_n))$ be the *color vector* obtained by applying greedy coloring on G following S. Note that greedy coloring essentially is a function $\mathbb{F}$ which takes graph G and an order S of its vertices as input and produce a color vector C as output. That is $C = \mathbb{F}(G, S)$. Note that in the color vector, colors of the vertices are stored in the ascending order of their vertex indexes. That is, if $l_1 = 5$, then v_5 will be colored first by the greedy coloring but its color will appear in the fifth position in C. With respect to the resulted color vector obtained by greedy coloring, an order may be considered as equivalent to another order as formally defined in Definition 1.

Definition 1 (Equivalent order) *Let S_1 and S_2 be two orders of the vertices of G. Then S_1 and S_2 are equivalent to each other if and only if $\mathbb{F}(G, S_1) = \mathbb{F}(G, S_2)$.*

The span of a color vector C denoted by $span(C)$ is the total number of distinct colors in C. As colors start from 1 and greedy coloring produces no-hole coloring, $span(C)$ is essentially same as the maximum color used in C.

Definition 2 *Let S_1 and S_2 be two orders of the vertices of G. Then $S_1 \lhd S_2$ if and only if $span(\mathbb{F}(G, S_1)) \leq span(\mathbb{F}(G, S_2))$.*

Note that $C = \mathbb{F}(G, S)$ essentially partitions the graph into $k = span(C)$ vertex disjoint independent sets each of which contains all the vertices of a particular color. We call each such independent set as a pseudo-vertex as formally defined in Definition 3.

Definition 3 (Pseudo-vertex) *Let S be an order of the vertices of G and $C = \mathbb{F}(G, S)$ with $k = span(C)$. A pseudo-vertex $V_i = \{v \in V : c(v) = i\}$ is a subset of vertices of G all of which get color i in C, where $1 \leq i \leq k$.*

Definition 4 (Cardinal order) *An order $S = (v_{l_1}, v_{l_2}, \cdots, v_{l_n})$ of the vertices of G is said to be a cardinal order if $C = \mathbb{F}(G, S)$ such that $c(v_{l_1}) \leq c(v_{l_2}) \leq \cdots \leq c(v_{l_n})$.*

Let $\Pi(S)$ be the set of all permutations of $V_1, V_2, \cdots, V_k$, and $\pi = (V_{l_1}, V_{l_2}, \cdots, V_{l_k}) \in \Pi(S)$ where $1 \leq l_i \leq k$. Let $L(\pi)$ be the set of all orders generated from π by permuting vertices within the same pseudo-vertex but keeping the order of the pseudo-vertices intact. Then all orders in $L(\pi)$ are considered as orders represented by π. Let $N_k = |L(\pi)| = n_1!n_2! \cdots n_k!$, where $n_i = |V_i|$, $\forall i$. Let $\pi_0(S) = (V_1, V_2, \cdots, V_k) \in \Pi(S)$. It is evident that each order $S' \in L(\pi_0(S))$ is a cardinal order.

Theorem 2.2.1 *Let S be an order of vertices of G and $V_1, V_2, \cdots, V_k$ be the k pseudo-vertices of $C = \mathbb{F}(G, S)$. Let $\pi = (V_{l_1}, V_{l_2}, \cdots, V_{l_k}) \in \Pi(S)$. All orders $\in L(\pi)$ are equivalent to each other.*

Proof : Let $S_1, S_2 \in L(\pi)$ such that $S_1 \neq S_2$. Also assume that $S_1 = (v_{r_1}, v_{r_2}, \cdots v_{r_n})$, where $1 \leq r_i \leq n$ for all i. Let $C_1 = \mathbb{F}(G, S_1)$ and $C_2 = \mathbb{F}(G, S_2)$. To show that S_1 is equivalent to S_2, we have to prove that $c_1(v_{r_i}) = c_2(v_{r_i})$ for all v_{r_i}. We prove this by induction on i. Since V_{l_1} is an independent set $c_1(v_{r_1}) = c_2(v_{r_1}) = 1$. Hence the base case is done. Our induction hypothesis is, for all vertex v_{r_j} appears before v_{r_i} in S_1, $c_1(v_{r_j}) = c_2(v_{r_j})$. We now left to prove $c_1(v_{r_i}) = c_2(v_{r_i})$. Let $v_{r_i} \in V_{l_m}$. Note that $c_1(v_{r_i})$ and $c_2(v_{r_i})$ depend only on the colors of those vertices which appear before v_{r_i} in S_1 and S_2 respectively. All the vertices of $V_{l_1}, V_{l_2}, \cdots, V_{l_{m-1}}$ and some vertices of V_{l_m} may only appear before v_{r_i} in both S_1 and S_2. Since V_{l_m} is an independent set, eventually $c_1(v_{r_i})$ and $c_2(v_{r_i})$ depend only on the vertices that belong to $\bigcup_{j=1}^{m-1} V_{l_j}$. But according to our induction hypothesis, $c_1(v_{r_j}) = c_2(v_{r_j})$ for all $v_{r_j} \in \bigcup_{j=1}^{m-1} V_{l_j}$. Hence the proof.

Theorem 2.2.2 *Let S be an order and $V_1, V_2, \cdots, V_k$ be the k pseudo-vertices of $C = \mathbb{F}(G, S)$. Then for each cardinal order $S' \in L(\pi_0(S))$, S' is equivalent to S.*

Proof : Let $S' = (v_{r_1}, v_{r_2}, \cdots, v_{r_n})$ and $C' = \mathbb{F}(G, S')$, where $1 \leq r_i \leq n$ for all i. To show S' is equivalent to S, we have to show that $c(v_{r_i}) = c'(v_{r_i})\ \forall i$. We will apply induction on i to prove this. For $i = 1$ the proof is trivial. Hence the base case is done. Our induction hypothesis is, $c(v_{r_j}) = c'(v_{r_j})$ for all vertices appearing before v_{r_i} in S'. Note that $c'(v_{r_i})$ depends only on the colors of those vertices which appear before v_{r_i} in S'. It is evident that all the vertices which have been colored with less than $c(v_{r_i})$ in C must appear before v_{r_i} in S' according to the construction of S'. Hence $c'(v_{r_i})$ cannot be less than $c(v_{r_i})$. Some vertices which have been colored with $c(v_{r_i})$ in C may also appear before v_{r_i} in S'. But all such vertices belong to an independent set in G. Hence $c(v_{r_i}) = c'(v_{r_i})$. Hence the proof.

Theorem 2.2.3 *Let S be an order and $V_1, V_2, \cdots, V_k$ be the k pseudo-vertices of $C = \mathbb{F}(G, S)$. If $\pi = (V_{l_1}, V_{l_2}, \cdots, V_{l_k}) \in \Pi(S)$, $\forall S' \in L(\pi)$, $S' \lhd S$.*

Proof : Let $S' = (v_{r_1}, v_{r_2}, \cdots, v_{r_n})$, where $1 \leq r_i \leq n$ for all i. Let $C = \mathbb{F}(G, S)$ and $C' = \mathbb{F}(G, S')$. Let v_{r_i} belongs to the pseudo-vertex which is in m_i-th position in π.

Our claim is that $c'(v_{r_i}) \leq m_i \; \forall i$. Since $1 \leq m_i \leq k$, if our claim is true, then we can immediately conclude that $span(C') \leq k = span(C)$.

So we are left to prove our claim. We prove this by induction on i. Since all vertices belong to the first pseudo-vertex in π get color 1 in C', our claim is trivially true for all such vertices. Hence base case is done. We now consider in induction hypothesis that for all vertices appear before vertex v_{r_i} in S' our claim is true. This implies that our claim is true for all vertices which belong to the pseudo-vertices of C appeared before m_i-th position in π. Note that $c'(v_{r_i})$ depends only on the colors of those vertices which appear before v_{r_i} in S'. There are $m_i - 1$ pseudo-vertices before m_i-th position in π. Hence according to our induction hypothesis, the colors of the vertices belong to those pseudo-vertices are at most $m_i - 1$. Note that some vertices belong to the pseudo-vertex in m_i-th position may also appear before v_{r_i} in S'. Since each pseudo-vertex is an independent set, $c'(v_{r_i})$ does not depend on those vertices. Hence $c'(v_{r_i}) \leq m_i$. Hence the proof.

Example 1 *We now illustrate the essence of theorems 2.2.1, 2.2.2 and 2.2.3 using the graph shown in Figure 2.1(1). Greedy coloring on $S = (v_1, v_2, v_3, v_4)$ produces color vector $C = (1, 1, 2, 3)$ with pseudo-vertices $V_1 = \{v_1, v_2\}$, $V_2 = \{v_3\}$ and $V_3 = \{v_4\}$ respectively. Note that $\Pi(S) = \{(V_1, V_2, V_3), (V_1, V_3, V_2), (V_3, V_2, V_1), (V_3, V_1, V_2), (V_2, V_1, V_3), (V_2, V_3, V_1)\}$. Consider $\pi = (V_3, V_1, V_2)$ and $\pi_0(S) = (V_1, V_2, V_3)$. Observe that $L(\pi) = \{(v_4, v_1, v_2, v_3), (v_4, v_2, v_1, v_3)\}$ and $L(\pi_0(S)) = \{(v_1, v_2, v_3, v_4), (v_2, v_1, v_3, v_4)\}$ respectively. Greedy coloring on both $S' = (v_1, v_2, v_3, v_4) \in L(\pi_0(S))$ and $S'' = (v_2, v_1, v_3, v_4) \in L(\pi_0(S))$ produces color vector $C' = C'' = (1, 1, 2, 3) = C$, so both S' and S'' are equivalent to S (Theorem 2.2.2). Greedy coloring on both $S' = (v_4, v_1, v_2, v_3) \in L(\pi)$ and $S'' = (v_4, v_2, v_1, v_3) \in L(\pi)$ produces color vector $C' = C'' = (2, 1, 2, 1)$ with pseudo-vertices $V_1 = \{v_2, v_4\}$ and $V_2 = \{v_1, v_3\}$ respectively as shown in Figure 2.1(2). So S' and S'' are equivalent to each other (Theorem 2.2.1). Since $span(C') = span(C'') < span(C)$, we get that $S' \lhd S$ and $S'' \lhd S$ (Theorem 2.2.3).*

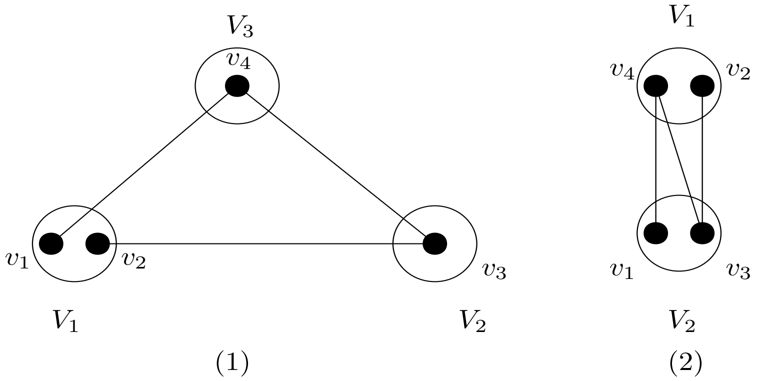

Figure 2.1: Vertices and pseudo-vertices

Theorem 2.2.4 *Given any coloring C' of G we could generate an order $S = (v_{l_1}, v_{l_2}, \cdots, v_{l_n})$ by sorting the vertices of G according to ascending order of their colors in C'. If $C = \mathbb{F}(G, S)$ then $span(C) \leq span(C')$.*

Proof : We claim that $c(v_{l_i}) \leq c'(v_{l_i})$ for all l_i, $1 \leq l_i \leq n$. We prove this by induction on i. For the first vertex, the claim is trivially true. So the base case is done. In induction hypothesis we assume that the claim is true for each vertex which appears prior to v_{l_i} in S. Let $N(v_{l_i})$ be the set of neighbors of v_{l_i} in G which appear before v_{l_i} in S. As S is constructed by sorting the vertices according to ascending order of their colors in C', we get $c'(v_{l_1}) \leq c'(v_{l_2}) \leq \cdots \leq c'(v_{l_i})$. Since $c'(v_{l_i})$ is a valid coloring of v_{l_i} in C', $N(v_{l_i})$ can not contain $c'(v_{l_i})$. In other words, $c'(v_{l_j}) \leq c'(v_{l_i}) - 1$ for $\forall v_{l_j} \in N(v_{l_i})$. From the induction hypothesis, we get $c(v_{l_j}) \leq c'(v_{l_j})$ for $\forall v_{l_j} \in N(v_{l_i})$. The previous two arguments together imply $c(v_{l_j}) \leq c'(v_{l_i}) - 1$ for $\forall v_{l_j} \in N(v_{l_i})$. Since C is obtained by greedy coloring, $c(v_{l_i})$ must be the minimum color which is not been used in any of the vertices of $N(v_{l_i})$, implying $c(v_{l_i}) \leq c'(v_{l_i})$. Hence the proof.

The following notations will be frequently used in the rest of this chapter: $x(n) \sim_n y(n) \iff \lim_{n\to\infty} \frac{x(n)}{y(n)} = 1$, $x(n) \lesssim_n y(n) \iff \lim_{n\to\infty} \frac{x(n)}{y(n)} \leq 1$ **and** $y(n) = h(x(n)) \iff x(n) \lesssim_n y(n)$**.**

2.3 Selective Search (SS) Algorithm

Given any order S, we can apply greedy coloring on S and generate $C = \mathbb{F}(G, S)$. Again from C we can construct k pseudo-vertices $V_1, V_2, \cdots, V_k$ where $k = span(C)$. It is also evident that for each order $S' \in L(\pi)$ where $\pi \in \Pi(S)$, $S' \lhd S$. Also all $N_k = \prod_{i=1}^{k} |V_i|! = h((\frac{n}{k}!)^k) = h((\frac{n}{ek})^n)$ (Using Stirling's approximation) orders belong to $L(\pi)$ are equivalent to S' and $\forall \pi_1, \pi_2 \in \Pi(S)$, $\pi_1 \neq \pi_2 \iff L(\pi_1) \bigcap L(\pi_2) = \emptyset$. From the above discussion we can think of a natural algorithm which can be stated as: Starting with a random order S build $C = \mathbb{F}(S, G)$. Consider a random permutation $\pi \in \Pi(S)$, a random order $S' \in L(\pi)$ and build $C' = \mathbb{F}(G, S')$. Repeat the process with the aim to improve the span. Terminate the process if no improvement is found in ρ consecutive orders, and return the color vector with minimum span among the orders considered. SS executes $h(\rho)$ steps, where a *step* is defined as the event of applying greedy coloring on an order and finding its color vector. Time complexity of executing a step is $O(|V| + |E|)$. Let K be the set of spans that could be produced by applying greedy coloring on all possible orders of vertices of G. Clearly $\chi(G) \leq |K| \leq \Delta(G) + 1$, where $\chi(G)$ is the chromatic number and $\Delta(G)$ is the maximum degree of G. Hence total time and space complexities of SS are $O(\rho|K|(|V| + |E|))$ and $O(|V| + |E|)$, where ρ is an input parameter. Formally SS is presented in Algorithm 1.

Remark 1 *Starting with an order S, SS might not hit the optimum even if it searches all the $k!$ permutations of $V_1, V_2, \cdots, V_k$. One possible condition on the vertices belongs to different pseudo-vertices such that further reduction of span by SS is impossible is as follows. Suppose a vertex u in V_x has at least one neighbor in each of the other pseudo-vertices V_y, $1 \leq y \leq k$, $x \neq y$. If this condition holds for all vertices in V_x and for all x, $1 \leq x \leq k$, SS cannot further reduce the span.*

Example 2 *We now elaborate the discussion in Remark 1 through an example. Consider the graph shown in Figure 2.2. Note that by applying greedy coloring on the order $S =$*

Algorithm 1: *Selective Search (SS) Algorithm*

```
Input: $G, \rho$
Output: $C$
Generate a random order $S$;
$C = C' = \mathbb{F}(G, S)$;
for $i = 1, 2, \cdots \rho$ do
    Consider a random permutation $\pi \in \Pi(S)$ and a random order $S' \in L(\pi)$;
    $C' = \mathbb{F}(G, S')$;
    if $span(C') < span(C)$ then
        $C = C'$;
        $S = S'$;
        Reset $i = 1$;
return $C$
```

$(v_1, v_2, v_3, v_4, v_5, v_6)$ *we get color vector* $C = (1, 1, 2, 2, 3, 3)$ *with* 3 *pseudo-vertices* $V_1 = \{v_1, v_2\}$, $V_2 = \{v_3, v_4\}$ *and* $V_3 = \{v_5, v_6\}$ *respectively. Since the graph is bipartite, optimum span is* 2*. Note that if we start SS with S then the condition mentioned in Remark 1 is satisfied and hence, even considering all* 3! *permutations of* V_1, V_2 *and* V_3 *we still stuck at span* 3 *and can't ever achieve the optimum.*

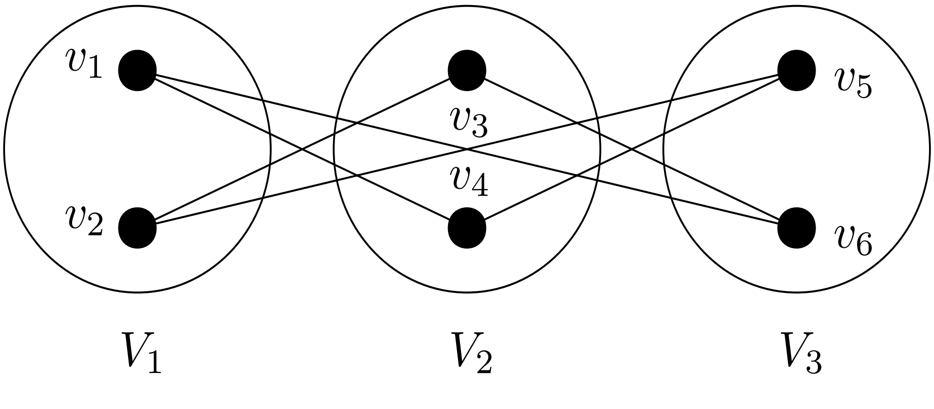

Figure 2.2: Example of a situation when SS cannot improve span

2.3.1 Analysis of SS

So far, we get that SS starts with an order having span k and basically applies greedy coloring on some $h(\rho)$ specifically chosen orders having span $\leq k$. The valid question is that if we randomly choose the same number of orders and apply greedy coloring on them, would it give the same result, or SS is likely to give a better result. To show the benefit of SS, in this sub-section, we will find the condition for which the optimum hitting probability of SS would be higher than greedy coloring when applied on the same number of orders.

Let S_1 and S_2 be two distinct random orders such that upon applying greedy coloring on them, they produce the span $\leq k$. Note that the set of $h(N_k)$ orders equivalent to S_1 and the set of $h(N_k)$ orders equivalent to S_2 might not be disjoint. Hence by evaluating ρ distinct random orders, greedy coloring essentially evaluates $\leq_n h(\rho N_k)$ orders. On the other hand, if SS starts with an order with span $\leq k$ and considers ρ orders from ρ distinct permutations, it eventually evaluates $h(\rho N_k)$ orders.

Let p_{SS} be the probability that SS starting with an order S having span $\leq k$, essentially evaluates $h(\rho N_k)$ distinct orders. Note that this is equivalent to evaluating ρ orders from ρ distinct permutations. Let p_g be the probability that greedy coloring when applied on ρ random orders each having span $\leq k$, essentially evaluates $h(\rho N_k)$ distinct orders. Note that this is equivalent to evaluating ρ orders such that the sets of their corresponding equivalent orders are mutually disjoint.

Theorem 2.3.1 *Optimum hitting probability of SS starting with an order S having span $\leq k$ is asymptotically greater than the same of greedy coloring when both executed on ρ orders, where $1 < \rho < \min(\frac{n!}{h(N_k)}, k!)$.*

Proof : It is evident that we only have to show $p_g \leq_n p_{SS}$. Note that $\binom{n}{x_1, x_2, \cdots, x_m}$ is the number of ways of splitting a set of n elements into the disjoint sets of $x_1, x_2, \cdots, x_m$ elements. Also $\binom{n}{r}$ is the number of ways of choosing r elements from a set of n elements. It is evident that total number of ways to choose ρ *distinct* permutations from $k!$ permutations is

$$\binom{k!}{1, 1, \cdots, 1, k! - \rho}.$$

Also the total number of ways to choose ρ permutations (*not necessarily distinct*) from $k!$ permutations is

$$\binom{k!}{1}^{\rho}.$$

Hence

$$\begin{aligned} p_{SS} &= \frac{\binom{k!}{1,1,\cdots,1,k!-\rho}}{\binom{k!}{1}^{\rho}} \\ &= \prod_{i=1}^{\rho-1}\left(1-\frac{i}{k!}\right) \in \left(\left(1-\frac{\rho-1}{k!}\right)^{\rho-1}, \left(1-\frac{1}{k!}\right)^{\rho-1}\right), \end{aligned} \qquad (2.1)$$

where $\chi(G) \leq k \leq n$. Again total number of ways to choose ρ orders each with span $\leq k$, from $n!$ orders, such that greedy coloring eventually evaluate $h(\rho N_k) = \rho n!x$ *distinct* orders is

$$\binom{n!}{n!x, n!x, \cdots, n!x, n!(1-\rho x)},$$

where $x = \frac{h(N_k)}{n!} \in (0, \frac{1}{\rho})$. On the other hand, the number of ways to choose ρ orders (*not necessarily distinct*) each with span $\leq k$, from $n!$ orders, is

$$\binom{n!}{n!x}^{\rho}.$$

Hence

$$\begin{aligned} p_g &= \frac{\binom{n!}{n!x, n!x, \cdots, n!x, n!(1-\rho x)}}{\binom{n!}{n!x}^{\rho}} \\ &\sim_n \frac{\dfrac{(n!)^{n!}}{(n!x)^{n!x\rho}(n!(1-\rho x))^{n!(1-\rho x)}}}{\left(\dfrac{(n!)^{n!}}{(n!x)^{(n!x)}(n!(1-x))^{n!(1-x)}}\right)^{\rho}} \text{ [Applying Stirling's approximation]} \end{aligned}$$

$$\sim_n \quad \left(\frac{(1-x)^{(1-x)\,\rho}}{(1-\rho x)^{(1-\rho x)}} \right)^{n!}$$

It has been shown in [71] that if $x \in (0, \dfrac{1}{y}), y > 1, f(x) = y(1-x)\log(1-x) - (1-xy)\log(1-xy)$ then $\dfrac{df}{dx} = -y(\log(1-x)+1) + y(\log(1-xy)+1) = \log((\dfrac{1-xy}{1-x})^y) < 0$. That is f(x) is monotonically decreasing function of x, with $f(0) = 0$. By considering $y = \rho$ and $x = \frac{h(N_k)}{n!} \in (0, \frac{1}{\rho})$, we get

$$p_g \sim_n \left(\frac{(1-x)^{(1-x)\rho}}{(1-\rho x)^{(1-\rho x)}} \right)^{n!} = o(1) \implies p_g \leq_n p_{SS}. \tag{2.2}$$

Hence the proof.

2.4 Incremental Search Heuristic (ISH) Algorithm and its Parallel Version

SS starts with a random order S having span k and then generates and evaluates orders in a specific manner to improve the span. The process terminates after ρ failed attempts to improve the span. By evaluating an order, SS essentially evaluates $h(\rho N_k)$ orders. Note that SS starting with an order S having span k, can essentially evaluate at most $h((\Delta(G)+1-\chi(G))k!N_k)$ orders which can be $< n!$. Thus starting with order S with span k, SS may not reach the optimum order even for arbitrarily large ρ. This section proposes our incremental search heuristic (ISH) algorithm to hit the optimum with a positive probability. ISH essentially generates some random orders, calls SS for each of them, and finally reports the best span obtained. We also propose a parallel version of ISH, which is more time-efficient if multiple processors are present.

2.4.1 Incremental Search Heuristic (ISH)

ISH generates ρ_1 random orders and calls SS with $\rho = \rho_2$ for each of them and finally returns the color vector having the minimum span. ISH is formally presented in Algorithm 2. It is evident that ISH executes $h(\rho_1\rho_2)$ steps. Similar to SS the time and space complexities of ISH are $O(\rho_1\rho_2|K|(|V| + |E|))$ and $O(|V| + |E|)$ respectively.

Algorithm 2: *Incremental Search Heuristic (ISH)*

```
Input: $G$,$\rho_1$,$\rho_2$
Output: $C$
$C = (1, 2, \cdots, n)$;
for $i = 1, 2, \cdots, \rho_1$ do
    $C_i = SS(G, \rho_2)$;
    if $span(C_i) < span(C)$ then
        $C = C_i$;
return $C$;
```

2.4.2 Parallel Incremental Search Heuristic (PISH)

ISH could be interpreted as parallel programming. For this purpose, first, we build a parallel version of SS, called parallel selective search (PSS) algorithm, and present it in Algorithm 3. Using PSS, we build the parallel version of ISH, called parallel incremental search heuristic (PISH), and present it in Algorithm 4. In PISH, we essentially evaluate SS in a parallel manner with different orders. We fix a global color vector C_o and a global variable l_o which store the minimum span and the minimum number of steps to get C_o from different threads running PSS, respectively. It is evident that if multiple processors are present, we can hit the optimum more time-efficiently using PISH. Note that both PSS and PISH executes $h(\rho_2)$ steps. Time and space complexities of both PSS and PISH are $O(\rho_2|K|(|V| + |E|))$ and $O(|V| + |E|)$.

Algorithm 3: *Parallel Selective Search (PSS) Algorithm*

```
Input: $G$, $\rho$, Global $(C_o, l_o, \mu)$
Generate a random order $S$ of the vertices of $G$;
$x = 0$;
$C = \mathbb{F}(G, S)$;
$x = x + 1$;
$\mu.lock()$;
if $span(C) < span(C_o)$ then
    $C_o = C$;
    $l_o = x$;
else if $span(C) = span(C_o)$ and $l_o > x$ then
    $l_o = x$;
$\mu.unlock()$;
for $i = 1, 2, \cdots \rho$ do
    Let $\pi \in \Pi(S)$ and $S' \in L(\pi)$;
    $C' = \mathbb{F}(G, S')$;
    $x = x + 1$;
    if $span(C') < span(C)$ then
        $C = C'$;
        $\mu.lock()$;
        if $span(C) < span(C_o)$ then
            $C_o = C$;
            $l_o = x$;
        $\mu.unlock()$;
        Reset $i = 1$;
```

Algorithm 4: *Parallel Incremental Search Heuristic (PISH)*

```
Input: $G$,$\rho_1$,$\rho_2$
Output: $C_o$
Set $C_o = (1, 2, \cdots, |V(G)|)$;
Set $l_o = 0$;
Declare mutex $\mu$;
Do in parallel using $\rho_1$ threads
    $PSS(G, \rho_2, \text{Global}(C_o, l_o, \mu))$;
return $C_o$;
```

Remark 2 *It is evident that though PISH is a parallel version of ISH it still need the information of the whole graph to run. It is interesting to build a decentralized version of ISH and would be considered in future work.*

2.4.3 Analysis of ISH and PISH

Let $\phi(k)$ and $\phi'(k)$ be the expected number of steps to hit span k by ISH and PISH, respectively. Since PISH makes ρ_1 PSS calls in parallel manner, $\phi'(k) \leq \dfrac{\phi(k)}{\rho_1}$. So, in the rest of this section, we analyze ISH only. By replacing $\phi(k)$ with $\phi'(k)$, we get the corresponding results for PISH. Let K be the set of spans that could be produced by applying greedy coloring on all possible orders of vertices of G. Let $A(k)$ and $A(\leq k)$ be the set of orders with span k and $\leq k$ respectively. Let $p(k|k') = \dfrac{|A(k)|}{|A(\leq k')|}$ be the probability that a random order chosen from the set of orders with span $\leq k'$, hits span k. Let $p(k)$ be the probability that a random order, chosen from $n!$ orders, hits span k. Clearly, $p(k) = \dfrac{|A(k)|}{n!} = \dfrac{h(N_k)}{n!} = h(\dfrac{1}{k^n})$ and $p(k|k') = \dfrac{p(k)}{\sum\limits_{k'' \in K \text{ \& } k \leq k'' \leq k'} p(k'')}$. Let $\chi(G)$ be the chromatic number of G. Since Theorems 2.2.1 and 2.2.3 show that there are $h(\chi(G)!N_{\chi(G)})$ optimum orders, $p(\chi(G)) = h(\frac{\chi(G)!}{(\chi(G))^n})$.

Theorem 2.4.1 *The expected number of steps required by ISH to hit span k is* $\phi(k) = o(1)$.

Proof : It is evident that ISH calls SS with a random order with span k'. SS essentially partitions the set of orders $A(\leq k')$ into disjoint sets of $h(N_{k'})$ equivalent orders. It then chooses ρ random partitions and evaluates one order from each such partition. Thus SS hits span k from k' with probability $h(N_{k'}) \times \dfrac{|A(k)|}{|A(\leq k')|} = h(N_{k'} \times p(k|k'))$. Thus given $k' \geq k$, total number of steps required by SS to hit span k, is $\dfrac{1}{h(N_{k'} \times p(k|k'))}$. Again, the initial order of SS has span k' with probability $p(k')$. Thus the number of steps required by ISH to find span k is:

$$\begin{aligned}
\phi(k) &= \sum_{k' \in K \text{ \& } k' \geq k} p(k') \frac{1}{h(N_{k'} \times p(k|k'))} \\
&= \sum_{k' \in K \text{ \& } k' \geq k} \quad \sum_{k'' \in K \text{ \& } k \leq k'' \leq k'} p(k'') \frac{1}{n! p(k)} \leq \frac{n}{n! p(k)} = \frac{n}{h(N_k)}
\end{aligned}$$

$$= \begin{cases} o(1) & \text{If} k < \frac{n}{e} \\ o(n(\frac{ek}{n})^n) & \text{Otherwise.} \end{cases}$$

Since $p(\chi(G)) = h(\dfrac{\chi(G)!}{(\chi(G))^n})$, we are left to show $\phi(\chi(G)) = o(1)$ when $\chi(G) \geq \frac{n}{e}$. Note that if $\chi(G) \geq \frac{n}{e}$, $\chi(G) = \epsilon n$, for some $\epsilon \in [e^{-1}, 1]$. Hence $\phi(\chi(G)) = o(n\dfrac{(e\epsilon)^n}{(n\epsilon)!}) = o(n\dfrac{(e^{1+\epsilon}\epsilon)^n}{(n\epsilon)^{(n\epsilon)}}) = o(1)$. Hence the proof.
Since $\phi(\chi(G)) = o(1)$ and each step requires $O(|V| + |E|)$ time and space complexity, the expected time and space complexity of ISH is $O(|V| + |E|)$. Since each vertex and edge must be visited to color vertices of a graph, these complexities are asymptotically minimum among all randomized algorithms for this problem. Since NP-complete problems are inter-reducible in polynomial time, from Theorem 2.4.1 we can conclude: *For each problem in NP, there is a randomized algorithm that solves it in expected polynomial time.*

2.5 Simulation Results

In this section, we first simulate ISH and PISH on 136 well-known benchmark instances and compare with 10 states of the art algorithms, and then we verify our theoretical results on random graphs.

2.5.1 Simulation on benchmarks

We simulate ISH and PISH with $\rho_1 = \rho_2 = \rho = 10^6$ and compare with 10 state of the art algorithms on 136 challenging benchmarks taken from [72, 73, 74, 75, 76, 77, 78, 79, 80] and show the results in Table2.1. These benchmarks are well-known graph coloring problems, including geometric graphs which is the graphs can be generated as interference graphs in D2D communication. Here n represents the number of vertices, e represents the number of edges, and χ_u represents the best known upper bound of the chromatic number of the corresponding benchmarks. LAVCA represents a variable action-set learning automaton reported in [67]. DBG represents a hybrid genetic algorithm reported in [68]. MCOACOL represents a modified cuckoo optimization algorithm reported in [37]. Best of [81] represents the minimum span produced and its corresponding time among the five integer linear programming based algorithms REP, POP, POP2, ASS+(c), and ASS+(e) reported in [81]. DR represents a Doglas-Ranchford algorithm reported in [69]. EBDA represents an enhanced binary dragonfly algorithm reported in [70]. Note that these algorithms are considered by many authors [82, 83, 84, 85]. For a particular algorithm, χ and T represents smallest span produced and the time taken in seconds (s) (rounded up to 4-th decimal point) by corresponding algorithm to reach its corresponding value of χ. Here l represents the number of steps required to reach χ for both ISH and PISH. For ISH and PISH let $\sigma(T)$ be the standard deviation of time T taken by those algorithms. Since $\sigma(T)$ depends on T, we present $\frac{\sigma(T)}{T}$ for both ISH and PISH respectively. For each state of the art algorithm, we count and report the number of instances for which its span (χ) is *greater, equal and smaller* than that of ISH and PISH, in the format x/y, at the bottom of the table, where x and y represent the corresponding number of instances for ISH and PISH respectively. For each algorithm we also count and report the number of instances, for which span χ $\leq$ that of ISH and PISH, and time T is *greater, equal and smaller* than that of ISH and PISH. For both ISH and PISH, $\sigma(T) \leq T$, which is in accordance with the geometric distribution. We observe both ISH and PISH hit the optimum span for all benchmarks and PISH takes the minimum time among all algorithms. Moreover for a benchmark, if a state-of-the-art algorithm hits the best known upper bound, then most of the times, time taken by ISH is less than that of that algorithm. These indeed are reflections of theoretical findings stated in Theorem 2.4.1.

Benchmark				LAVCA[67]		DBG[68]		MCOACOL[37]		Best of [81]		DR[69]		EBDA[70]		ISH				PISH			
Instances	n	e	χ_u	χ	T	χ	T	χ	T	χ	T	χ	T	χ	T	χ	l	T	$\frac{\sigma(T)}{T}$	χ	l	T	$\frac{\sigma(T)}{T}$
1-FullIns-3	30	100	4	–	–	–	–	4	0.4	–	–	–	–	–	–	4	1	0.0001	0.0	4	1	0.0001	0.0
1-FullIns-4	93	593	5	–	–	–	–	–	–	–	–	–	–	–	–	5	1	0.0014	0.0	5	1	0.0012	0.0
1-FullIns-5	282	3247	6	–	–	–	–	6	1.9	6	1.54	–	–	–	–	6	3	0.0016	0.02	6	1	0.0005	0.0
1-Insertions-4	67	232	5	–	–	–	–	–	–	–	–	–	–	5	2.63	5	1	0.0002	0.01	5	1	0.0002	0.01
1-Insertions-5	202	1227	6	–	–	–	–	6	1.2	–	–	–	–	–	–	6	1	0.0003	0.0	6	1	0.0003	0.01
1-Insertions-6	607	6337	7	–	–	–	–	7	8.1	–	–	–	–	–	–	7	2	0.0024	0.01	7	1	0.0011	0.01
2-FullIns-3	52	201	5	–	–	–	–	5	0.4	–	–	–	–	–	–	5	1	0.0008	0.0	5	1	0.0007	0.01
2-FullIns-4	212	1621	6	–	–	–	–	6	1.2	6	0.01	–	–	–	–	6	2	0.0005	0.01	6	1	0.0002	0.0
2-FullIns-5	852	12201	7	–	–	–	–	7	10.7	7	5.02	–	–	–	–	7	4	0.003	0.0	7	2	0.0014	0.0
2-Insertions-3	37	72	4	–	–	4	2.23	4	0.4	–	–	–	–	4	2.08	4	1	0.0001	0.0	4	1	0.0001	0.0
2-Insertions-4	149	541	5	–	–	–	–	5	1.1	–	–	–	–	5	5.71	5	1	0.0001	0.0	5	1	0.0001	0.0
2-Insertions-5	597	3936	6	–	–	–	–	6	6.5	–	–	–	–	–	–	6	2	0.0027	0.02	6	1	0.0012	0.01
3-FullIns-3	80	346	6	–	–	–	–	6	0.4	–	–	–	–	–	–	6	1	0.0009	0.01	6	1	0.0008	0.01
3-FullIns-4	405	3524	7	–	–	–	–	7	1.6	7	0.02	–	–	–	–	7	2	0.0011	0.01	7	1	0.0005	0.0
3-FullIns-5	2030	33751	8	–	–	–	–	8	30.5	–	–	–	–	–	–	8	3	0.0375	0.05	8	1	0.0113	0.03
3-Insertions-3	56	110	4	–	–	4	3.37	4	0.5	–	–	–	–	4	3.49	4	1	0.0001	0.0	4	1	0.0001	0.0
3-Insertions-4	281	1046	5	–	–	–	–	5	2.1	–	–	–	–	–	–	5	1	0.0006	0.0	5	1	0.0006	0.0
3-Insertions-5	1406	9695	6	–	–	–	–	6	45	–	–	–	–	–	–	6	3	0.0097	0.0	6	1	0.0029	0.0
4-FullIns-3	114	541	7	–	–	–	–	7	0.7	–	–	–	–	–	–	7	1	0.0012	0.01	7	1	0.0011	0.01
4-FullIns-4	690	6650	8	–	–	–	–	8	7.7	8	0.02	–	–	–	–	8	2	0.003	0.02	8	1	0.0014	0.02
4-FullIns-5	4146	77305	9	–	–	–	–	9	147.5	–	–	–	–	–	–	9	1	0.1013	0.09	9	1	0.0912	0.14
4-Insertions-3	79	156	4	–	–	–	–	4	0.6	–	–	–	–	4	3.72	4	1	0.0001	0.0	4	1	0.0001	0.0
4-Insertions-4	475	1795	5	–	–	–	–	5	3.7	–	–	–	–	–	–	5	2	0.0027	0.0	5	1	0.0012	0.0
5-FullIns-3	154	792	8	–	–	–	–	8	0.5	–	–	–	–	–	–	8	1	0.01	0.02	8	1	0.009	0.01
5-FullIns-4	1085	11395	9	–	–	–	–	9	28	9	0.04	–	–	–	–	9	2	0.0053	0.0	9	1	0.0024	0.01
abb313GPIA	1557	53356	12	–	–	–	–	12	224	–	–	–	–	–	–	12	46329	251	0.43	12	1	0.0049	0.02
anna	138	493	11	–	–	11	11.63	11	0.8	–	–	11	1.04	11	2.53	11	1	0.0001	0.0	11	1	0.0001	0.0
ash331GPIA	662	4181	4	–	–	–	–	5	42.2	4	3.29	–	–	–	–	4	530	8	0.27	4	2	0.0272	0.01
ash608GPIA	1216	7844	4	–	–	–	–	4	0.5	–	–	–	–	–	–	4	53	1.023	0.01	4	2	0.0347	0.08
ash958GPIA	1916	12506	5	–	–	–	–	5	471.6	–	–	–	–	–	–	5	79	1.3	0.09	5	3	0.0444	0.05
C2000.5	2000	999836	151	–	–	151	8421	–	–	–	–	–	–	–	–	151	5120201	75241	0.36	151	5120	67.7142	0.04
C2000.9	2000	1799532	411	–	–	411	7368	–	–	–	–	–	–	–	–	411	1065727	14936	0.44	411	85	1.0721	0.1
C4000.5	4000	4000268	282	–	–	282	212881	–	–	–	–	–	–	–	–	282	1222327	77721	0.41	282	1222	69.9302	0.37
david	87	406	11	–	–	11	10.89	11	0.5	–	–	11	0.26	11	2.61	11	1	0.0001	0.0	11	1	0.0001	0.0
DSJC1000.1	1000	49629	20	20	43.107	20	4982	–	–	–	–	–	–	20	3975	20	692353	3448	0.2	20	692	3.1016	0.24
DSJC1000.5	1000	249826	83	84	205.1	83	2164	–	–	–	–	–	–	83	2118	83	602275	4276	0.04	83	602	3.8466	0.36
DSJC1000.9	1000	449449	223	224	423	224	8092	–	–	–	–	–	–	223	5789	223	50599	339	0.04	223	51	0.3075	0.22
DSJC125.1	125	736	5	5	0.009	6	13.12	–	–	–	–	–	–	5	4.02	5	1075	0.4643	0.0	5	1	0.0004	0.0
DSJC125.5	125	3891	17	17	17.04	17	19.71	–	–	20	3600	–	–	17	13.48	17	3909	15.4665	0.21	17	2	0.0071	0.04
DSJC125.9	125	6961	44	44	41.17	44	35.87	–	–	44	3600	–	–	–	–	44	696	3.0325	0.33	44	2	0.0078	0.0
DSJC250.1	250	3218	8	8	16.015	8	26.07	8	3	–	–	–	–	8	5.77	8	111536	4921	0.08	8	101	2.7	0.25
DSJC250.5	250	15668	28	28	42.01	28	89	33	600	–	–	–	–	30	80.12	28	13067	2312	0.39	28	301	37.809	0.42
DSJC250.9	250	27897	72	72	67.3	72	75.81	–	–	–	–	–	–	72	63.28	72	1765	112	0.45	72	3	0.1713	0.12
DSJC500.1	500	12458	12	12	32.2	12	112	–	–	–	–	–	–	12	110.34	12	1939	19.4	0.3	12	1	0.009	0.02
DSJC500.5	500	62624	48	48	106	48	147	–	–	–	–	–	–	48	147.3	48	7398719	9594	0.03	48	16	0.0187	0.0
DSJC500.9	500	112437	126	127	114.5	126	190	–	–	–	–	–	–	126	173.18	126	40583	73	0.13	126	1	0.0016	0.02
DSJR500.1	500	3555	12	–	–	–	–	12	6.9	–	–	–	–	12	140.27	12	11	0.015	0.01	12	3	0.0037	0.01
DSJR500.1c	500	121275	85	–	–	85	581	–	–	85	0.33	–	–	–	–	85	229	0.0335	0.06	85	2	0.0003	0.0
DSJR500.5	500	58862	122	–	–	124	728	–	–	122	572.01	–	–	122	587	122	15145	2545	0.16	122	11	1.6636	0.02
flat1000-50	1000	245000	50	–	–	50	802	–	–	–	–	–	–	50	648	50	196762	985	0.15	50	8	0.036	0.05
flat1000-60	1000	245830	60	–	–	60	1344	–	–	–	–	–	–	60	1197	60	275447	1934	0.46	60	18	0.1137	0.04
flat1000-76	1000	246708	82	–	–	82	3795	–	–	–	–	–	–	–	–	82	119581	748	0.23	82	120	0.6756	0.08
flat300-20	300	21375	20	–	–	20	4982	–	–	–	–	–	–	–	–	20	411	0.5434	0.02	20	1	0.0012	0.01

flat300-26	300	21633	26	–	–	26	275	–	–	–	–	–	–	26	272	26	35153	73	0.1	26	35	0.0654	0.08
flat300-28	300	21695	28	–	–	28	319	–	–	–	–	–	–	28	348	28	102431	171	0.21	28	1029	1.546	0.14
fpsol2.i.1	496	11654	65	–	–	65	82.03	65	3.6	65	82.03	65	463.94	–	–	65	1	0.0011	0.0	65	1	0.001	0.02
fpsol2.i.2	451	8691	30	–	–	30	69.79	30	7.4	–	–	30	495.94	–	–	30	1	0.001	0.01	30	1	0.0009	0.01
fpsol2.i.3	425	8688	30	–	–	30	67.14	30	6.4	–	–	30	480.27	–	–	30	1	0.0015	0.01	30	1	0.0013	0.01
games120	120	638	9	–	–	9	10.77	9	0.8	–	–	9	0.24	9	3.24	9	1	0.0001	0.0	9	1	0.0001	0.0
homer	561	1628	13	–	–	–	–	13	4.3	–	–	13	59.01	13	6.67	13	1	0.002	0.01	13	1	0.0018	0.0
huck	74	301	11	–	–	11	11.1	11	0.5	–	–	11	0.11	11	4.78	11	1	0.0001	0.0	11	1	0.0001	0.0
inithx.i.1	864	18707	54	–	–	–	–	54	19.4	–	–	54	2443.43	54	93.14	54	1	0.0048	0.01	54	1	0.0043	0.01
inithx.i.2	645	13979	31	–	–	–	–	31	9	–	–	31	1500.45	31	39.16	31	1	0.0034	0.02	31	1	0.003	0.01
inithx.i.3	621	13969	31	–	–	–	–	31	6	–	–	31	1432.43	31	42.71	31	1	0.0017	0.01	31	1	0.0015	0.01
jean	80	254	10	–	–	10	9.92	10	0.5	–	–	10	0.13	10	4.42	10	1	0.0001	0.0	10	1	0.0001	0.0
latin-square-10	900	307350	99	–	–	99	1267	–	–	–	–	–	–	99	813	99	1305079	7635	0.38	99	13	0.0684	0.06
le450-15a	450	8168	15	15	31.29	–	–	–	–	–	–	15	1944.35	15	2.63	15	1292373	830	0.3	15	1	0.0006	0.01
le450-15b	450	8169	15	15	40.14	–	–	–	–	15	700.5	15	2076.54	15	4.23	15	18519	1497	0.49	15	31	2.2553	0.26
le450-15c	450	16680	15	15	77.12	15	93	–	–	25	3600	15	173.1	–	–	15	1785	7824	0.42	15	19	69.408	0.46
le450-15d	450	16750	15	15	59.15	15	228	–	–	26	3600	15	619.74	–	–	15	124273	13055	0.25	15	43	4.0655	0.36
le450-25a	450	8260	25	–	–	–	–	25	4.4	–	–	25	68.93	25	3.65	25	156	0.001	0.01	25	1	0.0001	0.0
le450-25b	450	8263	25	–	–	–	–	25	7	–	–	25	65.82	25	5.58	25	34	0.0014	0.01	25	1	0.0001	0.0
le450-25c	450	17343	25	25	74.2	25	740	–	–	30	3600	–	$infty$	–	–	25	16848	9298	0.39	25	26	12.9139	0.37
le450-25d	450	17425	25	25	80.103	25	382	–	–	30	3600	–	$infty$	–	–	25	13713	8051	0.18	25	26	13.7383	0.11
le450-5a	450	5714	5	–	–	–	–	–	–	5	21.17	5	82.47	–	–	5	2938	26.6782	0.21	5	2	0.0163	0.01
le450-5b	450	5734	5	–	–	–	–	–	–	5	140.16	5	238.33	–	–	5	2418	28	0.39	5	2	0.0208	0.06
le450-5c	450	9803	5	–	–	–	–	6	279	–	–	5	68.68	–	–	5	819	4.1101	0.17	5	2	0.009	0.02
le450-5d	450	9757	5	–	–	–	–	7	80.8	–	–	5	44.49	–	–	5	750	4.1442	0.18	5	2	0.0099	0.01
miles1000	128	3216	42	–	–	–	–	42	1	–	–	42	2.43	42	5.73	42	12	0.0037	0.01	42	1	0.0003	0.0
miles1500	128	5198	73	–	–	–	–	73	1.2	–	–	73	24.65	73	9.34	73	2	0.0005	0.01	73	1	0.0002	0.0
miles250	128	387	8	–	–	8	4.39	8	1.1	–	–	8	0.4	8	4.52	8	5	0.0004	0.0	8	1	0.0001	0.0
miles500	128	1170	20	–	–	20	14.48	20	1.2	–	–	20	1.07	20	5.11	20	7	0.0006	0.01	20	1	0.0001	0.0
miles750	128	2113	31	–	–	–	–	31	1.5	–	–	31	2.54	31	5.64	31	22	0.0085	0.01	31	1	0.0003	0.0
mug100-1	100	166	4	–	–	4	8.34	4	0.8	4	0.13	4	0.07	4	2.9	4	1	0.0001	0.0	4	1	0.0001	0.0
mug100-25	100	166	4	–	–	4	8.21	4	0.5	4	0.31	4	0.06	4	2.57	4	1	0.0003	0.0	4	1	0.0002	0.0
mug88-1	88	146	4	–	–	4	4.05	4	1.1	–	–	4	0.05	4	2.46	4	1	0.0001	0.0	4	1	0.0001	0.0
mug88-25	88	146	4	–	–	4	3.67	4	1.3	–	–	4	0.05	4	1.65	4	1	0.0002	0.0	4	1	0.0002	0.01
mulsol.i.1	197	3925	49	–	–	49	25.75	49	1.2	–	–	49	18.79	49	9.68	49	1	0.0009	0.01	49	104	0.0805	0.1
mulsol.i.2	188	3885	31	–	–	31	22.07	31	1.1	–	–	31	63.18	31	7.43	31	1	0.0003	0.0	31	1	0.0002	0.0
mulsol.i.3	184	3916	31	–	–	31	23.49	31	1.3	–	–	31	55.88	31	6.49	31	1	0.0006	0.0	31	1	0.0005	0.0
mulsol.i.4	185	3946	31	–	–	31	25.22	31	1.7	–	–	31	60.71	31	6.75	31	1	0.0002	0.0	31	1	0.0001	0.0
mulsol.i.5	186	3973	31	–	–	31	28.33	31	1.9	–	–	31	62.72	–	–	31	1	0.0045	0.01	31	1	0.0041	0.01
myciel3	11	20	4	–	–	4	0.01	4	0.2	–	–	–	–	4	0.48	4	1	0.0003	0.0	4	1	0.0003	0.01
myciel4	23	71	5	–	–	5	0.89	5	0.3	–	–	–	–	5	0.82	5	1	0.0031	0.02	5	1	0.0028	0.01
myciel5	47	236	5	–	–	5	5.51	6	0.3	–	–	–	–	6	1.97	5	1	0.005	0.01	5	1	0.0045	0.03
myciel6	95	755	7	–	–	7	12.77	7	0.5	–	–	–	–	7	1.72	7	1	0.015	0.02	7	1	0.0135	0.02
myciel7	191	2360	8	–	–	8	20.19	8	3.8	–	–	–	–	8	4.36	8	2	0.0008	0.0	8	1	0.0004	0.01
qg.order30	900	26100	30	–	–	–	–	30	326.4	–	–	–	–	–	–	30	20689	104	0.23	30	1	0.0045	0.03
qg.order40	1600	62400	40	–	–	–	–	41	746.3	40	534.83	–	–	–	–	40	1251	1.2772	0.12	40	1	0.0009	0.0
qg.order60	3600	212400	62	–	–	–	–	–	–	62	3600	–	–	–	–	62	573461	14554	0.11	62	2	0.0457	0.11
queen10-10	100	2940	12	–	–	–	–	12	9.6	12	3600	–	–	–	–	12	8297	163	0.06	12	2	0.0354	0.01
queen11-11	121	3960	13	–	–	–	–	14	2.4	13	3600	–	–	–	–	13	390913	5136	0.37	13	217	2.16	0.21
queen12-12	144	5192	15	–	–	–	–	15	2.8	–	–	–	–	–	–	15	7769	227	0.13	15	3	0.0789	0.05
queen13-13	169	6656	16	–	–	–	–	16	9.3	–	–	–	–	–	–	16	107311	1397	0.43	16	200	2.3433	0.36
queen14-14	196	4186	17	–	–	–	–	17	22.2	–	–	–	–	–	–	17	105369	2037	0.44	17	407	7.0813	0.17
queen15-15	225	5180	18	–	–	–	–	18	17.9	–	–	–	–	–	–	18	1863427	39645	0.4	18	458	8.7697	0.43
queen16-16	256	12640	19	–	–	–	–	19	53.6	–	–	–	–	–	–	19	1694437	5130	0.05	19	100	0.2725	0.1
queen5-5	25	320	5	–	–	5	1.2	5	0.3	–	–	–	–	5	2.01	5	1	0.001	0.01	5	1	0.0009	0.01
queen6-6	36	580	7	–	–	7	1.32	8	5.7	–	–	–	–	7	2.47	7	126	0.0135	0.05	7	1	0.0001	0.0

queen7-7	49	952	7	–	–	7	6.91	7	30.5	–	–	–	–	7	6.17	7	1470	0.068	0.06	7	3	0.0001	0.0
queen8-12	96	1368	13	–	–	–	–	13	1.1	–	–	–	–	–	–	13	567	0.2885	0.07	13	13	0.006	0.04
queen8-8	64	728	9	–	–	9	9.87	10	1.4	–	–	–	–	10	10.24	9	549	2.942	0.28	9	2	0.0096	0.01
queen9-9	81	1056	10	–	–	10	13.96	11	94	–	–	–	–	–	–	10	743	8.6337	0.12	10	2	0.0209	0.02
r1000.1	1000	14378	20	–	–	20	4982	–	–	–	–	–	–	20	644	20	191	0.2837	0.07	20	1	0.0013	0.01
r1000.1c	1000	485090	98	–	–	98	1329	–	–	–	–	–	–	–	–	98	650	4.758	0.44	98	1	0.0066	0.01
r1000.5	1000	238267	242	–	–	242	1507	–	–	–	–	–	–	242	1422	242	48660	307	0.4	242	49	0.2782	0.01
r125.1	125	209	5	–	–	–	–	5	1.2	–	–	–	–	5	4.02	5	2	0.0001	0.0	5	1	0.0001	0.0
r125.1c	125	7501	46	–	–	–	–	46	1.8	–	–	–	–	46	4.45	46	6	0.0004	0.01	46	1	0.0001	0.0
r125.5	125	3838	37	–	–	–	–	37	6.7	–	–	–	–	37	13.48	37	900	0.2782	0.08	37	2	0.0006	0.0
r250.1	250	867	8	–	–	8	308	8	2.4	–	–	–	–	8	264	8	3	0.001	0.01	8	1	0.0003	0.0
r250.1c	250	30227	64	–	–	64	266	64	5	–	–	–	–	64	189	64	22	0.0074	0.04	64	1	0.0003	0.0
r250.5	250	14849	65	–	–	65	261	–	–	–	–	–	–	–	–	65	67475	25.781	0.25	65	67	0.023	0.06
school1	385	19095	14	–	–	–	–	14	1934.2	–	–	–	–	–	–	14	597	0.491	0.27	14	1	0.0007	0.01
school1-nsh	352	14612	14	–	–	–	–	14	1352.6	14	12.76	–	–	–	–	14	773	0.3238	0.09	14	2	0.0008	0.0
wap01a	2368	110871	42	42	16.186	–	–	–	–	–	–	–	–	–	–	42	93919	1151	0.06	42	11	0.1213	0.04
wap02a	2464	111742	41	41	32.057	–	–	–	–	–	–	–	–	–	–	41	112393	1168	0.21	41	17	0.159	0.1
wap03a	4730	286722	44	44	90.321	–	–	–	–	–	–	–	–	–	–	44	136011	63957	0.33	44	10	4.2321	0.19
wap04a	5231	294902	43	43	68.25	–	–	–	–	–	–	–	–	–	–	43	2153473	132671	0.05	43	5	0.2772	0.25
wap05a	905	43081	50	–	–	–	–	50	12.4	50	125.45	–	–	–	–	50	11	0.0125	0.0	50	1	0.001	0.0
wap06a	947	43571	41	41	3.23	–	–	–	–	–	–	–	–	–	–	41	792708	3335	0.25	41	5	0.0189	0.0
wap07a	1809	103368	42	42	20.195	–	–	–	–	–	–	–	–	–	–	42	2419467	18797	0.34	42	10	0.0699	0.05
wap08a	1870	104176	42	42	28.302	–	–	–	–	–	–	–	–	–	–	42	96102768	770191	0.47	42	15	0.1082	0.1
will199GPIA	701	6772	7	–	–	–	–	8	21.8	7	6.68	–	–	–	–	7	460	0.9154	0.25	7	2	0.0036	0.01
zeroin.i.1	211	4100	49	–	–	49	28.24	49	1.7	–	–	49	27.1	49	7.74	49	1	0.0011	0.01	49	1	0.001	0.01
zeroin.i.2	211	3541	30	–	–	30	83.39	30	2.1	–	–	30	39.08	30	15.11	30	1	0.0011	0.01	30	1	0.001	0.02
zeroin.i.3	206	3540	30	–	–	30	27.06	30	1.6	–	–	30	34.51	30	9.87	30	1	0.0009	0.01	30	1	0.0008	0.01
Greater				3/3	3/22	3/3	51/65	11/11	72/81	5/5	17/23	0/0	37/39	3/3	54/65								
Equals				22/22	0/0	65/65	0/0	81/81	0/0	23/23	0/0	39/39	0/0	65/65	0/0								
Smaller				0/0	19/0	0/0	14/0	0/0	9/0	0/0	6/0	0/0	2/0	0/0	11/0								

Table 2.1: Span and time requirements of different algorithms on benchmarks

The summary of Table 2.1 is presented in Figure 2.3. Here algorithms LAVCA ([67]), DBG ([68]), MCOACOL ([37]), Best of [81], DR ([69]) and EBDA ([70]) are represented in x-axis as $a_1, a_2, \cdots, a_6$ respectively. It is evident from Table 2.1, that ISH hits the best known upper bound for each of the 136 benchmarks. On the other hand, each other algorithm a_i considered only a subset of them. In y-axis for each algorithm a_i, we plot total number of benchmarks considered by algorithm a_i:

1. Where ISH hits the best known upper bound (i.e., $\chi(ISH) = \chi_u$). (Since ISH hits the best known upper bound for each of the 136 benchmarks, this number essentially represents the total number of benchmarks considered by a_i);
2. Where a_i hits the best known upper bound (i.e., $\chi(a_i) = \chi_u$);
3. Where a_i hits the best known upper bound (i.e., $\chi(a_i) = \chi_u$) and time taken by a_i is less than that of ISH (i.e., $T(a_i) \leq T(ISH)$);

respectively. It is observed from Figure 2.3 that,

1. For each algorithm $a_i \in \{a_1, a_2, \cdots, a_6\}$, ISH produces the best known upper bound for all the benchmarks considered by algorithm a_i.
2. For each algorithm $a_i \in \{a_2, a_3, \cdots, a_6\}$, only for a small fraction of the benchmarks considered by a_i, ISH takes more time than a_i to produce the best known upper bound.

Hence ISH hits the best known upper bound for each of the 136 benchmarks, and in most of the cases, it takes lesser time than the other algorithms.

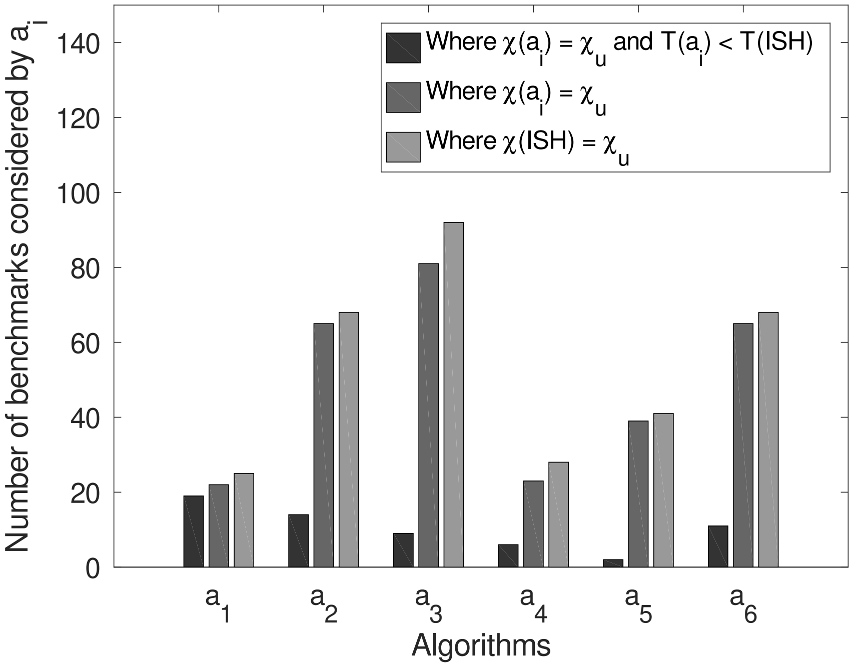

Figure 2.3: Summary of Table 2.1

2.5.2 Simulation on random graph

In this subsection, we simulate ISH on random graphs $G(n, p)$. We run ISH with $\rho_1 = \rho_2 = \rho$ for different values of n, p and ρ, and compare with greedy coloring. For each value of n, p and ρ we simulate ISH and greedy coloring over 10^4 random graphs and report the average result obtained from them.

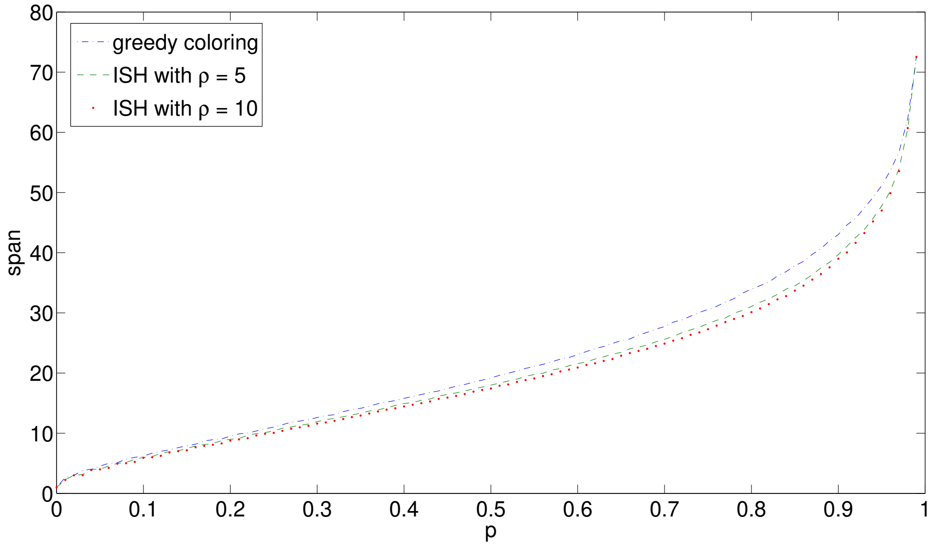

Figure 2.4: ISH vs greedy algorithm with varying p

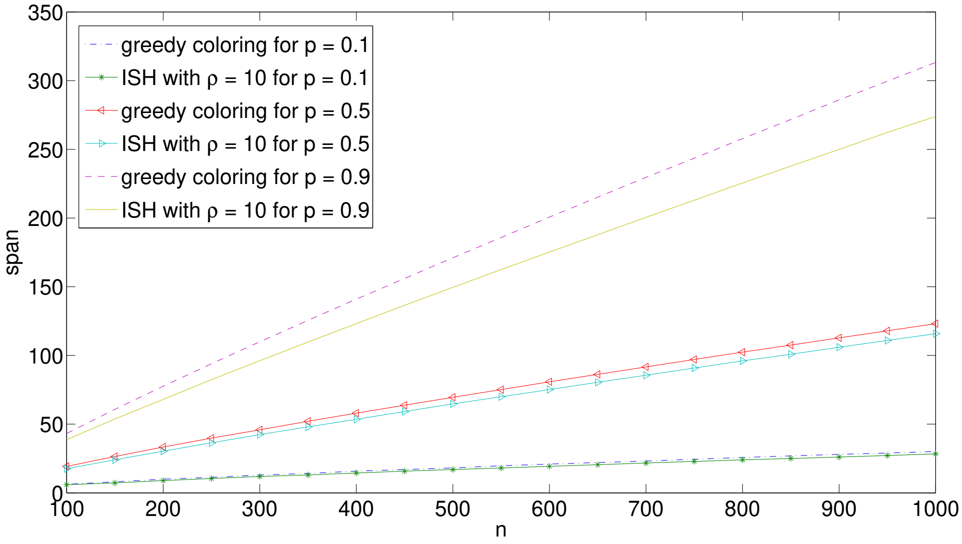

Figure 2.5: ISH vs greedy algorithm with varying n

Figure 2.4 shows the resulted spans produced by ISH and greedy coloring on $G(n, p)$ for varying p and fixed n. We fix $n = 100$ and vary p from 0 to 1 with an increment of 0.01. We plot the spans produced by ISH with $\rho = 5$ and $\rho = 10$. Recall that ISH evaluates $h(\rho^2)$ steps. Hence, we evaluate greedy coloring on 100 random orders and report the best span for a fair comparison. We observe that the span produced by the greedy algorithm is higher than the span produced by the ISH for both values of ρ. We also observe that with higher ρ, ISH produces a lesser expected span.

Figure 2.5 shows the resulted spans produced by ISH and greedy coloring on $G(n, p)$ for varying n and fixed p. We fix $p = 0.1, 0.5$ and 0.9 and vary n from 100 to 1000 with an increment of 50. Here we fix $\rho = 10$ for ISH. Again, we evaluate greedy coloring on 100 random orders and report the best span for a fair comparison. It can be observed that the performance of ISH is better than the greedy coloring. Moreover, with increasing n and p, spans produced by both ISH and greedy coloring increases.

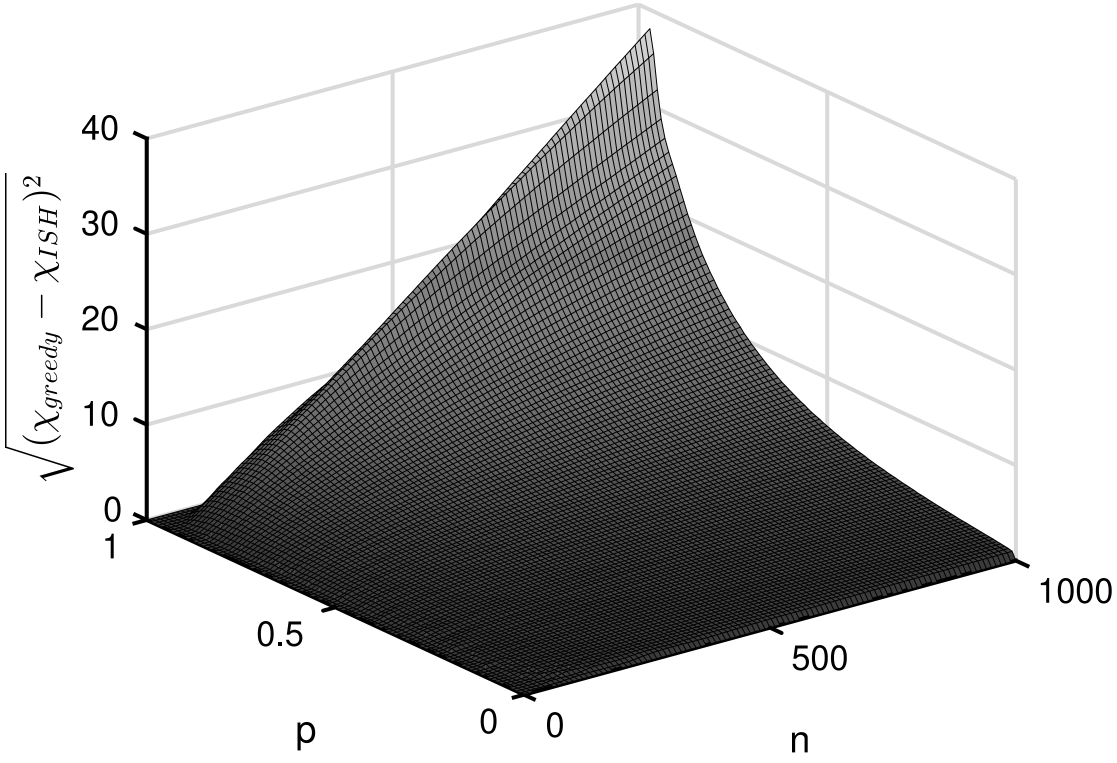

Figure 2.6: $\sqrt{(\chi_{greedy} - \chi_{ISH})^2}$ with varying n and p

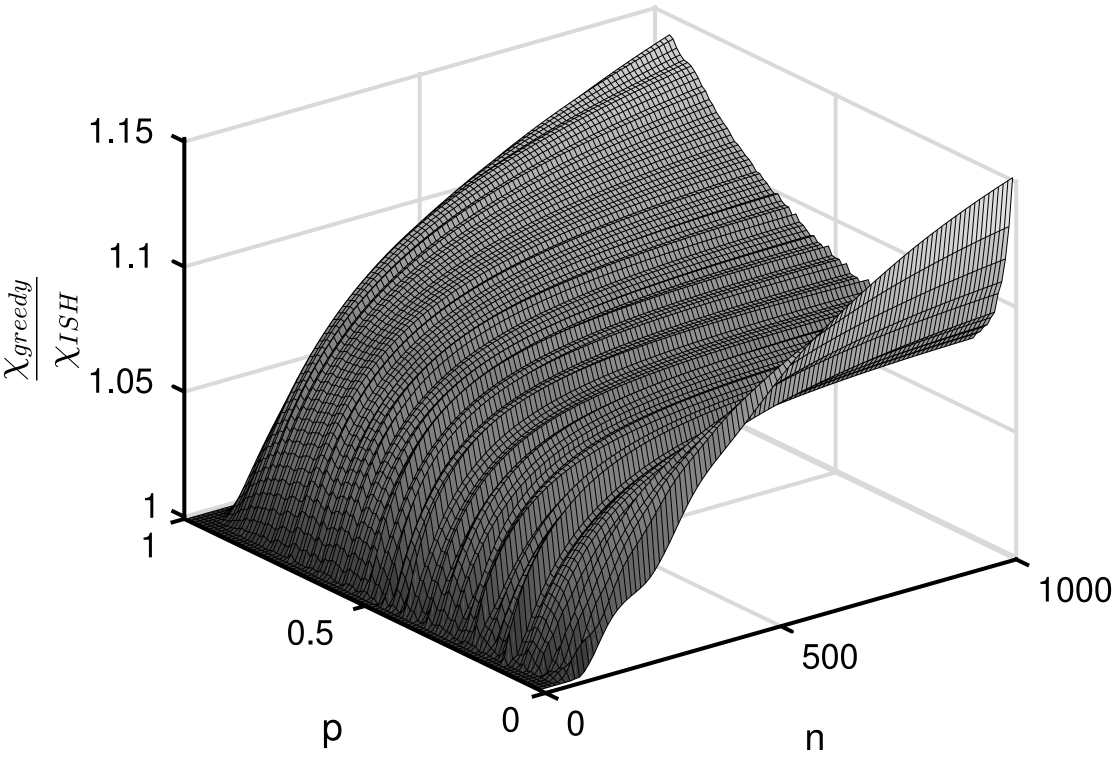

Figure 2.7: $\frac{\chi_{greedy}}{\chi_{ISH}}$ with varying n and p

Let χ_{ISH} and χ_{greedy} be the spans produced by ISH with $\rho = 10$ and greedy coloring on 100 random orders, respectively. We vary n from 0 to 1000 in an interval of 10 and p from 0 to 1 in an interval of 0.01 and plot $\sqrt{(\chi_{greedy} - \chi_{ISH})^2}$ and $\frac{\chi_{greedy}}{\chi_{ISH}}$ in Figures 2.6 and 2.7 respectively. We observe that, for a fixed p, $\sqrt{(\chi_{greedy} - \chi_{ISH})^2}$ increases with n. We also observe that, for a fixed n, with increasing p, $\sqrt{(\chi_{greedy} - \chi_{ISH})^2}$ initially increases from 0 and then decreases down to 0. This is because when $p \to 0$ then the graph is highly sparse and hence both greedy coloring and ISH can reach the optimum. Similarly when $p \to 1$, the graph is highly dense and hence both of the above mentioned algorithms perform the same. In rest of the cases ISH performs better than greedy coloring. We observe from Figure 2.7 that $\frac{\chi_{greedy}}{\chi_{ISH}} \in [1, 2]$. This statistically shows that ISH performs better than greedy coloring on random graphs. It is known from [86] that greedy coloring on random graphs produces exactly 2-

factor of the expected optimum span χ_{opt}. Hence $\frac{\chi_{ISH}}{\chi_{opt}} = \frac{1}{\frac{\chi_{greedy}}{\chi_{ISH}}} \times \frac{\chi_{greedy}}{\chi_{opt}} = \frac{2}{\frac{\chi_{greedy}}{\chi_{ISH}}}$. This statistical result implies that ISH produces $\leq$ 2-factor of the expected optimum span.

2.6 Conclusion

In this chapter, we have reduced the D2D channel assignment problem for minimizing $Y(t)$ alone a graph coloring problem. We have proposed an incremental search heuristic (ISH) and its parallel version PISH for coloring graphs. The salient feature of ISH is that it solves the graph coloring problem in expected polynomial time. We simulate ISH on 136 challenging benchmarks and found that it performs significantly better than 10 state of the art algorithms. We also have validated our theoretical findings on random graphs. It is evident that ISH essentially partitions the set of orders into different sets of equivalent orders and then chooses only one order from an equivalent set. This idea can be applied to develop orders based heuristic for other problems like minimum vertex cover, maximum independent set and minimum sum coloring in expected polynomial time.

Chapter 3

Minimizing the total perturbation

3.1 Introduction

In chapter 2 we minimized maximum channel required to keep every link active at time t. In this Chapter we essentially capture the effect of mobility of users on the resource allocation problem. We consider the D2D overlaid scenario, where a set of channels $\{1, 2, \cdots, Y_{\max}\}$ is reserved for D2D links (DLs). Due to the mobility of users, same channel might not be allocated to same link in consecutive time instances to avoid interference. Hence several channel switches/ color changes/ perturbations may occur. This leads to switching overhead which in turn may increase delay and packet loss. In this chapter, we address the problem of minimization of number of perturbations explicitly.

It is evident that each D2D link i requires a channel $c_i(t)$ to communicate at time t. Since channels are costly we have to allocate the same channel to multiple links. Since two users forming a D2D link reside in close proximity and the geometric region is large, a link can be represented by the middle point joining the endpoints of a link. Since each user is moving and we assume that each link remains active in the time of consideration, that middle point also moves with a velocity $\leq \mathbf{v}$. A pair of links resides within some r distance (interference range) apart can interfere with each other if they use the same channel. Thus at time t, the interference relationship of the active DLs can be modelled as an interference graph $g(t) = (V(t), E(t))$, where each DL is considered as a vertex $v \in V(t)$ and a pair of vertices forms an edge if their representing DLs are interfering to each other. Consider that some n links are placed within a large geometric region and they are active for a long time. Now from law of large number and central limit theorem we can say that the average number of DLs per unit area converges to a finite constant. Due to mobility of users, $g(t)$ varies over time, but $V(t)$ remains constant, as links are active for a sufficiently long time as per our assumption. Thus $g(t) = G(n, \lambda)$ can be considered as a sparse random graph with n vertices where each edge is generated with probability $\frac{\lambda}{n}$ and $\lambda = O(1)$.

Since $g(t)$ varies over time, a channel assigned to a link at time $(t-1)$ may not remain as an interference-free channel anymore at time t and hence we need to do channel switching which involves switching delay and degrades quality of service (QoS). To maintain QoS, we have to minimize the total number of channel switches or perturbations $A(t) = \sum_i I_i(C(t), C(t-1))$, where $I_i(C(t), C(t-1))$ is an indicator variable which indicates 1 if $c_i(t) \neq c_i(t-1)$, 0 otherwise. This chapter deals with the perturbation minimization problem, the objective of which is to find $C(t)$ given $g(t)$ and $C(t-1)$ such that $A(t)$ is minimized. It is evident that if we copy $C(t-1)$ to $C(t)$ some monochromatic edges may appear in $g(t)$. Graph induced by those monochromatic edges is termed as conflict graph $g_c(t)$. Since to resolve each monochromatic edge we have to recolor at least one endpoint of it, to resolve all edges in $g_c(t)$ we have to recolor the minimum vertex cover $V_c(t)$ of it. It is well-known that finding a minimum vertex cover of a graph is an NP-complete [11] problem. Note that we not only have to find a minimum vertex cover of $g_c(t)$ but also recolor the vertices in it. This makes the perturbation minimization problem more challenging.

In this chapter we essentially solve the problem of generating $C(t)$ given $g(t)$ and

$C(t-1)$ such that $A(t)$ is minimized. It is evident that if we can afford to allocate different channel to each link then no perturbations will ever be needed but in that case, $Y(t)$ will be large. On the other hand, if we fix $Y(t)$ at the chromatic number of $g(t)$, then perturbations will be huge. This implies that $Y(t)$ and $A(t)$ have natural trade-offs among them. In [59] an incremental coloring technique (IC) to minimize $A(t)$ is proposed. IC copies $C(t-1)$ to $C(t)$ and then traverse the vertices following an order. While visiting vertex v it checks whether v is an endpoint of a monochromatic edge. If yes, it recolors v using first-fit. First-fit essentially puts the minimum color to a vertex that is absent in all of its neighboring vertices. However, there is no guarantee that $A(t)$ produced by IC is $\leq \epsilon |V_c(t)|$ for any $\epsilon > 1$. To give such guarantee, we propose a differential coloring (DC) technique. DC first copies $C(t-1)$ to $C(t)$ and then builds $g_c(t)$. Next it applies a well-known 2-approximation minimum vertex cover algorithm to find the minimum vertex cover, removes the color of those vertices and then recolors them using first-fit. This implies that DC indeed provides a guarantee of producing $A(t)$ within 2-factor of its optimum value. Note that DC always applies first-fit for recoloring purpose. So it is efficient if number of channels are very small (indeed $O(\lambda)$). However, by allowing more channels to be allocated, we should be able to reduce $A(t)$ further. Owing to the trade-off between $Y(t)$ and $A(t)$, we propose a random coloring (RC) technique to further reduce $A(t)$. RC follows the same algorithm as DC but instead of first-fit it apples random-fit. Random-fit on vertex v essentially puts a random color chosen from a range $\{1, 2, \cdots, k\}$ to v that is absent in all of its neighboring vertices. Here $k \geq \Delta(g(t)) + 1$ where $\Delta(g(t))$ is the maximum degree of $g(t)$.

It is evident that both DC and RC are centralized approaches. We also propose a decentralized differential coloring (DDC) technique. In DDC using channel state information (CSI) and message passing we essentially implement the 2-approximation minimum vertex cover algorithm to find $V_c(t)$, without generating $g_c(t)$. After finding the $V_c(t)$, DDC also recolors those vertices in $V_c(t)$. For recoloring purpose, DDC essentially applies a distributed version of random-fit on the vertices to be recolored. Finally we calculate the expected cost produced by DC, DDC and RC and compare with some existing algorithms. We also verified the theoretical findings through simulation.

Rest of the chapter is organized as follows. We formally show that perturbation minimization is a NP-hard in Section 3.2. In section 3.3 we present our DC and RC algorithms. The DDC algorithm is presented in Section 3.4. In Section 3.5 we present our analysis of DC, DDC and RC. Here we first show that DC, DDC and RC provide 2-factor approximation of $A(t)$ and then calculate expected maximum color $\mathbb{E}[Y(t)]$ and expected number of perturbations $\mathbb{E}[A(t)]$ generated by DC, DDC and RC. In Section 3.6, we compare the performance of DC, DDC and RC with existing approaches in terms of $\mathbb{E}[Y(t)]$ and $\mathbb{E}[A(t)]$. In Section 4.7 we verify our theoretical findings through simulation. Finally Section 3.8 concludes the chapter.

The following notations will be frequently used in the rest of this chapter: $x(n) \sim_n y(n) \iff \lim_{n\to\infty} \dfrac{x(n)}{y(n)} = 1$ and $x(n) \lesssim_n y(n) \iff \lim_{n\to\infty} \dfrac{x(n)}{y(n)} \leq 1$.

3.2 Hardness of the problem

In this section we show that perturbation minimization is NP-hard.

Theorem 3.2.1 *Given $g(t)$ and $C(t-1)$ producing $C(t)$ such that $A(t)$ is minimized is NP-hard.*

Proof: We will show that given $g(t)$ and $C(t-1)$ producing $C(t)$ such that $A(t)$ is minimized is NP-hard, by reducing our problem to minimum vertex cover problem. Consider an oracle algorithm $Alg(g(t), C(t-1))$ produces $C(t)$ such that $A(t)$ is minimized. Suppose we want to find minimum vertex cover of graph G. Consider $g(t) = G$ and $c_v(t-1) = 1$, $\forall v$. If we set $C(t) = C(t-1)$, then $g_c(t) = g(t)$, and $V_c(t)$ is the minimum vertex cover of $g(t)$. Hence to minimize

$A(t)$, $Alg(g(t), C(t-1))$ have to recolor the vertices of the minimum vertex cover of $g(t)$. We can filter out the minimum vertex cover by putting the vertices who have changed colors in $C(t)$ from $C(t-1)$ in a set. Since finding minimum vertex cover is NP-complete, our problem is NP-hard. Hence the proof.

3.3 Differential Coloring (DC) and Random Coloring (RC)

Given $g(t)$ and $C(t-1)$ DC first sets $C(t) = C(t-1)$, then builds $g_c(t)$ and finally applies classical 2-approximation minimum vertex cover algorithm to find $V_c(t)$. Classical 2-approximation minimum vertex cover algorithm is as follow: Initially set $V_c(t) = \emptyset$. If there exists any monochromatic edge in $g_c(t)$ choose one random edge from $g_c(t)$ and include both of its endpoint in $V_c(t)$ and repeat the process. Note that this algorithm will essentially find a maximal matching and both endpoints of each of its edges are members of $V_c(t)$. Both DC and RC remove colors of the vertices of $V_c(t)$ from $C(t)$ and then color those vertices following a random order using first-fit (in case of DC) or random-fit (in case of RC) with $k \geq \Delta(g(t)) + 1$. We present DC and RC formally in Algorithms 5 and 6 respectively.

Algorithm 5: *Differential coloring (DC)*

```
Input: $g(t)$,$C(t-1)$
Output: $C(t)$
Set $C(t) = C(t-1)$;
Generate $g_c(t)$;
Find $V_c(t)$ of $g_c(t)$ using classical 2-approximation maximal matching algorithm;
Remove color of each vertex $\in V_c(t)$ from $C(t)$;
Recolor vertices of $V_c(t)$ using first-fit in a random order return $C(t)$;
```

Algorithm 6: *Random coloring (DC)*

```
Input: $g(t)$,$C(t-1)$,$k \geq \Delta(g(t)) + 1$
Output: $C(t)$
Set $C(t) = C(t-1)$;
Generate $g_c(t)$;
Find $V_c(t)$ of $g_c(t)$ using classical 2-approximation maximal matching algorithm;
Remove color of each vertex $\in V_c(t)$ from $C(t)$;
Recolor vertices of $V_c(t)$ using random-fit, with choosing colors from $\{1, 2, \cdots, k\}$,
  in a random order;
return $C(t)$;
```

3.4 Decentralized differential coloring (DDC)

In this section we present our decentralized differential coloring (DDC) algorithm to minimize $A(t)$. Initially all links in $g(t)$ will retain their previous 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 within r distance apart 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 which are within r distance apart. Communication power of a vertex will be set to transmit up to 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 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. Then i and j will then choose two random colors from $\{1, 2, \cdots, k\}$ and $\{1, 2, \cdots, k\}$, where k is a input parameter $\geq \Delta(g(t)) + 1$, 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 in $g(t)$ 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 7 which will be run by the user-pair forming link i.

Remark 3 *It is evident that at time t, each link has to exchange messages with only its neighbors. Again to form the pairing, each link may have to exchange its information to each of its neighbor $O(\frac{1}{(1 - \frac{1}{\Delta(g(t))+1})^{\Delta(g(t))}}) = O(e) = O(1)$ times. Hence expected message complexity for each link, to execute DDC at time t, is $O(\Delta(g(t)))$.*

3.5 Analysis of DC, DDC and RC

In this section we first show that given $g(t)$ and $C(t-1)$, DC, DDC and RC provide 2-factor approximation of $A(t)$ and then calculate expected maximum color $\mathbb{E}[Y(t)]$ and expected number of perturbation $\mathbb{E}[A(t)]$ generated by DC, DDC and RC.

Theorem 3.5.1 *Given $g(t)$ and $C(t-1)$, DC, DDC and RC provide 2-factor approximation of $A(t)$.*

Proof: Since DC, DDC and RC apply 2-approximation algorithm to calculate $V_c(t)$ of $g_c(t)$ and then color the endpoints of each vertex in the calculated $V_c(t)$, it can at most recolor 2-times the size of optimal $V_c(t)$. Since $|V_c(t)|$ is at least the optimum $A(t)$, the proof follows.

Algorithm 7: *Decentralized differential coloring (DDC)*

```
Input: $g(t)$,$C(t-1)$,$k \geq \Delta(g(t)) + 1$
Output: $C(t)$
For each vertex common in $V(g(t))$ and $V(g(t-1))$ set $c_i(t) = c_i(t-1)$;
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 random colors from $\{1, 2, \cdots, k\}$ and $\{1, 2, \cdots, k\}$, such that for
              each link the color of that link is not being used by their 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;
```

Theorem 3.5.2 *Given $g(t) = G(n, \lambda)$, $\lambda = O(1)$, and $C(t-1)$, DC produces*

$$\mathbb{E}[Y(t)] \quad \sim_n \quad \lambda$$

$$\textit{and}$$

$$\mathbb{E}[A(t)] \quad \sim_n \quad \lambda$$

respectively.

Proof: It is evident from [87] that first-fit on $g(t) = G(n, \lambda)$, $\lambda = O(1)$, over a random order, produces $\mathbb{E}[Y(t)] \sim_n \lambda$. Since DC does not recolor the vertices of $g(t)$

which do not belong to a vertex cover of $g_c(t)$, this may increase $\mathbb{E}[Y(t)]$ over time. If $Y(t-1) \sim_n \lambda$ then a vertex u can have color $c_u(t) \sim_n \lambda$ only if there exists neighbors of u with colors $1, 2, \cdots, \lambda$ in $C(t)$. Probability of this event is $\sim_n (\frac{\lambda}{n})^{\lambda} \frac{1}{\lambda^{\lambda}} \sim_n \frac{1}{n^{\lambda}}$. This probability $\rightarrow 0$ when $n \rightarrow \infty$. Hence DC produces $\mathbb{E}[Y(t)] \sim_n \lambda$. It is evident that an edge uv is in $g_c(t)$ with probability $P(uv \in g_c(t)) = P(uv \in g(t)$ & $uv \notin g(t-1)$ & $c_u(t-1) = c_v(t-1)) \sim_n \dfrac{\lambda}{n} \times (1 - \dfrac{\lambda}{n}) \times \dfrac{1}{\lambda} \sim_n \dfrac{1}{n}$. Since vertex u belongs to $g_c(t)$ only if it has at least one incident monochromatic edge, probability of that event is $P(u \in g_c(t)) \sim_n (1 - (1 - \dfrac{1}{n})^{\lambda}) \sim_n \dfrac{\lambda}{n}$. Hence expected number of vertices and expected average degree of $g_c(t)$ are $n \times P(u \in g_c(t)) \sim_n n \times \dfrac{\lambda}{n} \sim_n \lambda$ and $n \times P(uv \in g_c(t)) \sim_n n \times \dfrac{1}{n} \sim_n 1$ respectively. That means, each vertex in $g_c(t)$ has only one monochromatic edge incident to it. That is, $g_c(t)$ is essentially a collection of disjoint edges in expected sense and one endpoint of each monochromatic edge is a member of minimum vertex cover. Hence $\mathbb{E}[|V_c(t)|] \sim_n 0.5\lambda$. Since DC applies a 2-approximation algorithm to find the minimum vertex cover and recolor those vertices, from Theorem 3.5.1, it follows that $\mathbb{E}[A(t)] \sim_n 2\mathbb{E}[|V_c(t)|] \sim_n \lambda$.

Theorem 3.5.3 *Given $g(t) = G(n, \lambda)$, $\lambda = O(1)$, and $C(t-1)$, RC and DDC with parameter $k \geq \Delta(g(t)) + 1$ produces*

$$\begin{array}{rcl} \mathbb{E}[Y(t)] & \sim_n & k \\ & \textit{and} & \\ \mathbb{E}[A(t)] & \sim_n & \dfrac{\lambda^2}{k} \end{array}$$

respectively.

Proof: Since random-fit chooses a random color from $\{1, 2, \cdots, k\}$, using the proof of Theorem 3.5.2 we can show that $\mathbb{E}[Y(t)] \sim_n k$. Similar to the proof of Theorem 3.5.2 we can further show that $P(uv \in g_c(t)) \sim_n \dfrac{\lambda}{n}(1 - \dfrac{\lambda}{n})\dfrac{1}{nk} \sim_n \dfrac{\lambda}{nk}$. Thus expected number of vertices and expected average degree of $g_c(t)$ are $n \times P(u \in g_c(t)) \sim_n$

$n \times \dfrac{\lambda^2}{nk} \sim_n \dfrac{\lambda^2}{k}$ and $n \times P(uv \in g_c(t)) \sim_n n \times \dfrac{\lambda}{nk} \sim_n \dfrac{\lambda}{k}$ respectively. This implies together with Theorem 3.5.1 that $\mathbb{E}[A(t)] \sim_n \dfrac{\lambda^2}{k}$. It is evident that DDC is only a decentralized version of RC. Hence the results hold for DDC also. Hence the proof.

Remark 4 *It is evident that DDC is essentially a decentralized version of RC, except that $k = \Delta(g(t)) + 1$. Hence by replacing k by $\Delta(g(t)) + 1$ in the expression of RC in Theorem 3.5.3, we get the expected maximum and perturbations produce by DDC.*

Remark 5 *It is evident that RC essentially applies a random colors from $\{1, 2, \cdots, k\}$ to the vertices such that no monochromatic edge remains in $g(t)$. It is evident from Theorem 3.5.3 that $\mathbb{E}[Y(t)] \sim_n k$ and $\mathbb{E}[A(t)] \sim_n \frac{\lambda^2}{k}$. Thus by increasing k we can decrease $\mathbb{E}[A(t)]$ to a arbitrarily small value, with $\lim\limits_{k\to\infty} \mathbb{E}[Y(t)] = \infty$ and $\lim\limits_{k\to\infty} \mathbb{E}[A(t)] = 0$. This clearly indicates how RC efficiently uses the trade-off between $Y(t)$ and $A(t)$ to decrease $A(t)$.*

3.6 Comparison with other approaches

Authors in [53, 54, 55] have studied this problem with the property that at time t only one edge could appear or disappear. They call this incident as an update and try to maintain the coloring of the graph on a per-update basis. They fix a maximum color for the graph and upon occurrence of an update, change the color of some other vertices so that the proper coloring of the graph is maintained. At time t, some monochromatic edges may appear in $g(t)$. We consider each such monochromatic edge one by one and apply their algorithms. The computed $\mathbb{E}[Y(t)]$ and $\mathbb{E}[A(t)]$ of the above mentioned algorithms and the same produced by RC, DDC and DC along with their corresponding time and space complexities are presented in Table 3.1.

From Table 3.1 we observe the following. DC provides reasonably low $\mathbb{E}[A(t)]$ with keeping $\mathbb{E}[Y(t)]$ at its minimum, where the algorithm proposed in [54] provides the same $\mathbb{E}[Y(t)]$ though $\mathbb{E}[A(t)]$ is higher. It is evident from Table 3.1 that if $\mathbb{E}[Y(t)] = O(\lambda)$ then DC produces less $\mathbb{E}[A(t)]$ than the algorithm reported in [54]. Note that authors of [53] and [55] considered $\mathbb{E}[Y(t)] = O(\Delta(g(t)))$ and $\mathbb{E}[Y(t)] = O(\log(n))$ respectively. So if we adopt $k = \Delta(g(t)) + 1$ or $k = \log(n)$ in RC, RC produces lesser $\mathbb{E}[A(t)]$ than that produced by those two algorithms respectively. It is evident from Remark 5 that RC can decrease $\mathbb{E}[A(t)]$ to arbitrarily small value by considering $\mathbb{E}[Y(t)]$ to be very large. So we can conclude that due to the trade-off between $Y(t)$ and $A(t)$, by increasing color requirement we can decrease the number of perturbations. Therefore, according to the relative importance of $\mathbb{E}[Y(t)]$ and $\mathbb{E}[A(t)]$, sometimes DC and sometimes RC is the preferable choice for this problem.

Sl	Algorithm				Time & space
No	Paper	Name	$\mathbb{E}[Y(t)]$	$\mathbb{E}[A(t)]$	complexities
(1)		DC	λ	λ	$\Theta(n)$
(2)		RC	k	$\frac{\lambda^2}{k}$	$\Theta(n)$
(3)		DDC	$\Delta(g(t)) + 1$	$\frac{\lambda^2}{\Delta(g(t)) + 1}$	$\Theta(n)$
(4)	[53]		$O(\log(n))$	$O(0.5\frac{\lambda^2}{\log(n)})$	$\Theta(n)$
(5)	[54]		$O(\lambda)$	$O(0.5\lambda \times \text{polylog}(n))$	$\Theta(n)$
(6)	[55]		$O(\Delta(g(t)))$	$O(0.5\frac{\lambda^2 \log(\Delta(g(t)))}{\Delta(g(t))})$	$\Theta(n)$

Table 3.1: $\mathbb{E}[Y(t)]$ and $\mathbb{E}[A(t)]$ produced by different algorithms

3.7 Simulation

Since our analysis is asymptotic in nature, in this section, we verify our theoretical findings on DC and RC through simulations, considering finite values of n. We consider that n links are initially placed randomly inside a 1 km radius circular region and at each time instance t each link i chooses a random speed $v_i(t) \in [0, \mathbf{v}]$

and a random angle $\theta_i(t) \in [0, 2\pi]$ and move with it. That is, links are moving following the random way-point mobility model (RWM) [88]. At each time t, we build $g(t)$ by considering each link as a vertex and each pair of links residing within r distance have an edge between them. We consider the default values of n, r, $\mathbf{v}$ as $100,\ 100\ m,\ 10\ m/s$. We vary one of the parameters n, r and keep others at its default value and observe $Y = \mathbb{E}[Y(t)]$ and $A = \mathbb{E}[A(t)]$ generated by DC and RC with $k = 10$.

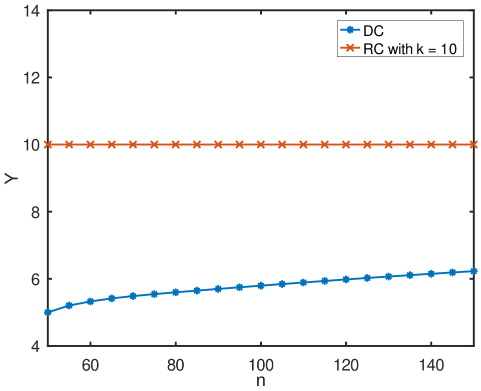

Figure 3.1: Y vs n

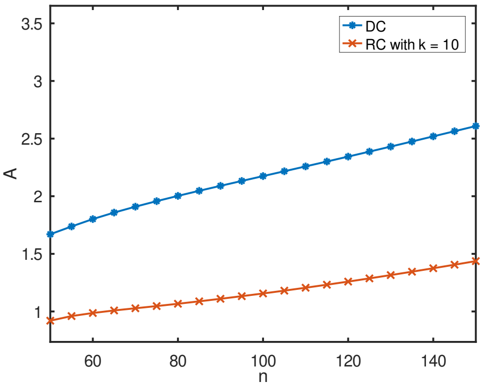

Figure 3.2: A vs n

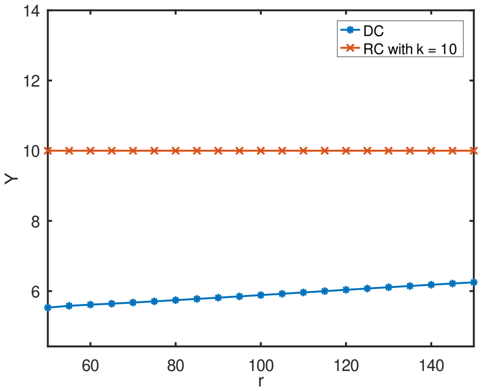

Figure 3.3: Y vs r

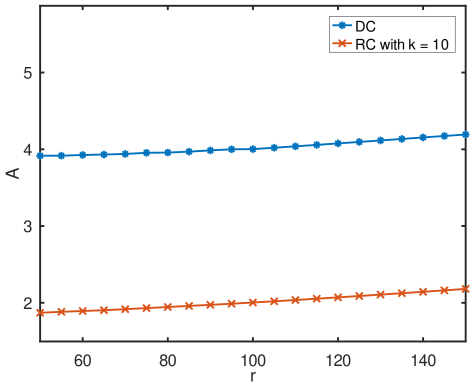

Figure 3.4: A vs r

In Figures 3.1 and 3.2 we vary n from 50 to 150 with an interval of 5 keeping $r = 100\ m$ and $\mathbf{v} = 10\ m/s$ and plot Y and A produced by DC and RC respectively. Also in Figures 3.3 and 3.4 we vary r from 50 m to 150 m with an interval of

5 m keeping $n = 100$ and $\mathbf{v} = 10\ m/s$ and plot Y and A produced by DC and RC respectively. It is evident from those figures that with increasing n and r, Y produced by DC increases while the Y produced by RC remains the same. Similarly with increasing n and r, A produced by DC and RC increases. Also Y produced by DC is lesser than that produced by RC. On the other hand, A produced by DC is greater than the same produced by the RC respectively. Note that average degree of $g(t)$ is $\lambda \propto nr^2$ where $\lambda \leq k$. From Table 3.1 we get that Y and A produced by DC are $\sim_n \lambda$ and λ respectively and those produced by RC are $\sim_n k$ and $\dfrac{\lambda^2}{k}$ respectively. Hence the experimental results are in accordance with theoretical finding present in Table 3.1.

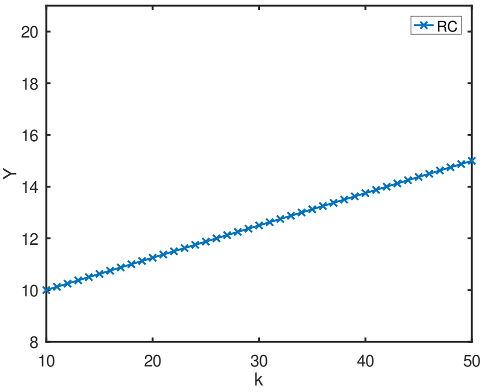

Figure 3.5: Y vs k

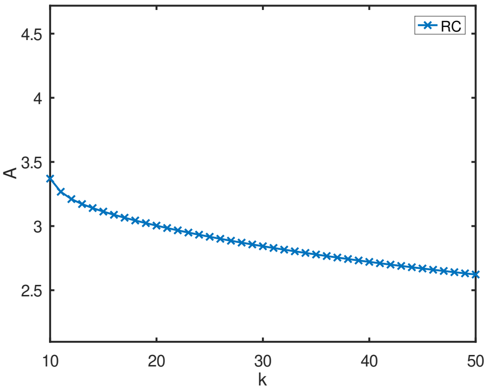

Figure 3.6: A vs k

In Figures 3.5 and 3.6 we vary k from 10 to 50 with an interval of 1, keeping $n = 100$ and $r = 100\ m$ and $\mathbf{v} = 10\ m/s$ and plot Y and A produced by RC. We observed that with increasing k, $Y \propto k$ and $A \propto \dfrac{1}{k}$. From Table 3.1 we get that Y and A produced by RC are $\sim_n k$ and $\dfrac{\lambda^2}{k}$ respectively. Hence the experimental results are again in accordance with theoretical finding present in Table 3.1.

3.8 Conclusion

In this chapter we have provided centralized and decentralized algorithms for minimizing $A(t)$ given $g(t)$ and $C(t-1)$. We first propose a DC algorithm which copies $C(t-1)$ to $C(t)$ and then find and color the minimum vertex cover of $g_c(t)$. Using the trade-off between $Y(t)$ and $A(t)$, we propose a RC algorithm which further minimize $A(t)$. To reduce the overhead of base station we then propose a decentralized differential coloring (DDC). Finally we calculated the expected maximum color and expected number of perturbations produced by our algorithms and compare those with some existing approaches. We also verify the theoretical results via simulation.

Chapter 4

Minimizing both maximum channel and total perturbation

4.1 Introduction

In chapter 3 we minimized total number of perturbations $A(t)$ required to keep every link active at time t. If we allocate each link a different channel then each link has no interference from other links and hence total perturbations $A(t)$ will be zero. But in that case, the maximum channel used $Y(t)$ is huge. Thus $Y(t)$ and $A(t)$ have a natural trade-offs among them. Motivated by this, in this chapter, we will minimize a cost defined as a linear combination of $Y(t)$ and $A(t)$.

We formulate the resource allocation problem as a cost minimization problem where cost is defined as $Y(t) + \alpha A(t)$. In device to device (D2D) communication, two proximity users can directly communicate among themselves, through a common channel, without the need of a base station [10]. At time t, a pair of D2D communicating users forms a link and each link v operates on a channel $c_v(t)$ [10]. Since users are moving and a pair of users forming a link resides in a close proximity, we can assume each link as a moving point in the geometric region. Since wireless channels are costly resources, we need to reuse the same channel for multiple links.

However, the links activated with the same channel may interfere to each other. Two links are said to be interfering to each other if they are $\leq r$ distance apart, where r denotes the interference range. At time t, interference relationship among the active links can be modeled as an interference graph $g(t) = (V(t), E(t))$, where each link represents a vertex and two vertices form an edge if and only if their representing links are interfering to each other. We assume that a pair of users forming a link is moving independently with speed $\leq \mathbf{v}$, where $\frac{\mathbf{v}}{r} = O(1)$. Consider that some n links are placed within a large geometric region and they are active for a long time. Hence $E(t)$ varies over time but $V(t)$ remains constant. This implies that $g(t)$ is indeed a temporal graph. To establish communication of each link, at time t, we have to allocate a channel/color vector $C(t) = (c_v(t))$, where $c_v(t) \in \{1, 2, \cdots, n = |V(t)|\}$ is the channel/color assigned to vertex $v \in V(t)$ at that time. For a large geometric region, due to law of large number and central limit theorem, the average number of links per unit area will converge to a finite constant. Hence $g(t) = G(n, \lambda)$ can be considered as a sparse random graph with n vertex, where each edge is generated with probability $\frac{\lambda}{n}$ and $\lambda = O(1)$. Note that average degree and maximum degree of $g(t)$ are $\sim_n \lambda$ and $\Delta(g(t)) \sim_n \Delta(G(n, \lambda)) = O(\frac{\log(n)}{\log\log(n)})$ respectively, where $x(n) \sim_n y(n) \implies \lim\limits_{n \to \infty} \frac{x(n)}{y(n)} = 1$. Since $g(t)$ varies over time, a channel assigned to a link at time $(t-1)$ may not remain as an interference-free channel anymore at time t and hence we need to do channel switching which involves switching delay and degrades quality of service (QoS). Consider $I_v(C(t), C(t-1))$ as an indicator variable which indicates 1 if $c_v(t) \neq c_v(t-1)$ and 0 otherwise. Since channels are costly and there is a delay associated with channel switch, at time t, to improve QoS, we have to minimize both maximum color $Y(t) = \max\limits_{v \in V(t)} c_v(t)$ used in $C(t)$ and total number of perturbations/channel switch $A(t) = \sum\limits_{v \in V(t)} I_v(C(t), C(t-1))$ in $C(t)$ from $C(t-1)$. It is evident that if we allocate different channel to each link then no perturbations will occur but in that case, $Y(t)$ will be large. On the other hand, if we fix $Y(t)$ at the chromatic number of $g(t)$, then perturbations will be huge. Hence $Y(t)$ and $A(t)$ have a natural trade-off [59]. Owing to this natural trade-off, we have adopted the expression of cost function $f(t) = Y(t) + \alpha A(t)$ as mentioned in [59]. Here $\alpha \in [0, \infty]$ is a constant representing the relative weights of $Y(t)$ and $A(t)$ with weight of $Y(t)$ normalized to 1. Thus perturbation sensitive channel assignment problem in D2D communication is essentially a temporal graph coloring problem which can formally be stated as: At each time $t \in \{1, 2, \cdots, T\}$, find $C(t)$ given $g(t) = G(n, \lambda)$ and $C(t-1)$ such that no monochromatic edge exists in $g(t)$ and the expected cost $\mathbb{E}[\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} f(t)]$ is minimized.

If α is small we have to minimize $Y(t)$ only. In the Chapter 2, we have dealt with the problem of minimizing $Y(t)$. If α is large, we have to minimize $A(t)$ only. In Chapter 3, we have dealt with the problem of minimizing $A(t)$. As mentioned in [53] and also to our best of knowledge; only little works had been done to solve the problem of maintaining coloring of a graph that evolves over time, which explicitly deals with trade-off between $Y(t)$ and $A(t)$. In this chapter, we consider the problem of minimizing $f(t) = Y(t) + \alpha A(t)$ by explicitly dealing with the trade-off between $Y(t)$ and $A(t)$.

Authors in [53, 54, 55] proposed another version of this problem. They consider that at time t only one edge could appear or disappear. They call this incident as an update and try to maintain the coloring of the graph on a per-update basis. They fix a maximum color for the graph and upon occurrence of an update, change the color of some other vertices so that the proper coloring of the graph is maintained. To do that they partition $V(t)$ into different subsets and arrange the subsets into different labels. Upon occurrence of an update, they exchange vertices between different labels. In their approaches they maintain various in-variants and use additional data structures. In [53] authors proposed a bucket based approach, which has later been improved by [54] in terms of required time per-update. In [55] authors proposed a randomized algorithm which partitions $V(t)$ into different labels and maintains the expected number of colors of $g(t)$ in $\log(\Delta(g(t)))$ time per-update, where $\Delta(g(t))$ is the maximum degree of $g(t)$.

As mentioned earlier, if we copy $C(t-1)$ to $C(t)$ some monochromatic edges may appear in $g(t)$. Graph induced by those monochromatic edges is termed as conflict graph $g_c(t)$. Since to resolve each monochromatic edge we have to recolor at least one endpoint of it, to resolve all edges in $g_c(t)$ we have to recolor the minimum vertex cover $V_c(t)$ of it. Considering the sequential nature of $g(t)$s, in [59] authors proposed a SNAP and a SMASH algorithm. SNAP colors $g(t)$ using *incremental coloring* (IC) [59], $\forall t$. IC copies $C(t-1)$ to $C(t)$ and then traverse the vertices following an order. While visiting vertex v it checks whether v is an endpoint of a monochromatic edge. If yes, it recolors v using first-fit, which puts to a vertex the minimum color absent in any of its neighbors. On the other hand SMASH minimizes $A(t)$ through minimizing $|V_c(t)|$. In $k+1$ time interval SMASH builds a SMASHed graph $g_s^k(t) = \bigcup_{\tau=0}^{k} g(t+\tau)$ and colors it applying IC, and retains that color for next k time instances. Though in this case, perturbation occurs only once in $k+1$ time instances, but to build a SMASHed graph, they require *future information*, which may not be available. To avoid this problem, in this chapter, we propose a geometric prediction (GP) based approach. GP uses the current position and maximum velocity of the users to *predict all possible graphs* that could be generated in future. In GP, in $k+1$ time interval, predicted graph $g_p^k(t)$ is built by adding edges between vertices whose corresponding links are $\leq (r + 2\mathbf{v}k)$ distance apart. We show that a pair of links currently residing at distance d, can come within r distance, in current and next k time instances, only if $d \leq (r + 2\mathbf{v}k)$. Thus $g_p^k(t)$ is the union of current and all possible graphs that could be generated in next k time instances. GP then colors the predicted graph using the differential coloring (DC) technique described in Chapter 3, and the obtained color vector is retained for the next k time instances.

As mentioned in Chapter 3, DC provides a 2-approximation solution to find and color the vertices of $V_c(t)$. Since DC limits recoloring the vertices of $V_c(t)$ only, $Y(t)$ may get increased over time. To overcome this limitation, in this chapter, we propose a perturbation sensitive greedy coloring (PGC) technique which finds the minimum $f(t) = Y(t) + \alpha A(t)$ given $g(t)$ and $C(t-1)$, in expected $O(n)$ time and space complexities, which is asymptotically minimum among all algorithms which can solve this particular problem. Since coloring of future graphs are functions of coloring of the current graph, to reduce $\mathbb{E}[f(t)]$, we also propose a graph union (GU) based approach which using the *past information* builds a union graph and then colors it using PGC. Here the union graph represents the union of current and *past k* graphs, where parameter k resembles how many previous graphs is considered for making the union graph. We finally compare GP and GU with some existing approaches and show that GU outperforms all of these existing approaches.

The remaining chapter is organized as follows: In Section 4.2 we propose the Integer Linear Programming (ILP) formulation of the problem. In Section 4.3 we present the geometric prediction (GP) based approach. In Section 4.4 we present the perturbation sensitive greedy coloring (PGC) algorithm using which in Section 4.5 we present the graph union based approach. We compare our approaches with existing works in Section 4.6. Section 4.8 concludes the Chapter.

4.2 Integer Linear Programming (ILP) formulation of the problem

In this section we present an Integer Linear Programming (ILP) formulation of the problem of minimizing $Y(t) + \alpha A(t)$. For each time instant t, we define the following binary optimization variables:

$$z_{ic}(t) = \begin{cases} 1 & \text{if color } c \text{ is assigned to vertex } i \\ 0 & \text{otherwise.} \end{cases}$$

$$y_c(t) \quad = \quad \begin{cases} 1 & \text{if color } c \text{ is assigned to at least one vertex} \\ 0 & \text{otherwise.} \end{cases}$$

Constraints 4.1 checks whether color c is allocated to at least least one vertex.

$$z_{ic}(t) \leq y_c(t) \; \forall \; i, c \tag{4.1}$$

Constraint 4.2 guarantees that each vertex is allocated with exactly one color.

$$\sum_{c=1}^{n} z_{ic} = 1 \; \forall \; i \tag{4.2}$$

Constraint 4.3 guarantees that endpoint of each edge gets different colors.

$$z_{ic} + z_{jc} \leq 1 \; \forall \; ij \in E(t) \; \forall c \tag{4.3}$$

Constraints 4.4 and 4.5 calculate $Y(t)$ and $A(t)$ respectively, where $\oplus$ represents the **XOR** operation. Since $z_{ic}(t-1)$ is a constant at time t, $z_{ic}(t) \oplus z_{ic}(t-1)$ is a linear operation.

$$Y(t) = \sum_{c=1}^{n} y_c(t) \tag{4.4}$$

$$A(t) = \frac{1}{2} \sum_{c=1}^{n} z_{ic}(t) \oplus z_{ic}(t-1) \tag{4.5}$$

Hence the problem is formulated as follows:

$$minimize \; Y(t) + \alpha A(t) \tag{4.6}$$

$$\textit{Subject to}$$

$$Constraints \text{ (4.1)-(4.5), } \textit{where}$$

$$y_c(t) \in \{0, 1\} \; \forall c \tag{4.7}$$

$$z_{ic}(t) \in \{0,1\} \; \forall i, c \tag{4.8}$$

4.3 Geometric prediction (GP) based approach

In geometric prediction based approach, using the current position and maximum velocity of the users, in $k+1$ time instances, we build a predicted graph which is a super-graph of union of current and all possible graphs that could be generated in current and next k instances. To build the predicted graph $g_p^k(t)$ at current time, we essentially join edges between pair of links which are $\leq (r + 2\mathbf{v}k)$ distance apart. We will show in Theorem 4.3.1 that two links can come within r distance, in current and next k time instances, only if they are currently residing at $\leq (r + 2\mathbf{v}k)$ distance apart. After building $g_p^k(t)$ we apply DC to color the vertices of it. GP is formally stated in Algorithm 8.

Algorithm 8: *Geometric prediction (GP) based approach*

```
Input: Current position of each link, $\mathbf{v}$, $r$, $k$, $C(t-1)$
Output: $C(t)$
if $t \equiv 0 \; mod \; (k+1)$ then
    Build $g_p^k(t)$ by joining an edge between each pair of vertices which are
      $\leq r + 2\mathbf{v}k$ distance apart;
    $C(t) = DC(g_p^k(t), C(t-1))$;
else
    $C(t) = C(t-1)$;
return $C(t)$;
```

Theorem 4.3.1 *Two links i and j can come within r distance, in current and next k time instances, only if they are currently residing at $\leq (r + 2\mathbf{v}k)$ distance apart.*

Proof: Since users are moving and a pair of users forming a link resides in a close proximity, we assumed each link as a moving point in the geometric region. Also we assumed that a pair of users forming a link is moving independently with speed $\leq \mathbf{v}$. Hence the maximum relative velocity of j with respect to i is $2\mathbf{v}$. Thus they can come within r distance, in current and next k time instances, only if they are currently residing at $\leq (r + 2\mathbf{v}k)$ distance apart. Hence the proof.

From Theorem 4.3.1 it is obvious that $g_p^k(t) \supseteq \bigcup_{\tau=0}^{k} g(t+\tau)$. GP generates and colors $g_p^k(t)$ using DC in $k+1$ time interval and retains that color for current and next k time intervals. Since GP colors $g_p^k(t)$ only once in $k+1$ time interval, it essentially minimizes the expected perturbation by a factor of k.

Theorem 4.3.2 *The expected cost produced by GP with parameter k is* $\mathbb{E}[f(t)] \sim_n \lambda(1 + \frac{2\mathbf{v}k}{r})^2(1 + \frac{\alpha}{k+1})$.

Proof: It is evident that average degree of a vertex of $g(t) = G(n, \lambda)$ in the large geometric area is proportional to r^2. Since average degree of $G(n, \lambda)$ is λ we can say $\lambda \propto r^2$. Since $g_p^k(t)$ is built by joining edges between the links residing within $\leq (r + 2\mathbf{v}k)$ distance apart, $g_p^k(t)$ has the average degree $\lambda(1 + \frac{2\mathbf{v}k}{r})^2$. That is $g_p^k(t) = G(n, \lambda(1 + \frac{2\mathbf{v}k}{r})^2)$.

In [87] authors state that first-fit following a random order S on $g(t) = G(n, \lambda)$ produces $\mathbb{E}[Y(t)] \sim_n \lambda$, where $x(n) \sim_n y(n) \iff \lim_{n\to\infty} \frac{x(n)}{y(n)} = 1$. This implies $g_p^k(t) = G(n, \lambda(1 + \frac{2\mathbf{v}k}{r})^2)$ has expected number of colors $\mathbb{E}[Y(t)] \sim_n \lambda(1 + \frac{2\mathbf{v}k}{r})^2$. The reason behind this is as follows. Consider $C(t-1)$ had $Y(t-1) \sim_n \lambda(1 + \frac{2\mathbf{v}k}{r})^2$. Now a vertex in $g_p^k(t)$ will get color $\sim_n \lambda(1 + \frac{2\mathbf{v}k}{r})^2 + 1$ only if all the colors $1, 2, \cdots, \lambda(1 + \frac{2\mathbf{v}k}{r})^2$ appear in its neighbors. Probability of this event is $(\frac{\lambda(1 + \frac{2\mathbf{v}k}{r})^2}{n})^{\lambda(1 + \frac{2\mathbf{v}k}{r})^2} \times \frac{1}{(\lambda(1 + \frac{2\mathbf{v}k}{r})^2)^{\lambda(1 + \frac{2\mathbf{v}k}{r})^2}} \sim_n \frac{1}{n^{\lambda(1 + \frac{2\mathbf{v}k}{r})^2}}$, which $\to 0$ when $n \to \infty$.

If we consider $C(t) = C(t-1)$ then the probability that an edge $uv \in g_c(t)$ is

$$P(uv \in g_c(t)) \quad = \quad P(uv \in g_p^k(t))P(uv \notin g_p^k(t - (k+1)))P(c_u(t) = c_v(t))$$

$$
\begin{aligned}
&\sim_n \frac{\lambda(1+\frac{2\mathbf{v}k}{r})^2}{n} \times (1 - \frac{\lambda(1+\frac{2\mathbf{v}k}{r})^2}{n}) \times \frac{1}{\lambda(1+\frac{2\mathbf{v}k}{r})^2} \\
&\sim_n \frac{1}{n}
\end{aligned}
$$

Again probability that a vertex u has at least one incident monochromatic edge is

$$
\begin{aligned}
P(u \in g_c(t)) &\sim_n 1 - (1 - P(uv \in g(t)))^{\lambda(1+\frac{2\mathbf{v}k}{r})^2} \\
&\sim_n \frac{\lambda(1+\frac{2\mathbf{v}k}{r})^2}{n}
\end{aligned}
$$

Thus total number of vertices in $g_c(t)$ and average degree of $g_c(t)$ are

$$
nP(u \in g_c(t)) \sim_n \frac{\lambda(1+\frac{2\mathbf{v}k}{r})^2}{n}
$$

and

$$
nP(uv \in g_c(t)) \sim_n 1
$$

respectively. This implies that each vertex in $g_c(t)$ has only one incident monochromatic edge and $g_c(t)$ is essentially collection of disjoint edges. Since in this case exactly one endpoint of each edge belongs to $V_c(t)$, $\mathbb{E}[|V_c(t)|] \sim_n 0.5\lambda(1+\frac{2\mathbf{v}k}{r})^2$. Since GP uses DC which applies a 2-approximation maximal matching based algorithm to find and recolor the vertices of $V_c(t)$ and perturbation occurs one once in the $k+1$ time intervals, $\mathbb{E}[A(t)] \sim_n \lambda(1+\frac{2\mathbf{v}k}{r})^2\frac{1}{k+1}$. Thus $\mathbb{E}[f(t)] \sim_n \lambda(1+\frac{2\mathbf{v}k}{r})^2(1+\frac{\alpha}{k+1})$. Hence the proof.

4.4 Perturbation sensitive greedy coloring (PGC)

It is evident that GP essentially minimizes $\mathbb{E}[f(t)]$ over time. In this section, we would discuss about a perturbation sensitive greedy coloring which guarantees to minimize $f(t)$ given $g(t)$ and $C(t-1)$ in expected polynomial time.

Greedy coloring is a well known technique to color vertices of a given graph. Greedy coloring on $g(t)$, visits the vertices following an order $S = \{v_{l_1}, v_{l_2}, \cdots, v_{l_n}\}$, $1 \leq l_i \leq n$, $1 \leq i \leq n$, and while visiting vertex v_{l_i} it applies first-fit on it. Greedy coloring on $g(t)$ following S produces a color vector $C^S(t)$.

In Chapter 2, we propose a selective search (SS) algorithm which starts with a random order S_1 and finds color vector $C^{S_1}(t)$ by applying greedy coloring on S_1. Note that $C^{S_1}(t)$ partitions $V(t)$ into $Y^{S_1}(t) = \max_v c_v^{S_1}(t)$ *pseudo-vertices* V_1, V_2, $\cdots$, $V_{Y^{S_1}(t)}$ where a pseudo-vertex $V_x = \{v \in V(t) : c_v^{S_1}(t) = x\}$, $1 \leq x \leq Y^{S_1}(t)$. Let $\pi(S_1)$ be a permutation of those pseudo-vertices and $L(\pi(S_1))$ be the set of all orders generated from $\pi(S_1)$ by permuting vertices within the same pseudo-vertex but keeping the order of the pseudo-vertices intact. SS actually considers $\rho - 1$ random permutations $\pi_i(S_1)$, $2 \leq i \leq \rho$ of those pseudo-vertices. For each such permutation SS chooses a random order $S_i \in L(\pi_i(S_1))$ and applies greedy coloring on it. Finally SS reports the minimum $Y(t)$ produced by greedy coloring on $S_1, S_2, \cdots S_\rho$. Note that SS deals with the case of $\alpha = 0$ where no perturbation cost is involve.

Here we adopt the essence of SS presented in Chapter 2 and develop a perturbation sensitive greedy coloring (PGC) algorithm for minimizing $f(t)$. PGC is exactly same as SS except *first-perturbation-fit* is used in place of first-fit when greedy coloring is applied on an order and finally the color vector with minimum cost is reported. When first-perturbation-fit is applied on a vertex v_{l_i} following order S, it puts the color which is absent in all of its allocated neighbors and for which $c^S_{v_{l_i}}(t) + \alpha I_{v_{l_i}}(C^S(t), C(t-1))$ is minimum. Let's define $Y^S_0(t) = 0$,

$Y_i^S(t) = \max(c^S_{v_{l_i}}(t), Y^S_{i-1}(t))$ and $A_i^S(t) = \sum_{j=1}^{i} I_{v_{l_j}}(C^S(t), C(t-1))$. It is evident that, while coloring vertex v_{l_i}, first-perturbation-fit essentially minimizes $f_i^S(t) = Y_i^S(t) + \alpha A_i^S(t)$, where $Y^S(t) = Y_n^S(t)$, $A^S(t) = A_n^S(t)$ and $f^S(t) = f_n^S(t)$. PGC is formally stated in Algorithm 9. We will later analytically show that PGC can find optimum $f(t)$ given $g(t)$ and $C(t-1)$ in expected polynomial time. Before proving this main result, we first show that there exists an order on which PGC produces optimal color vector and also present some properties of PGC which will be used to prove the main result.

Algorithm 9: *Perturbation sensitive greedy coloring (PGC)*

```
Input: $g(t)$,$C(t-1)$
Output: $C(t)$
Generate a random order $S$;
Calculate $C^S(t)$ by applying first-perturbation-fit following $S$;
$C(t) = C^S(t)$;
for $i = 1, 2, \cdots \rho$ do
    Consider a random permutation $\pi(S) \in \Pi(S)$ and a random order
      $S' \in L(\pi(S))$;
    Calculate $C^{S'}(t)$ by applying first-perturbation-fit following $S'$;
    if $f^{S'}(t) < f^S(t)$ then
        $C(t) = C^{S'}(t)$;
        $S = S'$;
        Reset $i = 1$;
return $C(t)$;
```

Lemma 4.4.1 *Given $g(t)$ and $C(t-1)$, there exists an order $S = (v_{l_1}, v_{l_2}, \cdots, v_{l_n})$, $1 \le l_i \le n$ and $1 \le i \le n$, which provides an optimum color vector $C^o(t)$ when first-perturbation-fit is applied on it.*

Proof: Let $C^o(t)$ be an optimal color vector such that $\sum_v c_v^o(t)$ is minimum among all optimal color vectors. We first build $S = (v_{l_1}, v_{l_2}, \cdots, v_{l_n})$ by sorting the vertices according to their colors in $C^o(t)$. Next we apply first-perturbation-fit on S. For each vertex v_{l_i}, first-perturbation-fit sets $c_{v_{l_i}}(t)$ as either the minimum color which is not present in any of its neighbors, or the color $c_{v_{l_i}}(t-1)$ corresponding to $C(t-1)$. Our claim is that for each v_{l_i}, $c^S_{v_{l_i}}(t) = c^o_{v_{l_i}}(t)$. We prove this by contradiction.

If possible, let i be the first position for which $c^S_{v_{l_i}}(t) \neq c^o_{v_{l_i}}(t)$. By definition of $C^o(t)$ this is possible only if $\exists\, v_{l_j}$, where $j > i$, such that $c^S_{v_{l_i}}(t) = c^o_{v_{l_j}}(t)$ and v_{l_i} and v_{l_j} have an edge between them. If there does not exists any such v_{l_j} then by replacing $c^o_{v_{l_i}}(t)$ by $c^S_{v_{l_i}}(t)$ in $C^o(t)$ and keeping the colors of all other vertices intact, we get a new color vector $C^{o'}(t)$ for which either $f^{o'}(t) < f^o(t)$ or $\sum\limits_v c^{o'}_v(t) < \sum\limits_v c^o_v(t)$. In both cases, we arrive at a contradiction.

If $\exists\, v_{l_j}$s with $j > i$, such that $c^S_{v_{l_i}}(t) = c^o_{v_{l_j}}(t)$ and v_{l_i} and v_{l_j} have an edge between them, we bring all such v_{l_j}s before v_{l_i} in S. Hence S is now modified and one of the v_{l_j}s would become the i-th vertex v_{l_i}. Since all v_{l_j}, where $j < i$ are already colored, we start coloring the new $v_{l_i}(t)$ using first-perturbation-fit. Note that now either $c^S_{v_{l_i}}(t) = c^o_{v_{l_i}}(t)$ or we have to repeat the above mentioned process. Since $|V|$ is finite, we will eventually be able to get a vertex at position i for which $c^S_{v_{l_i}}(t) = c^o_{v_{l_i}}(t)$. We then color the next vertices following the above mentioned process and finally get an updated order S for which $c^S_{v_{l_i}}(t) = c^o_{v_{l_i}}(t)$ for all i.

Lemma 4.4.2 *Consider that by applying first-perturbation-fit on a random order S we get color vector $C^S(t)$ with pseudo-vertices $V_1, V_2, \cdots, V_{Y^S(t)}$. Then*

1. *For any permutation $\pi(S)$, $\forall\, S', S'' \in L(\pi(S))$, $f^{S'}(t) = f^{S''}(t)$.*
2. $|L(\pi(S))| = \Omega((\frac{n}{eY^S(t)})^n)$.
3. $\pi_i(S) \neq \pi_j(S) \implies L(\pi_i(S)) \bigcap L(\pi_j(S)) = \emptyset$.

Proof: Since $S', S'' \in L(\pi(S))$, only the relative positions of the vertices within the same pseudo-vertex may be different, but the relative positions of the pseudo-vertices are not changed. Since a pseudo-vertex is an independent set and $c(t-1)$ is fixed, color of a vertex cannot change regardless of its position within the same pseudo-vertex. That is $\forall S \in L(\pi(S)), c_v^S(t) + \alpha I_v(C^S(t), C(t-1))$ remains the same $\forall v$. Hence the proof of Clause 1.

Since first-perturbation-fit produces $Y^S(t)$ pseudo-vertices $V_1, V_2, \cdots, V_{Y^S(t)}$, similar to [89] we can show $|L(\pi(S))| = \prod_{i=1}^{Y^S(t)}(|V_i|)! = \Omega((\frac{n}{eY^S(t)})^n)$, using Stirling's approximation. Hence the proof of Clause 2.

If $\pi_i(S) \neq \pi_j(S)$ then $\exists\ V_x, V_y, x, y \in \{1, 2, \cdots, Y(t)\}$ such that V_x appears before V_y in $\pi_i(S)$ and after V_y in $\pi_j(S)$. Hence the proof of Clause 3.

Theorem 4.4.1 *Given $g(t) = G(n, \lambda)$ and $C(t-1)$, PGC produces optimal color vector in expected $O(n)$ time and space complexities, which is asymptotically minimum among all possible randomized algorithm that solves this particular problem.*

Proof: In [87] authors state that first-fit following a random order S on $g(t) = G(n, \lambda)$ produces $\mathbb{E}[Y(t)] \sim_n \lambda$, where $x(n) \sim_n y(n) \iff \lim_{n\to\infty} \frac{x(n)}{y(n)} = 1$. It is evident that if $C(t-1)$ is colored with $\sim_n \lambda$ colors using PGC then $C(t)$ would be colored with $\sim_n \lambda$ colors as well. This is because vertex v in $g(t)$ can have color $\lambda + 1$ only if each of the colors $1, 2, \cdots, \lambda$ is present in its neighbors, probability of which is $\sim_n \frac{1}{\lambda^\lambda} \times (\frac{\lambda}{n})^\lambda = \frac{1}{n^\lambda} \to 0$, when $n \to \infty$. From Lemma 4.4.1 and Lemma 4.4.2 (Clauses 1 and 2), we can state that optimum hitting probability of a random order is $\frac{1}{n!} \times \Omega((\frac{n}{e\lambda})^n) = \Omega(\frac{1}{\lambda^n})$. Again using Lemma 4.4.2 (Clause 3) we get the expected number of orders, i.e., ρ, to hit optimum is $\mathbb{E}[\rho] = \frac{1}{\Omega((\frac{n}{e\lambda})^n) \times \Omega(\frac{1}{\lambda^n})} = O((\frac{e\lambda^2}{n})^n) = O(1)$. Since first-perturbation-fit has time and space complexity $O(|V(t)| + |E(t)|)$, where $|E(t)| \sim_n \frac{\lambda n}{2} = O(n)$ because $\lambda = O(1)$, PGC's expected time and space complexity to reach optimum is $O(n)$. It is also evident that each algorithm that solves this problem has to store $g(t)$ and search each vertex to find optimal $C(t)$. Hence the proof.

4.5 Graph union based approach

Since $g(t)$ evolves over time, coloring of $g(t)$ will affect the coloring of future graphs. To address the temporal nature of $g(t)$ we further propose a graph union based approach (GU). In GU instead of coloring $g(t)$, we build and color using PGC an union graph $g_u^k(t) = \bigcup_{\tau=0}^{k} g(t-\tau)$, which is union of current and *past* k graphs. Here parameter k resembles how many previous graphs will be considered for making the union graph. Let $g_c(t)$ be the graph induced by the monochromatic edges created in $g_u^k(t)$ if $C(t) = C(t-1)$. Note that an edge $uv \in g_c(t)$ only if $uv \in g(t)$ & $uv \notin g_u^k(t-1)$ & $c_u(t-1) = c_v(t-1)$. Thus with increasing k, GU basically decreases $\mathbb{E}[A(t)]$ at the cost of increasing $\mathbb{E}[Y(t)]$. GU is formally stated in Algorithm 10.

```
Algorithm 10: Graph union (GU) based approach
  Input: $g(t), g(t-1), \cdots, g(t-k), C(t-1), k$
  Output: $C(t)$
  Build $g_u^k(t) = \bigcup_{\tau=0}^{k} g(t-\tau)$;
  $C(t) = PGC(g_u^k(t), C(t-1))$;
  return $C(t)$;
```

Theorem 4.5.1 *Expected cost produced by GU is* $\mathbb{E}[f(t)] \sim_n \lambda(k+1+\frac{0.5\alpha}{k+1})$, *where* $\min_k \mathbb{E}[f(t)] \sim_n 2\lambda\sqrt{0.5\alpha}$.

Proof: As mentioned in the proof of Theorem 4.4.1, PGC and similarly first-fit on $g_u^k(t) = G(n,(k+1)\lambda)$ produces $\mathbb{E}[Y(t)] \sim_n (k+1)\lambda$. Note that $(k+1)\lambda$ is the expected chromatic number of $G(n,(k+1)\lambda)$ [87]. It is evident that for each edge uv, $P(uv \in g_c(t)) = P(uv \in g(t)$ & $uv \notin g_u^k(t-1)$ & $c_u(t-1) = c_v(t-1)) \sim_n \frac{\lambda}{n} \times (1 - \frac{(k+1)\lambda}{n}) \times \frac{1}{(k+1)\lambda} \sim_n \frac{1}{(k+1)n}$. Since only the edges belong to $g(t)$ can be monochromatic in $g_u^k(t)$, probability that a vertex u has at least one incident monochromatic edge is $P(u \in g_c(t)) \sim_n (1-(1-\frac{1}{(k+1)n})^{\lambda}) \sim_n \frac{\lambda}{(k+1)n}$. Hence expected number of vertices and expected average degree of $g_c(t)$ are $n \times P(u \in g_c(t)) \sim_n n \times \frac{\lambda}{(k+1)n} \sim_n \frac{\lambda}{k+1}$ and $n \times P(uv \in g_c(t)) \sim_n n \times \frac{1}{(k+1)n} \sim_n \frac{1}{(k+1)}$ respectively. That means each vertex in $g_c(t)$ has only one monochromatic edge incident to it in an expected sense and hence recoloring one vertex per edge is enough to resolve the monochromatic edges. That is $g_c(t)$ is essentially a collection of disjoint edges in expected sense. Hence $\mathbb{E}[|V_c(t)|] \sim_n 0.5\frac{\lambda}{k+1}$. At time t, given $C(t-1)$, if an oracle algorithm sets $C(t) = C(t-1)$ and then finds and recolors only the vertices of $V_c(t)$ with first-fit, it will produce $\mathbb{E}[f(t)] \sim_n \lambda(k+1+\frac{0.5\alpha}{k+1})$. Since $A(t) \geq |V_c(t)|$, from Theorem 4.4.1, it implies that PGC on $g_u^k(t)$ produces $\mathbb{E}[A(t)] = \mathbb{E}[|V_c(t)|]$. Hence $\mathbb{E}[f(t)] \sim_n \lambda(k+1+\frac{0.5\alpha}{k+1})$. Thus $\min_k \mathbb{E}[f(t)] \sim_n 2\lambda\sqrt{0.5\alpha}$ where $k+1 \sim_n \sqrt{0.5\alpha}$. Hence the proof.

Remark 6 *It is evident that in GU we have to store k previous graphs in queue to construct $g_u^k(t)$. We can decrease the memory burden by keeping n^2 variables $s_{ij}(t)$s, where $s_{ij}(t) = \iota J_{ij}(t) + (1-\iota)s_{ij}(t-1)$. Here $J_{ij}(t) = 1$ if edge $ij \in g(t)$ otherwise 0 and $\iota > 0.5$. At time t we make $g_u^k(t)$ by joining each edge ij if corresponding $s_{ij}(t) \geq \iota(1-\iota)^k$. Since $\mathbb{E}[J_{ij}(t)] \sim_n \frac{\lambda}{n}$ we further get that*

$$\mathbb{E}[s_{ij}(t)] = \iota\mathbb{E}[J_{ij}(t)] + (1-\iota)\mathbb{E}[s_{ij}(t-1)] \implies \mathbb{E}[s_{ij}(t)] \sim_n \frac{\lambda}{n}.$$

Thus through this method we can also find λ of an unknown network topology by calculating $\mathbb{E}[s_{ij}(t)]$ over time.

4.6 Comparison with existing approaches

We calculate $\mathbb{E}[f(t)]$ produced by SNAP [59], RC, GP using the techniques mentioned in the proofs of Theorems 4.4.1 and 4.5.1. Note that algorithms proposed by [53, 54, 55] maintain coloring of a graph on a per update basis. At time t, some monochromatic edges may appear in $g(t)$. We consider each such monochromatic edge one by one and apply their algorithms. The computed $\mathbb{E}[Y(t)]$, $\mathbb{E}[A(t)]$ and $\mathbb{E}[f(t)]$ of the above mentioned algorithms and the same produced by GU with $k+1 \sim_n \sqrt{0.5\alpha}$ along with their corresponding time and space complexities are presented Table 4.1. From Table 4.1 we can conclude that $\mathbb{E}[f(t)]$ produced by each of the above mentioned algorithms is Ω of the same produced by GU with $k+1 \sim_n \sqrt{0.5\alpha}$.

Sl	Algorithm					Time & space
No	Paper	Name	$\mathbb{E}[Y(t)]$	$\mathbb{E}[A(t)]$	$\mathbb{E}[f(t)]$	complexities
(1)	[59]	SNAP	λ	0.5λ	$\lambda(1+0.5\alpha) = \Omega(\lambda)$	$\Theta(n)$
(2)		RC	k	$\frac{\lambda^2}{k}$	$k+\alpha\frac{\lambda^2}{k} =$ $\Omega(\max(2\sqrt{\alpha}\lambda, \frac{\log(n)}{\log\log(n)}))$	$\Theta(n)$
(3)		GP	$\lambda(1+\frac{2\mathbf{v}k}{r})^2$	$\frac{\lambda(1+\frac{2\mathbf{v}k}{r})^2}{k+1}$	$\lambda(1+\frac{2\mathbf{v}k}{r})^2(1+\alpha\frac{1}{k+1}) = \Omega(\lambda)$	$\Theta((k+1)n)$
(4)	[53]		$O(\log(n))$	$O(0.5\frac{\lambda^2}{\log(n)})$	$O(\log(n)(1+0.5\alpha\frac{\lambda^2}{\log(n)}))$	$\Theta(n)$
(5)	[54]		$O(\lambda)$	$O(0.5\lambda \times \text{polylog}(n))$	$O(\lambda(1+0.5\alpha \times \text{polylog}(n)))$	$\Theta(n)$
(6)	[55]		$O(\Delta(g(t)))$	$0.5\frac{\lambda^2\log(\Delta(g(t)))}{\Delta(g(t))}$	$O(\Delta(g(t)) + O(0.5\alpha\frac{\lambda^2\log(\Delta(g(t)))}{\Delta(g(t))})$	$\Theta(n)$
(7)	GU		$\lambda\sqrt{0.5\alpha}$	$\lambda\sqrt{0.5\alpha}$	$2\lambda\sqrt{0.5\alpha} = \Omega(\lambda)$	$\Theta(n)$

Table 4.1: $\mathbb{E}[f(t)]$ produced by different algorithms

4.7 Simulation

Since our analysis is asymptotic in nature, in this section, we simulate GP and GU for finite values of n and verify the obtained results with that of the theoretical findings. Here we consider n links are initially placed randomly inside a 1 km radius circular region and at each time instance t each link i chooses a random speed $v_i(t) \in [0, \mathbf{v}]$ and a random angle $\theta_i(t) \in [0, 2\pi]$ and moves with it. That is, we assume that links follow random way-point mobility model (RWM) [88]. At each time t, we build $g(t)$ by considering each link as a vertex and each pair of links residing within r distance have an edge between them. We consider the default values of $n, r, \mathbf{v}, \alpha$ and k as 100, 100 m, 10 m/s, 10 and 2. We vary one of the parameters n, r, k and keep others at their default values and observe $Y = \mathbb{E}[Y(t)]$

and $A = \mathbb{E}[A(t)]$ generated by GP and GU.

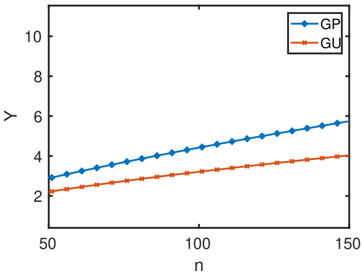

Figure 4.1: Y vs n

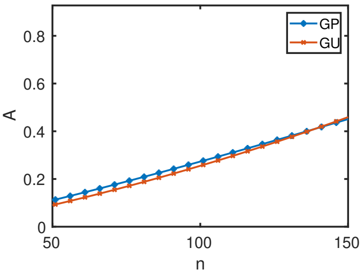

Figure 4.2: A vs n

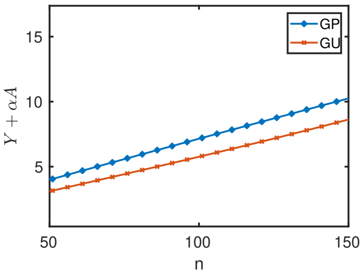

Figure 4.3: $Y + \alpha A$ vs n

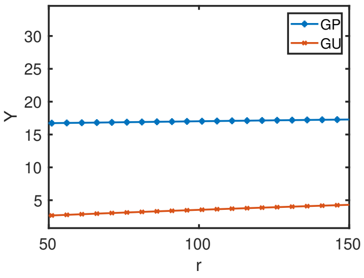

Figure 4.4: Y vs r

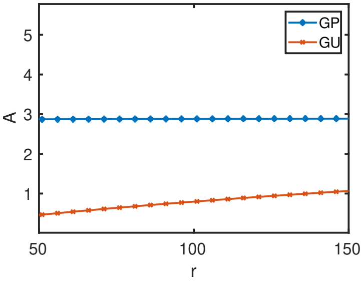

Figure 4.5: A vs r

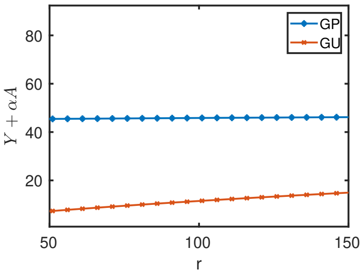

Figure 4.6: $Y + \alpha A$ vs r

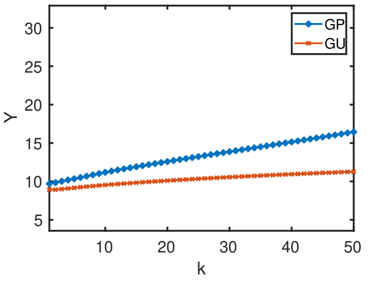

Figure 4.7: Y vs k

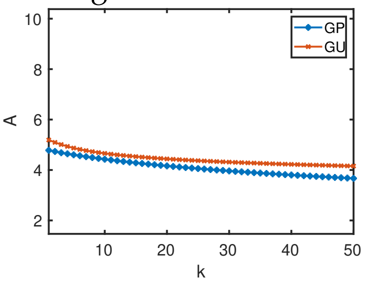

Figure 4.8: A vs k

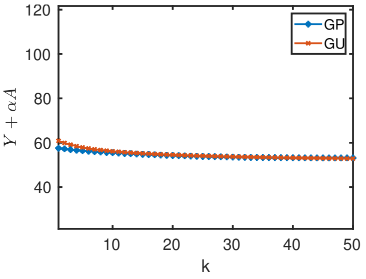

Figure 4.9: $Y + \alpha A$ vs k

In Figures 4.1-4.3 we vary n from 50 to 150 with an interval of 5, in Figures 4.4-4.6 we vary r from 50 m to 150 m with an interval of 5 m, and in Figures 4.7-4.9 we vary k from 1 to 50 with an interval of 1, keeping other parameters fixed at their default values, and plot Y, A and $Y + \alpha A$ produced by GP and GU respectively. From Figures 4.1-4.9, we observe that with increasing n and r, each of Y, A and $Y + \alpha A$ produced by both GP and GU increases. Again with increasing k, for both GP and GU, Y increases and A decreases. From Table 4.1 we get that for GP,

$Y \sim_n \lambda(1 + \frac{2\mathbf{v}k}{r})^2$, $A \sim_n \frac{\lambda(1+\frac{2\mathbf{v}k}{r})^2}{k+1}$ and $Y + \alpha A \sim_n \lambda(1 + \frac{2\mathbf{v}k}{r})^2(1 + \alpha\frac{1}{k+1})$ respectively. Again from Theorem 4.5.1 for GU, $Y \sim_n \lambda(k+1)$, $A \sim_n \frac{0.5\lambda}{k+1}$ and $Y + \alpha A \sim_n \lambda(k+1+\frac{0.5\alpha}{k+1})$ respectively. Note that with increasing n and r average degree of $g(t) \sim_n \lambda \propto nr^2$ increases. Also, according to the expression of both GP and GU, with increasing k, Y increases and A decreases. Hence the experimental results are in accordance with the theoretical findings.

4.8 Conclusion

In this chapter, we first model the interference relationship among the active links in D2D communication as a time-varying graph $g(t)$ and then proposed PGC algorithm which finds minimum cost $f(t)$ in expected polynomial time and space complexities. Next using PGC we proposed a GU algorithm. We have shown that $\mathbb{E}[f(t)]$ produced by some existing algorithms is Ω of the same produced by GU. Finally we compare our algorithm with SNAP and validate the theoretical findings through simulation.

Chapter 5

Minimizing both maximum channel and total power

5.1 Introduction

In the previous chapters we consider minimization of maximum channel and total perturbation. There we assumed that the power assignment is dealt by the base station. In this chapter we also consider the power allocation along with channel allocation in a general set up. If we allocate each link a different channel then each link has no interference from other links and hence can operate with minimum power. But in that case, the maximum channel used $Y(t)$ is huge. Thus $Y(t)$ and $P(t)$ have a natural trade-offs among them. Owing this natural trade-off, we will minimize a cost defined as a linear combination of $Y(t)$ and $P(t)$.

In device to device (D2D) communication, two users residing within the transmission range of each other can communicate directly among themselves over a common channel without involving the base station (BS). A cellular user communicates with the BS by forming a cellular link (CL) and a pair of D2D users communicate among themselves by forming a D2D link (DL). In D2D underlaid cellular network, DLs reuse the same up-link channel resources of CLs [90, 91, 64].

Each link (CL or DL) includes a transmitter and a receiver. The transmitter of each link has to be allocated sufficient power such that it can communicate with its receiver in the presence of noise and interference from other links operating on the same channel. More specifically, the allocated power of a transmitter must satisfy the required signal to interference plus noise ratio (SINR) at the receiver of that link. Each link requires certain level of SINR depending on its data rate requirement. Moreover, the allocated power at a transmitter must not exceed the residual power available at it.

Note that under a BS only one CL can use a particular channel. However, the channel of a CL may be shared by multiple DLs provided the required SINR is satisfied for each link sharing the channel. Hence, at time t, we have to find a channel vector $C(t) = (c_i(t))$ for the n links, where $c_i(t)$ represents the channel allocated to link i. Due to the scarcity of channels, we always have to minimize maximum channel $Y(t) \max_i c_i(t)$ used in the communication. We assumed that channels are positive integers starting from 1.

It is evident that maximum power that can be allocated to the transmitter of a link is a limited quantity. If link i is activated with power $x_i(t)$ then $x_i(t) \leq \eta_i(t)$, where $\eta_i(t)$ is the maximum power available at the transmitter of link i. Hence we have to find a power vector $X(t) = (x_i(t))$ for the n links, such that the total power requirement $P(t) = \sum_{i=1}^{n} x_i(t)$ is minimized.

The links activated with the same channel will interfere to each other. If more links are activated with the same channel to keep the channel requirement at low, the power requirement of the corresponding links would be high. On the other hand, if each link is activated with a different channel, the power requirement will be minimum but the channel requirement will be maximum. It is thus evident that $Y(t)$ and $P(t)$ have a natural trade-off. Owing to this natural trade-off, we define our minimization objective as a cost function $f(t) = Y(t) + \alpha P(t)$. Here α is a constant reflecting the relative weights of $Y(t)$ and $P(t)$, where the weight of

Y is normalized to 1. The joint power and channel allocation problem (JPCAP) deals with the problem of finding the channel vector $C(t)$ and the power vector $X(t)$ such that the required SINR criteria for each link is satisfied and the cost $f(t) = Y(t) + \alpha P(t)$ is minimized.

We formulate JPCAP as a cost minimization problem where the cost $f(t) = Y(t) + \alpha P(t)$ is designed such a manner that by properly tuning α we can set the goal of JPCAP to minimize $Y(t)$ only or $P(t)$ only or a joint objective of $Y(t)$ and $P(t)$. Then we reduce JPCAP to the classical graph coloring problem and thereby show that it is NP-hard and also providing $n^{1/\epsilon}$ approximation to JPCAP $\forall \epsilon > 0$ is NP-hard. Next we propose a mixed integer linear programming (MILP) formulation for JPCAP and subsequently develop a greedy channel and power allocation (GCPA) algorithm for it. GCPA works by taking an order of the links as input. We show that there exists an order of the links on which if GCPA is applied it will provide an optimal solution. Then we develop a method to search orders efficiently. We show that an order is equivalent to many orders. We develop an incremental algorithm (IA) which searches orders from different equivalent sets and thereby evaluating less number of orders, it essentially explores large number of orders. Finally, using IA, we design a randomized joint channel and power allocation (RJCPA) algorithm to find the near optimum solution. We also theoretically calculate the expected cost produced by RJCPA. Moreover, we identify some special cases where RJCPA can produce optimal result in expected polynomial time. We also compute the expected energy efficiency (EE) produced by RJCPA. Here EE is the data rate per spectrum per energy unit. We perform extensive simulations to show that RJCPA outperforms both the two-step approach [64] and RSBI algorithm [10] with respect to both cost and EE significantly. Finally we validate our theoretical findings through simulation.

5.1.1 Problem Formulation

Suppose there are n links within the coverage region of a BS where each link is either a CL or a DL. Let S_{CL} be the set of CLs and S_{DL} be the set of DLs where $n = |S_{CL} \cup S_{DL}|$. Let channels are represented by positive integers and $S_c = \{1, 2, \cdots, n\}$ be the set of available channels. Each link i requires a channel $c_i(t) \in S_c$ and power $x_i(t) \in [0, \eta_i(t)]$ for its activation, where $\eta_i(t)$ is the residual power available at the transmitter of link i. It is evident that $x_i(t)$ consists of power consumption of transmitter of link i and power loss at circuitry blocks of both transmitter and receiver of link i [10]. When communication is not taking place $x_i(t)$ is considered to be 0 by neglecting the minute leakage current [92]. If $x_j(t)$ power is allocated at the transmitter of link j then the power received at the receiver of link i can be expressed as $x_j(t)G_{ij}(t)$, where $G_{ij}(t) = h_{ij}^{fast}(t)h_{ij}^{slow}(t)d_{ij}^{-\beta}(t)$ [27] is the gain at the receiver of link i from the transmitter of link j, $h_{ij}^{fast}(t)$ is fast fading gain, a log-normally distributed random variable, $h_{ij}^{slow}(t)$ is slow fading gain, a exponentially distributed random variable, $d_{ij}(t)$ is the Euclidean distance between the transmitter of link j and the receiver of link i and $\beta > 1$ is the pathloss exponent. Note that our solution technique is independent of how $G_{i,j}(t)$ is computed.

Note that each link i will receive interference from every other link j for which $c_i(t) = c_j(t)$ where $i \neq j$. Let $\gamma_i(t)$ be minimum SINR required at the receiver of link i to satisfy its data rate requirement. Link i can be activated with $c_i(t)$ and $x_i(t)$ if $SINR_i(t)$, the SINR received at the receiver of link i, is greater than or equals to $\gamma_i(t)$. That is,

$$SINR_i(t) = \frac{x_i(t)G_{ii}(t)}{\sigma^2 + \sum\limits_{\substack{j\,:\,j \neq i \\ \& \\ c_i(t)=c_j(t)}} x_j(t)G_{i,j}(t)} \geq \gamma_i(t), \tag{5.1}$$

where σ^2 is the constant noise over each link.

We denote $y_c(t) = 1$ if channel $c \in S_c$ is allocated to at least one link, else 0. Note that each CL requires a different channel to communicate [64] and hence $c_i(t) \neq c_j(t)$ for all $i, j \in S_{CL}$ where $i \neq j$. But a DL may share channel with other CL and/or DL. Clearly total number of distinct channels used is given by $Y(t) = \sum_{c=1}^{n} y_c(t)$ and total power used in the communication is given by $P = \sum_{i=1}^{n} x_i(t)$.

Given a constant α, our objective is to find a channel vector $C = (c_i(t))$ and a power vector $X = (x_i(t))$ such that 1) cost $f(t) = Y(t) + \alpha P(t)$ is minimized, 2) each activated CL gets different channel and 3) $SINR_i(t)$, the SINR received at the receiver of each activated link i, satisfies Constraint (5.1).

Rest of the chapter is organized as follows. In Section 5.2 we formally prove that JPCAP is NP-hard and then in Section 5.3 we propose a mixed integer linear programming formulation for this problem. Since solving MILP is NP-hard, we propose a greedy channel and power allocation (GCPA) algorithm and analyse it in Section 5.4. Using GCPA, we develop a random joint power and channel allocation (RJCPA) algorithm and analyse it in Section 5.5. We theoretically calculated expected cost and expected energy efficiency in Section 5.6. In Section 5.7 we compare our algorithms with some existing algorithms and validate theoretical findings. Finally in Section 5.8 we conclude the chapter.

5.2 Hardness of JPCAP

Theorem 5.2.1 *JPCAP is NP-hard and even providing $n^{1/\epsilon}$ approximation of it is also NP-hard $\forall \epsilon > 0$.*

Proof: Consider $\alpha = 0$. Thus the problem becomes reducing only $Y(t)$. We consider $\sigma^2 = 1$, $\eta_i(t) = 1\ \forall i$, $\gamma_i(t) = 1\ \forall i$ and $S_{CL} = \emptyset$. Finally we consider $G_{ii}(t) = 1\ \forall i$ and $G_{ij}(t) = G_{ji} \in \{0, 1\}\ \forall i, j$ where $i \neq j$.

Now consider a graph $G(V, E)$, where V is the set of vertices and E is the set of edges of G. We consider each vertex as a link and hence V becomes the set of links. If there is an edge $\{i, j\} \in E$ between two links i and j, we set $G_{ij}(t) = G_{ji}(t) = 1$ else 0. Hence by this construction we can eventually capture any possible graph with $n = |V|$ vertices. Now if $G_{ij}(t) = G_{ji}(t) = 1$ for some $i,\ j$ where $i \neq j$, then Constraint (5.1) implies $x_i(t) \geq 1 + x_j(t)$ and $x_j(t) \geq 1 + x_i(t)$ for i and j respectively as $\sigma^2 = 1$ and $\gamma_i(t) = 1\ \forall i$. Also $x_i(t), x_j(t) \leq 1$ as $\eta_i(t) = 1\ \forall i$. It is evident that $x_i(t) \geq 1 + x_j(t)$, $x_j(t) \geq 1 + x_i(t)$ and $x_i(t), x_j(t) \leq 1$, together have no solution. Hence links i and j cannot get the same channel if and only if $\{i, j\} \in E$. Hence our problem gets reduced to the graph coloring problem. That is, if there is a solution $C(t)$ and $X(t)$ to our problem then that obtained $C(t)$ will be the solution of the classical graph coloring problem on graph G. The classical graph coloring problem is a well known NP-Complete problem [11] and even providing $n^{1/\epsilon}$ approximation $\forall \epsilon > 0$ of it is also NP-hard [12]. Hence the proof.

We now propose a mixed integer linear programming (MILP) formulation for JPCAP.

5.3 MILP Formulation

For all $c \in S_c$ and $i \in S_{CL} \cup S_{DL}$, we define the following binary optimization variables:

$$z_{ic}(t) = \begin{cases} 1 & \text{if channel } c \text{ is used by link } i \\ 0 & \text{otherwise.} \end{cases}$$

$$y_c(t) = \begin{cases} 1 & \text{if channel } c \text{ is used by at least one link} \\ 0 & \text{otherwise.} \end{cases}$$

Apart from the above binary variables, we define $x_i(t) \in [0, \eta_i(t)]$ as a real optimization variable representing the power allocated at link i. Based on these variables, we can formulate JPCAP as a mixed integer linear programming:
Constraint (5.2) checks whether channel c is being used by at least one link.

$$z_{ic}(t) \leq y_c(t) \ \forall i, c \tag{5.2}$$

Constraint (5.3) guarantees that a channel could be used by at most one CL.

$$\sum_{i \in S_{CL}} z_{ic}(t) \leq 1 \ \forall c \tag{5.3}$$

Constraint (5.4) guarantees that each link is allocated with exactly one channel.

$$\sum_{c=1}^{n} z_{ic}(t) = 1 \ \forall i \tag{5.4}$$

Let us define $\psi_{ic}(t) = x_i(t) z_{ic}(t)$ as an intermediate variable, where $\psi_{ic}(t) \in [0, \eta_i(t)]$. Note that $\psi_{ic}(t) = x_i(t) \leq \eta_i(t)$, the power $x_i(t)$ allocated to link i, if channel c is assigned to link i and 0, otherwise. Constraints (5.5), (5.6) and (5.7) linearize $\psi_{ic}(t) = x_i(t) z_{ic}(t)$ as follows. Constraints (5.5), (5.6) and (5.7) together guarantee that if $z_{ic}(t) = 0$ then $\psi_{ic}(t) \leq 0$, $\psi_{ic}(t) \leq x_i(t)$ and $\psi_{ic}(t) \geq x_i(t) - \eta_i(t) = -(\eta_i(t) - x_i(t))$ respectively, implying $\psi_{ic}(t) = 0$. Constraints (5.5), (5.6) and (5.7) together guarantee that if $z_{ic}(t) = 1$ then $\psi_{ic}(t) \leq \eta_i(t)$, $\psi_{ic}(t) \leq x_i(t)$ and $\psi_{ic}(t) \geq x_i(t)$ respectively, implying $\psi_{ic}(t) = x_i(t) \leq \eta_i(t)$.

$$\psi_{ic}(t) \leq \eta_i(t) z_{ic}(t) \ \forall i, c \tag{5.5}$$

$$\psi_{ic}(t) \leq x_i(t) \ \forall i, c \tag{5.6}$$

$$\psi_{ic}(t) \geq x_i(t) - (1 - z_{ic}(t)) \eta_i(t) \ \forall i, c \tag{5.7}$$

Constraint (5.8) ensures that if $z_{ic}(t)) = 1$, the SINR requirement of link i as represented by Equation (5.1) is satisfied. Here M represents a large positive value.

$$(1 - z_{ic}(t))M + x_{ic}(t)G_{ii}(t) \geq \gamma_i(t)(\sigma^2 + \sum_{j \neq i} G_{ij}(t)\psi_{jc}(t)), \ \forall i, c \tag{5.8}$$

Constraints (5.9) and (5.10) calculate $Y(t)$, the maximum channels assigned and $P(t)$, the total power $P(t)$ used, respectively.

$$Y(t) = \sum_{c=1}^{n} y_c(t) \tag{5.9}$$

$$P(t) = \sum_{i=1}^{n} x_i(t) \tag{5.10}$$

Hence JPCAP can be formulated as:

$$minimize\ Y(t) + \alpha P(t) \tag{5.11}$$

$$\textit{Subject to}$$

$$Constraints\ (5.2)\text{-}(5.10), \textit{ where}$$

$$y_c(t) \in \{0, 1\}\ \forall c \tag{5.12}$$

$$x_i(t) \in [0, \eta_i(t)]\ \forall i \tag{5.13}$$

$$z_{ic}(t) \in \{0, 1\}\ \forall i, c \tag{5.14}$$

$$\psi_{ic}(t) \in [0, \eta_i(t)]\ \forall i, c \tag{5.15}$$

As solving MILP is a NP-hard problem we now formulate a greedy channel and power allocation algorithm to solve JPCAP.

5.4 Greedy Channel and Power Allocation Algorithm

In this section, we propose a greedy channel and power allocation (GCPA) algorithm to allocate channels and powers to the links. Let $S = (l_1, l_2, \cdots, l_n)$ be an arbitrary order of the links. In GCPA, we visit the links one by one following S and allocate channels and powers to them. Thus while allocating link l_i, all the links $l_1, l_2, \cdots, l_{i-1}$ have already been allocated. In other words, $c_{l_j}(t)$ and $x_{l_j}(t)$, $1 \leq j \leq i-1$, are already known before the allocation of $c_{l_i}(t)$ and $x_{l_i}(t)$. Let $k \in S_c$ be a channel. We now consider the situation that will arise if we allocate channel k to link l_i. Note that if we allocate $c_{l_i}(t) = k$ then the powers allocated to all the links in $S_k^i(t) = \{j \mid c_{l_j}(t) = k, 1 \leq j \leq i\}$ may need to be modified while others will remain intact. Let $x_{k,l_1}(t), x_{k,l_2}(t), \cdots, x_{k,l_i}(t)$ be the powers that will be *temporarily* allocated to the links $l_1, l_2, \cdots, l_i$ and $P_k^i(t) = \sum_{j=1}^{i} x_{k,l_j}(t)$ if we set $c_{l_i}(t) = k$. Let $y_k^i(t)$ be the total number of distinct channels used for the links $l_1, l_2, \cdots, l_i$ if we set $c_{l_i}(t) = k$. For link l_i, we first set $c_{l_i}(t) = k$ and $x_{k,l_j}(t) = x_{l_j}(t)$ for $j \in \{1, 2, \cdots, i\} \setminus S_k^i(t)$. Then for link l_i, we solve the linear programming (LP) problem stated below to get $P_k^i(t)$ and corresponding $x_{k,l_j}(t)$ where $j \in S_k^i(t)$.

$$\textbf{LP:}\ \textit{minimize} \sum_{j \in S_k^i} x_{k,l_j}(t) \tag{5.16}$$

$$\textit{Subject to}$$

$$G_{l_j l_j}(t) x_{k,l_j}(t) \geq \gamma_{l_j}(t)(\sigma^2 + \sum_{m \in S_k^i(t) \setminus \{j\}} x_{k,l_m}(t) G_{l_j l_m}(t))\ \forall j \in S_i^k(t) \tag{5.17}$$

$$x_{k,l_j}(t) \leq \eta_{l_j}(t)\ \forall j \in S_i^k(t) \tag{5.18}$$

After computing $P_k^i(t)$ for all $k \in \{1, 2, \cdots, y_{c_{l_{i-1}}(t)}^{i-1}(t) + 1\}$, we find the channel k' such that

$$y_{k'}^i(t) + \alpha P_{k'}^i(t) = min\{y_k^i(t) + \alpha P_k^i(t) \mid k \in \{1, 2, \cdots, y_{c_{l_{i-1}}(t)}^{i-1}(t) + 1\}\}. \tag{5.19}$$

If multiple k' exists then we choose the minimum one. If k' exists then we reset $c_{l_i}(t) = k'$, $x_{l_j}(t) = x_{k',l_j}(t)$ for all $j \in S_{k'}^i$, else we mark link l_i as un-allocated. GCPA is formally stated in Algorithm 11.

Complexity of GCPA: If $LP_t(n,m)$ and $LP_s(n,m)$ are the time and space complexities of the **LP** running on n variables and m constraints and $\zeta = \max_{k=1}^{n} |\{l_i \mid c_{l_i}(t) = k\}|$ then the time and space complexities of GCPA are $O(n^2 \times LP_t(\zeta,\zeta))$ and $O(n^2 + LP_s(\zeta,\zeta))$ respectively.

Algorithm 11: *Greedy Channel and Power Allocation (GCPA) Algorithm*

```
Input: $(\gamma_{l_i}(t))$, $\sigma^2$, $(G_{l_il_j}(t))$, $S_{CL}$, $S_{DL}$, $S_c$, $(\eta_{l_i}(t))$, $\alpha$ and an order of links
       $S = (l_1, l_2, \cdots, l_n)$
Output: $C$,$X$
$C = (\emptyset)$;
$X = (0)$;
for $i = 1$ to $n$ do
    Calculate $P^i_k(t)$ for $\forall k \in S_c$ by solving the LP;
    Find $y^i_k(t)$ for $\forall k \in S_c$;
    Find $k'$ such that
      $y^i_{k'}(t) + \alpha P^i_{k'}(t) = min\{y^i_k(t) + \alpha P^i_k(t) \mid k \in \{1, 2, \cdots y^{i-1}_{c_{l_{i-1}}(t)} + 1\}\}$;
    if such $k'$ exists then
        Reset $c_{l_i}(t) = k'$ and update the corresponding $x_{l_j}(t)$s for all $j \in S^i_{k'}(t)$;
    else
        Keep $l_i$ as un-allocated;
return $C$,$X$;
```

Remark 7 *It is evident that though in JPCAP formulation we assume cost $f(t) = Y(t) + \alpha P(t)$ but it could essentially be any function $f(Y(t), P(t))$ which is increasing function of both $Y(t)$ and $P(t)$. In that case, Equation* (5.19) *needs to be replaced by*

$$f(y^i_{k'}(t), P^i_{k'}(t)) = min\{f(y^i_k(t), P^i_k(t)) \mid k \in \{1, 2, \cdots, y^{i-1}_{c_{l_{i-1}}(t)} + 1\}\}. \tag{5.20}$$

If instead of Equation (5.19) *we use Equation* (5.20) *in GCPA, all the results presented later in Subsections 5.4.1-5.5.4 will also remain valid.*

5.4.1 Analysis of GCPA

It is evident that GCPA takes an order S as input and produces $C(t)$ and $X(t)$ as output. Hence the cost $f(t) = Y(t) + \alpha P(t)$ is entirely depends on the order it chooses. We now show that there is an order for which GCPA produces optimum result. To prove this, we first introduce the concept of pseudo-vertex as defined below.

Definition 5 (Pseudo-vertex) *Let $C(t)$ be a channel vector having m distinct channels in it. Given $C(t)$, a pseudo-vertex $V_k = \{j : c(l_j) = k, 1 \leq j \leq n\}$ is defined as the set of all links which have been allocated with channel k, where $1 \leq k \leq m$.*

Theorem 5.4.1 *There is an order of links for which GCPA produces optimum solution.*

Proof: Let $C_o(t) = (c_{o,l_i}(t))$ and $X_o(t) = (x_{o,l_i}(t))$ be the channel and power vector of an optimal solution. Let $V_1, V_2, \cdots, V_m$ be the m pseudo-vertices obtained from C_o, where $n_1 = |V_1|, n_2 = |V_2|, \cdots, n_m = |V_m|$ and $n = n_1 + n_2 + \cdots + n_m$. Let $l^k_1, l^k_2, \cdots, l^k_{n_k}$ be the links in V_k, where $1 \leq k \leq m$. We now build an order $S = (l^1_1, l^1_2, \cdots, l^1_{n_1}, l^2_1, l^2_2, \cdots, l^2_{n_2}, \cdots, l^m_1, l^m_2, \cdots, l^m_{n_m})$, where the links in V_1 appear first, then appear the links of V_2 and so on, and finally appear the links of V_m. Links in the same pseudo-vertex may appear in arbitrary order. Our claim is that if we assign channels to the links following S, every link belongs to V_k will get channel k.

Clearly if we assign channels following S then the first link l^1_1 of V_1 will get channel 1. Let $l^1_i \in V_1$ be the first link of V_1 which did not get channel 1. If channel 1 is not assigned to l^1_i, then the power of l^1_i will be independent of the links $l^1_1, l^1_2, \cdots, l^1_{i-1}$ all of which have been assigned with channel 1. Let B be the power of link l^1_i when channel 1 is not assigned to link l^1_i. Let $A(k, \{l^1_j\})$, where $j \leq k$, be the total power of links in $\{l^1_1, l^1_2, \cdots, l^1_k\} \setminus \{l^1_j\}$ obtained by solving the LP, assuming all of them have been assigned with channel 1. Clearly GCPA will not assign channel 1 to link l_i^1 only if

$$\begin{aligned} & 2+\alpha(A(i,\{l_i^1\})+B) < 1+\alpha A(i,\emptyset) \\ \implies & 1/\alpha + B < (A(i,\emptyset) - A(i,\{l_i^1\})). \end{aligned} \tag{5.21}$$

It is evident that $A(k,\emptyset) - A(k,\{l_j^1\})$ is an increasing function of the power allocated to link l_j^1 assuming its channel as 1. This is because the power assigned to link l_i^1 will interfere with all other links assigned with channel 1 which in turn leads to increase their powers.

Now consider the optimal solution. If channel $m+1$ is assigned to link l_i^1 instead of channel 1 then the power of l_i^1 will be independent of the links $V_1 \setminus \{l_i^1\}$. In that case, the power of link l_i^1 will be same as B. Since $\{l_1^1, l_2^1, \cdots, l_i^1\} \subseteq V_1$, $x_o(l_i^1)$ must be higher than the power of l_i^1 obtained by solving the LP assuming all the links $l_1^1, l_2^1, \cdots, l_i^1$ are assigned with channel 1. As $A(k,\emptyset) - A(k,\{l_j^1\})$ is an increasing function of the power allocated to link l_j^1 assuming its channel as 1, we get

$$A(n_1,\emptyset) - A(n_1,\{l_i^1\}) \geq A(i,\emptyset) - A(i,\{l_i^1\}). \tag{5.22}$$

From Equations (5.22) and (5.21) it follows that channel $m+1$ will be a better choice of link l_i^1 in the optimal solution. A contradiction! This implies GCPA will assign all links of V_1 with channel 1.

Similarly we can show that if GCPA finds it beneficial to assign channel 1 to link l_1^2 then in optimal solution also it would have been beneficial to assign channel 1 to link l_1^2 and thereby reaching a contradiction. By this way we can show that GCPA will assign channel k to all links of V_k, $1 \leq k \leq m$.

Note that GCPA finally assigns power to the links using the LP solution when the last link of each pseudo-vertex is considered. Hence the power assignment will be the same as the optimum one. Hence the proof.

So far we know that the performance of GCPA depends on the order it chooses.

Also there exists an order on which GCPA produces optimal results. Now to compare between a pair of orders, we introduce the notion of equivalent order in Definition 6.

Definition 6 (Equivalent order) *Two orders S_1 and S_2 are said to be equivalent if upon applying GCPA on them they produce channel vectors $C_1(t)$ and $C_2(t)$ such that $C_1(t) = C_2(t)$.*

Let S be an order and upon applying GCPA on S we get channel vector C with pseudo-vertices $V_1, V_2, \cdots V_m$. For a particular permutation $\pi = (V_{c_1}, V_{c_2}, \cdots, V_{c_m})$ of $V_1, V_2, \cdots V_m$, let $L(\pi)$ be the set of all orders generated by permuting links within each pseudo-vertex while keeping the order of the pseudo-vertices unchanged. Let $\Pi(S)$ be the set of all $m!$ permutations of $V_1, V_2, \cdots V_m$. If $\pi_0(S) = (V_1, V_2, \cdots V_m)$ then Theorem 5.4.2 implies that all orders belongs to $L(\pi_0(S))$ are equivalent to S.

Theorem 5.4.2 *There are $\Omega((\frac{n}{em})^n)$ many orders which are equivalent to S.*

Proof: Consider the case where GCPA is applied on an order $S' \in L(\pi_0(S))$. In this case, similar to the proof of Theorem 5.4.1, we can show that all links in V_k will get channel k, where $1 \leq k \leq m$. Again $|L(\pi_0(S))| = (|V_1|)!(|V_2|)! \cdots, (|V_m|)!) = \Omega(((\frac{n}{m})!)^m) = \Omega((\frac{n}{em})^n)$ using Stirling's approximation. Hence the proof.

If total number of pseudo-vertices in optimum order is m_o then from Theorems 5.4.1 and 5.4.2 following corollaries are immediate.

Corollary 1 *Total number of optimum orders is*

$$\Omega((\frac{n}{em_o})^n).$$

Corollary 2 *Probability that an order is optimum is*

$$p = \Omega(\frac{1}{n!}(\frac{n}{em_o})^n) = \Omega(\frac{1}{m_o^n}).$$

Let S_1 and S_2 be two orders of links and upon applying GCPA on them we get channel vectors C_1 and C_2 with pseudo-vertices $V_1^1, V_2^1, \cdots, V_{m_1}^1$ and $V_1^2, V_2^2, \cdots, V_{m_2}^2$ respectively. Now without loss of generality, in the following lemma, we will prove that no two pseudo-vertices in $V_1^1, V_2^1, \cdots, V_{m_1}^1$ can be merged into a pseudo-vertex in $V_1^2, V_2^2, \cdots, V_{m_2}^2$.

Lemma 5.4.1 *$\nexists\, V_x^1, V_y^1$ and V_z^2 such that $V_x^1 \bigcup V_y^1 \subseteq V_z^2$ where $x \neq y$.*

Proof: If possible let $\exists V_x^1, V_y^1$ and V_z^2 such that $V_x^1 \bigcup V_y^1 \subseteq V_z^2$. As V_z^2 is build using GCPA then $\exists V_t^1$ such that $V_x^1 \bigcup V_y^1 \subseteq V_t^1$. Since GCPA upon applying on S_1 puts two different channel to the links of V_x^1 and V_y^1, this is impossible. Hence the proof.

We now consider two different permutations π_1, π_2 of $V_1, V_2, \cdots V_m$. If we apply GCPA on $S_1 \in L(\pi_1)$ and $S_2 \in L(\pi_2)$ we get channel vectors C_1 and C_2 with pseudo-vertices $V_1^1, V_2^1, \cdots, V_{m_1}^1$ and $V_1^2, V_2^2, \cdots, V_{m_2}^2$ respectively. It is evident that when we choose $\pi_1 \neq \pi_0(S)$ then pseudo-vertices $V_1, V_2, \cdots V_m$ may split and merge with other pseudo-vertices to form a new set of pseudo-vertices $V_1^1, V_2^1, \cdots, V_{m_1}^1$. Since by split and merge the total number of pseudo-vertices can only decrease, hence $m_1 \leq m_0$. Consider that $S_1, S_1' \in L(\pi_1)$, $S_1 \neq S_1'$ then upon applying GCPA our resulted set of pseudo-vertices corresponding to S_1 and S_1' may be different. Hence $L(\pi_1)$ may not be an equivalent set. But what about $\pi_3 = (V_1^1, V_2^1, \cdots, V_{m_1}^1)$ and $\pi_4 = (V_1^2, V_2^2, \cdots, V_{m_2}^2)$? Clearly $L(\pi_3)$ and $L(\pi_4)$ are two equivalent sets, each with exponential number of elements, according to Theorem 5.4.2. Now we would like to show that these two equivalent sets are disjoint. That is, $L(\pi_3) \bigcap L(\pi_4) = \emptyset$. In that case by evaluating $S_1 \in L(\pi_1)$ and $S_2 \in L(\pi_2)$, $S_1 \neq S_2$, we can eventually search element from two disjoint equivalent sets each with exponential number of orders. We formally prove $L(\pi_3) \bigcap L(\pi_4) = \emptyset$ in Theorem 5.4.3. In order to prove $L(\pi_3) \bigcap L(\pi_4) = \emptyset$, we first show that there exists a position x in π_3 and π_4 such that $V_x^1 \neq V_x^2$, in the following lemma.

Lemma 5.4.2 *$\exists x$, such that $V_x^1 \neq V_x^2$.*

Proof: Let $\pi_1 = (V_{a_1}, V_{a_2}, \cdots V_{a_m})$ and $\pi_2 = (V_{b_1}, V_{b_2}, \cdots V_{b_m})$. Since $\pi_1 \neq \pi_2$, let i be the first position in π_1 such that $a_i \neq b_i$. Since according to Lemma 5.4.2, $\nexists V^1_{x_1}$ and $V^2_{x_2}$ such that $V_{a_i} \bigcup V_{b_i} \subseteq V^1_{x_1}$ or $V_{a_i} \bigcup V_{b_i} \subseteq V^2_{x_2}$, $\exists e_1 \in V_{a_i}$ and $e_2 \in V_{b_i}$ such that e_1 appears before e_2 in any order in $L(\pi_3)$ and e_1 appears after e_2 in any order in $L(\pi_4)$. Hence the proof.

Theorem 5.4.3 $L(\pi_3) \bigcap L(\pi_4) = \emptyset$.

Proof: If possible, assume $L(\pi_3) \bigcap L(\pi_4) \neq \emptyset$. Since $L(\pi_3) \bigcap L(\pi_4) \neq \emptyset$, all links in V^2_1 must appear in V^1_1 and all links in V^1_1 must appear in V^2_1. That is, $V^2_1 \subseteq V^1_1$ and $V^1_1 \subseteq V^2_1$ implying $V^2_1 = V^1_1$. Similarly, we can show that $V^2_2 = V^1_2$, $V^2_3 = V^1_3$ and so on. Hence $m_1 = m_2 = m$ and $V^2_x = V^1_x$, $1 \leq x \leq m$. But from Lemma 5.4.1, we know that $\exists x$ such that $V^1_x \neq V^2_x$. Contradiction! Hence the proof.

5.5 Randomized Algorithm

In this section we first propose an incremental algorithm (IA) to minimize the cost and then propose a randomized joint channel and power allocation algorithm (RJCPA) which uses IA with parameter ρ to further minimize the cost. Next we present the analysis of RJCPA. Since the optimum hitting probability of RJCPA is a function of ρ^2, we then compute the expected value of ρ^2 to find the optimum.

5.5.1 Incremental Algorithm (IA)

We now propose our incremental algorithm (IA) which starts with an order S, applies GCPA on S and constructs C and X. It then considers a random permutation π of pseudo-vertices of C and applies GCPA on a random order $S_r \in L(\pi)$. This step is repeated for ρ times where ρ is an input parameter. If total cost improves, we set $S = S_r$ and repeat the process. Formally IA is presented in Algorithm 12. It is evident that IA with an order S searches $\Omega(\rho(\frac{n}{em})^n)$ orders.

Complexity of IA: The time complexity of IA is ρ times the time complexity of GCPA and space complexity of IA is same as the space complexity of GCPA.

Algorithm 12: *Incremental Algorithm (IA)*

```
Input: $(\gamma_{l_i}(t))$, $\sigma^2$, $(G_{l_il_j}(t))$, $S_{CL}$, $S_{DL}$, $S_c$, $(\eta_{l_i}(t))$, $\alpha$, an order $S$, $\rho$
Output: $C_{min}$, $X_{min}$
Set $f_{min}(t) = \infty$;
$C(t)$, $X(t)$ = GCPA($(\gamma_{l_i}(t))$, $\sigma^2$, $(G_{l_il_j}(t))$, $S_{CL}$, $S_{DL}$, $S_c$, $(\eta_{l_i}(t))$, $\alpha$, $S$);
Calculate cost $f(t)$;
Set $C_{min}(t) = C(t)$;
Set $X_{min}(t) = X(t)$;
Set $f_{min}(t) = f(t)$;
Generate pseudo-vertices $V_1, V_2, \cdots, V_m$ of $C(t)$;
Set $n_k = |V_k|$ for $1 < k < m$;
for $r = 1, 2, \cdots \rho$ do
    Consider an arbitrary permutation $\pi$ of $V_1, V_2, \cdots, V_m$;
    Generate random order $S_r \in L(\pi)$;
    $C(t)$, $X(t)$ = GCPA($(\gamma_{l_i}(t))$, $\sigma^2$, $(G_{l_il_j}(t))$, $S_{CL}$, $S_{DL}$, $S_c$, $(\eta_{l_i}(t))$, $\alpha$, $S_r$);
    Calculate cost $f(t)$;
    if $f(t) < f_{min}(t)$ then
        Set $C_{min}(t) = C(t)$;
        Set $X_{min}(t) = X(t)$;
        Set $f_{min}(t) = f(t)$;
        Set $S = S_r$;
        Generate pseudo-vertices $V_1, V_2, \cdots, V_m$ of $C(t)$;
        Set $n_k = |V_k|$ for $1 < k < m$;
        Restart loop with $r = 1$;
return $C_{min}(t)$, $X_{min}(t)$;
```

5.5.2 Randomized Joint Channel and Power Allocation (RJCPA) Algorithm

It is evident that the performance of IA depends on the initial order S. With initial order S, the set of all orders and their equivalent orders that could eventually be searched with positive probability by IA has size

$$|\bigcup_{\pi\in\Pi(S)}\bigcup_{S'\in L(\pi)} L(\pi_o(S'))| = \Omega\left(m!(\frac{n}{em})^n\right), \tag{5.23}$$

which could be less than $n!$. That is, if IA starts with S then there may exist some orders which are searched with 0 probability. To search each of the $n!$ orders with a positive probability, we introduce the randomized joint channel and power allocation (RJCPA) algorithm. RJCPA chooses some ρ random orders, applies IA on each of them and reports the best result. RJCPA is formally presented in Algorithm 13.

Algorithm 13: *Randomize Joint Channel and Power Allocation (RJCPA) Algorithm*

```
Input: $(\gamma_{l_i}(t))$, $\sigma^2$, $(G_{l_i l_j}(t))$, $S_{CL}$, $S_{DL}$, $S_c$, $(\eta_{l_i}(t))$, $\alpha$,$\rho$
Output: $C_{min}(t)$, $X_{min}(t)$
Set $f_{min}(t) = \infty$;
for $r = 1$ to $\rho$ do
    Generate a random order $S$ of the links;
    $C(t)$, $X(t)$ = IA(($\gamma_{l_i}(t)$), $\sigma^2$, ($G_{l_i,l_j}(t)$), $S_{CL}$, $S_{DL}$, $S_c$, ($\eta_{l_i}(t)$), $\alpha$, $S$,$\rho$);
    Calculate cost $f_S$ generated by GCPA on order $S$;
    if $f_S(t) < f_{min}(t)$ then
        $f_{min}(t) = f_S(t)$;
        $C_{min}(t) = C(t)$;
        $X_{min}(t) = X(t)$;
        Restart loop with $r = 1$;
return $C_{min}(t)$, $X_{min}(t)$;
```

Complexity of RJCPA: The time complexity of RJCPA is ρ times the time complexity of IA and space complexity of RJCPA is same as the space complexity of IA.

5.5.3 Analysis of RJCPA

Let $SO(S_i)$ be the set of *exactly* those orders and their equivalent orders that would be searched by IA with initial order S_i. Clearly

$$|SO(S_i)| = \Omega(\rho(\frac{n}{em})^n). \tag{5.24}$$

Definition 7 *If $x \in \{0, 1, \cdots\}$, then $H(x) \in (0, \infty)$ is defined as $\frac{H(x)}{x} \to 0$ when $x \to 0$ or $x \to \infty$.*

Lemma 5.5.1 *If $|SO(S_i)| = H(\frac{\sqrt{n!}}{\rho})$ $\forall i \in \{1, 2, \cdots, \rho\}$, where either $\rho = o(\sqrt{n!})$ or $\sqrt{n!} = o(\rho)$, then*

$$\lim_{n\to\infty} \mathbb{P}(\forall i, j \in \{1, 2, \cdots, \rho\},\ i < j,\ SO(S_i) \bigcap SO(S_j) = \emptyset) = 1.$$

Proof: It is evident that for an order S,

$$\mathbb{P}(S \in SO(S_i)) = \frac{|SO(S_i)|}{n!}. \tag{5.25}$$

Now,

$$\begin{aligned}
& \mathbb{P}(\forall i, j \in \{1, 2, \cdots, \rho\},\ i < j,\ SP(S_i) \bigcap SO(S_j) = \emptyset) \\
= & \prod_{i=1}^{\rho} \prod_{i<j} \mathbb{P}(SO(S_i) \bigcap SO(S_j) = \emptyset) \\
= & \Big(1 - \mathbb{P}(SO(S_i) \bigcap SO(S_j) \neq \emptyset)\Big)^{\binom{\rho}{2}} \\
= & (1 - \mathbb{P}\left(\bigcup_{\forall S} (S \in SO(S_i) \ \&\ S \in SO(S_j))\right))^{\binom{\rho}{2}} \\
\geq & (1 - \sum_{\forall S} \mathbb{P}(S \in SO(S_i) \ \&\ S \in SO(S_j)))^{\binom{\rho}{2}} \qquad (5.26)
\end{aligned}$$

(because for any finite collection of events $\{A_i \mid \forall i\}$,

$\mathbb{P}(\bigcup_{\forall i} A_i) \leq \sum_{\forall i} \mathbb{P}(A_i))$.

Also the events that an order belongs to $SO(S_i)$ and an order belongs to $SO(S_j)$ are independent to each other. Hence from Equation (5.26) we get that

$$\begin{aligned}
& \mathbb{P}(\forall i,j \in \{1,2,\cdots,\rho\},\ i<j,\ SP(S_i)\bigcap SO(S_j)=\emptyset) \\
\geq\ & (1-n!\times\mathbb{P}(S\in SO(S_i))\times\mathbb{P}(S\in SO(S_j)))^{\binom{\rho}{2}} \\
=\ & (1-n!\times\frac{|SO(S_i)|}{n!}\times\frac{|SO(S_j)|}{n!})^{\binom{\rho}{2}} \\
=\ & (1-\frac{|SO(S_i)|\times|SO(S_j)|}{n!})^{\binom{\rho}{2}}.
\end{aligned} \tag{5.27}$$

Again as $|SO(S_i)| = H(\frac{\sqrt{n!}}{\rho}), \forall i \in \{1,2,\cdots,\rho\}$, where either $\rho = o(\sqrt{n!})$ or $\sqrt{n!} = o(\rho)$, we get from Equation (5.27) that,

$$\begin{aligned}
& \lim_{n\to\infty}\mathbb{P}(\forall i,j \in \{1,2,\cdots,\rho\},\ i<j,\ SO(S_i)\bigcap SO(S_j)=\emptyset) \\
& = \lim_{n\to\infty}(1-\frac{H(\frac{\sqrt{n!}}{\rho^2})}{\frac{\sqrt{n!}}{\rho^2}}\times\frac{1}{\rho^2})^{\binom{\rho}{2}} = 1.
\end{aligned} \tag{5.28}$$

Hence the proof.

Using Lemma 5.5.1 we will calculate the total number of orders RJCPA eventually searches in Theorem 5.5.1.

Theorem 5.5.1 *Let $\tau(m)$ be the total number of orders eventually searched by RJCPA. Then*

$$\tau(m) = \begin{cases} \Omega(\rho^2(\frac{n}{em})^n) & \textit{if } \rho = o(\sqrt{n!}) \textit{ or } \sqrt{n!} = o(\rho)\ \&\ m > n^{\frac{1-\frac{1}{\log_e(n)}}{2}} + \rho^{\frac{2}{n}} \\ \Omega(\rho(\frac{n}{em})^n) & \textit{else} \end{cases}$$

Again if $\rho = o(\sqrt{n!})$ or $\sqrt{n!} = o(\rho)$ and $m > n^{\frac{1-\frac{1}{\log_e(n)}}{2}} + \rho^{\frac{2}{n}}$ then $\lim_{m\to n}\tau(m) = \rho^2$ else $\lim_{m\to 1}\tau(m) = n!$.

Proof: If $|SO(S_i)| = H(\frac{\sqrt{n!}}{\rho})$, where either $\rho = o(\sqrt{n!})$ or $\sqrt{n!} = o(\rho)$, from

Equation 5.24 we get that,

$$
\begin{aligned}
& \Omega\left(\rho(\frac{n}{em})^n\right) = |SO(S_i)| = H(\frac{\sqrt{n!}}{\rho}) \\
\Longrightarrow \quad & \lim_{n\to\infty} \frac{\rho^2(\frac{n}{em})^n}{\sqrt{n!}} = 0 \\
\Longrightarrow \quad & \lim_{n\to\infty} \frac{\rho^2}{m^n}(\frac{n}{e})^{n/2} = 0. \qquad (5.29)
\end{aligned}
$$

Putting $m = n^{\frac{1}{x}}$ into Equation (5.29) we get that,

$$
\begin{aligned}
& \lim_{n\to\infty} \frac{\rho^2}{n^{\frac{n}{x}}}(\frac{n}{e})^{n/2} = 0 \\
\Longrightarrow \quad & \lim_{n\to\infty} n^{\frac{n}{2}(1-\frac{1}{\log_e(n)})+\frac{\log_e(\rho^2)}{\log_e(n)}-\frac{n}{x}} = 0 \\
\Longrightarrow \quad & \lim_{n\to\infty} n^{\frac{n}{2}(1-\frac{1}{\log_e(n)})+\frac{\log_e(\rho^2)}{\log_e(n)}-\frac{n\log_e(m)}{\log_e(n)}} = 0. \qquad (5.30)
\end{aligned}
$$

Note that in Equation (5.30) if power of n is < 0 then the limit in left hand side would converge to 0 when $n \to \infty$. Hence from Equation (5.30) we get that

$$
\begin{aligned}
& \frac{n}{2}(1 - \frac{1}{\log_e(n)}) + \frac{\log_e(\rho^2)}{\log_e(n)} < \frac{n\log_e(m)}{\log_e(n)} \\
\Longrightarrow \quad & m > n^{\frac{1-\frac{1}{\log_e(n)}}{2}} + \rho^{\frac{2}{n}} \qquad (5.31)
\end{aligned}
$$

where either $\rho = o(\sqrt{n!})$ or $\sqrt{n!} = o(\rho)$, is the sufficient condition for $SO(S_i) \bigcap SO(S_j) = \emptyset, \forall i, j \in \{1, 2, \cdots, \rho\}, i \neq j$, when $n \to \infty$. As $|SO(S_i)| = \Omega(\rho(\frac{n}{em})^n)$ hence we get that $\tau(m) = \Omega(\rho^2(\frac{n}{em})^n)$ with

$$
\lim_{m\to n} \tau(m) = \rho^2,
$$

(because with $m = n$ each of the n channels is allocated to a different link and hence $|SP(S_i)| = \rho$ and $\tau(n) = \rho^2$).

Else in worst case all of the $SP(S_i)$, where $1 \leq i \leq \rho$, may be exactly equal, hence $\tau(m) = \Omega(\rho \times (\dfrac{n}{em})^n)$, with $\lim\limits_{m \to 1} \tau(m) = n!$ (because $\tau(m) \leq n!$), when $n \to \infty$. Hence the proof.

Remark 8 *From Theorem 5.5.1 we get that, if $\rho = o(\sqrt{n!})$ or $\sqrt{n!} = o(\rho)$ and m is big ($m > n^{\frac{1-\frac{1}{\log_e(n)}}{2}} + \rho^{\frac{2}{n}}$) then in RJCPA we essentially search disjoint $SQ(S_i)$ for different S_i's. Hence the total number of orders searched is $\Omega(\rho(\frac{n}{em})^n)$. On the other hand if m is small there might be overlapping between different $SO(S_i)$'s. But in this case, as m gets smaller $SO(S_i)$ gets bigger. So in either case, RJCPA searches orders very efficiently. Moreover, RJCPA always produces the optimum result for the limiting cases when $m = 1$ or $m = n$.*

Remark 9 *From Equation* (5.31) *we get that, if $\rho = o(\sqrt{n!})$ or $\sqrt{n!} = o(\rho)$ then with increasing ρ^2 the lower bound of necessary value of m increases. Since the maximum value of m is n, if*

$$n^{\frac{1-\frac{1}{\log_e(n)}}{2}} + \rho^{\frac{2}{n}} \geq n$$
$$\implies \rho \geq (n - n^{\frac{1-\frac{1}{\log_e(n)}}{2}})^{\frac{n}{2}},$$

$\exists i, j \in \{1, 2, \cdots, \rho\}$, $i \neq j$, such that $SO(S_i) \bigcap SO(S_j) \neq \emptyset$ even if $n \to \infty$. Thus with higher value of ρ we need higher value of m to keep each $SO(S_i)$ disjoint.

It is evident from Theorem 5.5.1 that total number of orders eventually searched by RJCPA is $\Omega(\rho^2(\frac{n}{em})^n)$, which is a function of ρ^2. Hence in the next Section we will calculate expected value of ρ^2 to find the optimum.

5.5.4 Expected value of ρ^2

Let's define a step as applying GCPA on an order and obtaining C and X. Clearly IA and RJCPA have $\Omega(\rho)$ and $\Omega(\rho^2)$ steps. Hence time complexity of RJCPA to hit the optimum is a function of ρ^2. We now calculate the expected value of ρ^2. Let m' be the maximum number of distinct channels in a step and n be the total number of links. It is evident from Theorem 5.4.2 that there are $\Omega((\frac{n}{em'})^n)$ orders equivalent to S. Note that value of ρ^2 to hit optimum follows a Geometric distribution with mean $\frac{1}{\eta}$ where

$$\eta \quad = \quad \begin{cases} 1 & \text{If } m'm_o \leq \frac{n}{e} \\ \Omega((\frac{n}{em'})^n \times \frac{1}{m_o^n}) = \Omega((\frac{n}{em'm_o})^n) & \text{Otherwise} \end{cases}.$$

This fact implies Theorem 5.5.2.

Theorem 5.5.2 *Expected number of steps RJCPA takes to hit optimum is*

$$\mathbb{E}[\rho^2] = \frac{1}{\eta} = \begin{cases} 1 & \textit{if } m'm_o \leq \frac{n}{e} \\ O((\frac{em'm_o}{n})^n) & \textit{Otherwise} \end{cases}. \tag{5.32}$$

From Theorem 5.5.2 and the fact that $(1-x)^{\frac{1}{x}} \leq e^{-1}\ \forall x \in (0,1)$, Corollary 3 is immediate.

Corollary 3 *If $m'm_o \leq \frac{n}{e}$, expected number of steps to reach optimum is $\mathbb{E}[\rho^2] = O(1)$. If $m'm_o > \frac{n}{e}$, RJCPA hits the optimum in $\rho^2 = \frac{\epsilon}{\eta} = \epsilon\mathbb{E}[\rho^2]$ steps with probability*

$$\begin{aligned} &= \quad (1-(1-\eta)^{\rho^2}) \\ &= \quad (1-(1-\eta)^{\frac{\epsilon}{\eta}}) \\ &\geq \quad (1-e^{-\epsilon}). \end{aligned} \tag{5.33}$$

Hence if $m'm_o \leq \frac{n}{e}$ we can solve JPCAP in expected polynomial time, else we can solve it in slowly growing expected exponential time with very high probability. It is evident from Chebyshev's in-equality that the distribution of ρ^2 to find the optimum is highly concentrated around $\mathbb{E}[\rho^2] = \frac{1}{\eta}$. That is

$$\mathbb{P}\left(|\rho^2 - \mathbb{E}[\rho^2]| \geq \epsilon\sigma[\rho^2]\right) \leq \frac{1}{\epsilon^2}, \tag{5.34}$$

where standard deviation of ρ^2 is

$$\sigma[\rho^2] = \sqrt{\mathbb{E}[(\rho^2 - \mathbb{E}[\rho^2])^2]} = \sqrt{\frac{1-\eta}{\eta^2}}.$$

5.6 Expected Cost and Energy Efficiency

We now analyze the expected cost and energy efficiency (EE) produced by RJCPA. For the analysis purpose we will first state some assumptions and notations. We assume that n links are situated inside a R radius cell with base station placed centered at it and $\kappa = \frac{|S_{DL}|}{n}$. Let $g = \mathbb{E}[G_{l_i l_i}(t)]$, where $l_i \in S_{DL} \bigcup S_{CL}$, and $G = \mathbb{E}[G_{l_i l_j}(t)]$, where $i \neq j$ and $l_i, l_j \in S_{DL} \bigcup S_{CL}$. Also $\Gamma = \mathbb{E}[\gamma_{l_i}(t)]$. Let $\mathbb{E}[Y(t)] = \overline{Y}$, $\mathbb{E}[P(t)] = \overline{P}$, $\mathbb{E}[x_{l_i}(t)] = \overline{x} = \frac{\overline{P}}{n}$, $\mathbb{E}[f(t)] = \overline{f}$ and $\mathbb{E}[\eta_{l_i}(t)] = \eta$. Let $EE(t)$ be the energy efficiency and $\mathbb{E}[EE(t)] = \overline{EE}$. We also assume that expected value of $\eta_{l_i}(t)$ over DL is $\eta_{DL}(t)$ and over CL is $\eta_{CL}(t)$. Let's also define $a(n) \sim_n b(n) \iff \lim_{n\to\infty} \dfrac{b(n)}{a(n)} = 1$. This section is organized into three subsections. In the first subsection we calculate the expected cost $\overline{f}$. In the second subsection we calculate the expected energy efficiency $\overline{EE}$. Since all expressions of the first two subsections are functions of g, G and η in the third subsection we calculate them.

5.6.1 Calculation of expected cost

We first calculate $\overline{Y}$ and $\overline{P}$ to calculate expected cost $\overline{f}$. Clearly, when $n \to \infty$ due to symmetry of the links, in expected case, inequalities in the LP calculating $x(l_i)$'s will turn out to be equality's and each channel will be assigned to equal number of links, hence

$$g\overline{x} = \Gamma(\sigma^2 + (\frac{n}{\overline{Y}} - 1)\overline{x}G) \implies \overline{P} = \Gamma(n\sigma^2 + (\frac{n}{\overline{Y}} - 1)\overline{P}G). \tag{5.35}$$

It is evident that

$$\overline{Y} \geq |S_{CL}| \quad = \quad (1-\kappa)n. \tag{5.36}$$

Note that one pseudo-vertex can at most have $\frac{n}{\overline{Y}}$ links only if

$$\alpha\frac{n}{\overline{Y}}\overline{x} \leq 1 \quad \implies \quad \alpha\overline{P} \leq \overline{Y}. \tag{5.37}$$

Again

$$\overline{x} \leq \eta \quad \implies \quad \overline{P} \leq n\eta. \tag{5.38}$$

Now from Equations (5.35), (5.37), (5.38) and (5.36) we get

$$\overline{P} \leq n \times \min(\frac{\sigma^2\alpha + G}{\alpha(\frac{g}{\Gamma} + G)}, \eta), \text{and} \tag{5.39}$$

$$\overline{Y} \geq n \times \max(\min(\frac{\sigma^2\alpha + G}{\frac{g}{\Gamma} + G}, \alpha\eta), 1-\kappa). \tag{5.40}$$

Since when $n \to \infty$ In-equations (5.39) and (5.40) becomes equations and hence we get

$$\overline{P} \quad \sim_n \quad n \times \min(\frac{\sigma^2 + \frac{G}{\alpha}}{\frac{g}{\Gamma} + G}, \eta), \tag{5.41}$$

$$\overline{Y} \quad \sim_n \quad n \times \max(\min(\frac{\sigma^2\alpha + G}{\frac{g}{\Gamma} + G}, \alpha\eta), (1-\kappa)), \text{and} \tag{5.42}$$

$$\overline{f} \sim_n n \times (\max(\min(\frac{\sigma^2\alpha + G}{\frac{g}{\Gamma} + G}, \alpha\eta), (1-\kappa)) + \min(\frac{\sigma^2\alpha + G}{\frac{g}{\Gamma} + G}, \alpha\eta)). \tag{5.43}$$

From Equations (5.42), (5.41) and (5.43) we get that $\overline{Y}$, $\overline{P}$ and $\overline{f}$ are increasing functions of n and η. Also $\overline{f}$ and $\overline{Y}$ are non-decreasing function of α and $\overline{P}$ is a non-increasing function of α. Finally $\overline{Y}$ is a non-increasing function of κ.

5.6.2 Calculation of expected energy efficiency

We now calculate the expression for expected energy efficiency $\overline{EE}$. It is evident from [63, 64, 10, 61, 62] that

$$EE = \frac{1}{\sum\limits_i x_{l_i}(t)} \sum_{c=1}^{n} \sum_{\substack{\forall i \text{ s.t.} \\ c_{l_i}(t)=c}} \log_e(1 + \frac{x_{l_i}(t) G_{l_i l_i}(t)}{\sigma^2 + \sum\limits_{\substack{j\,:\,j \neq i \\ \& \\ c_{l_i}(t)=c_{l_j}(t)}} x_{l_j}(t) G_{l_i l_j}(t)}).$$

Thus when $n \to \infty$ we get

$$\begin{aligned}
\overline{EE} &= \frac{n}{\overline{P}} \log_e(1 + \frac{\overline{x} g}{\sigma^2 + (\frac{n}{\overline{Y}} - 1)\overline{x} G}) \\
&\sim_n \frac{n}{\overline{P}} \log_e(1 + \frac{g}{\frac{n\sigma^2}{\overline{P}} + (\frac{n}{\overline{Y}} - 1) G}) \\
&\sim_n \frac{\alpha}{\min(\frac{\sigma^2\alpha + G}{\frac{g}{\Gamma}+G}, \alpha\eta)} \times \log_e(1 + g/(\frac{\sigma^2\alpha}{\min(\frac{\sigma^2\alpha+G}{\frac{g}{\Gamma}+G}, \alpha\eta)} + \\
&\quad (\frac{1}{\max(\min(\frac{\sigma^2\alpha+G}{\frac{g}{\Gamma}+G}, \alpha\eta), (1-\kappa))} - 1) G)).
\end{aligned} \tag{5.44}$$

Note that $\overline{EE}$ is a non-decreasing functions of α and η, and a non-increasing function of κ.

5.6.3 Calculations of expected value of parameters

Since all the above expressions are in terms of g, G and Δ we now calculate them. Let's assume users are placed uniformly inside the R radius circle. We also assume that r is the expected distance of transmitter and receiver of a DL. Then using the fact that $h^{slow}_{l_il_j}(t)$ and $h^{fast}_{l_il_j}(t)$ are random log-normally and exponentially distributed variable with mean 0, standard deviation 1 and mean 1 respectively, and $r_0 << r << R$ is the minimum distance between devices we get

$$\begin{aligned}
g &\sim_n \left(\kappa \int_{d_{l_il_i}=r}^{r_0} \frac{d_{l_il_i}^{-\beta}}{r} d(d_{l_il_i}) + (1-\kappa)\int_{d_{l_il_i}=R}^{r_0} \frac{d_{l_il_i}^{-\beta}}{R} d(d_{l_il_i})\right) \\
&\quad \times \mathbb{E}[h^{slow}_{l_il_j}(t)] \times \mathbb{E}[h^{fast}_{l_il_j}(t)] \\
&\sim_n \left(\frac{1}{\beta-1}(\frac{\kappa}{r}(\frac{1}{r_0^{\beta-1}} - \frac{1}{r^{\beta-1}}) + \frac{1-\kappa}{R}(\frac{1}{r_0^{\beta-1}} - \frac{1}{R^{\beta-1}}))\right) \\
&\quad \times \mathbb{E}[h^{slow}_{l_il_j}(t)] \times \mathbb{E}[h^{fast}_{l_il_j}(t)],
\end{aligned} \tag{5.45}$$

$$\begin{aligned}
G &\sim_n \frac{1}{(\beta-1)R}(\frac{1}{r_0^{\beta-1}} - \frac{1}{R^{\beta-1}}) \\
&\quad \times \mathbb{E}[h^{slow}_{l_il_j}(t)] \times \mathbb{E}[h^{fast}_{l_il_j}(t)],
\end{aligned} \tag{5.46}$$

$$\Delta \sim_n \kappa\Delta_{DL} + (1-\kappa)\Delta_{CL}. \tag{5.47}$$

It is evident from *weak law of large number* and *central limit theorem* that calculated values of $\overline{Y}$, $\overline{P}$, g and G over N number of experiments converge to the exact values of $\overline{Y}$, $\overline{P}$, g and G respectively, with their variances → 0, when $N \to \infty$.

Remark 10 *Since Equation* (5.42) *states that* $\frac{n}{\overline{Y}} = O(1)$, *the expected time and space complexities of RJCPA becomes* $O((\rho n)^2 \times LP_t(\frac{n}{\overline{Y}}, \frac{n}{\overline{Y}})) = O((\rho n)^2)$ *and* $O(n^2 + LP_s(\frac{n}{\overline{Y}}, \frac{n}{\overline{Y}})) = O(n^2)$ *respectively. Note that in expected case the time and space complexities of* ***LP*** *are* $O(1)$. *Thus by calling* ***LP*** *repeatedly we can solve the problem in relatively small that is* $O(n^2)$ *time.*

5.7 Simulation

This section comprises of three subsection. In the first subsection we elaborate the simulation environment. In the second subsection we compare RJCPA with two existing algorithms. In the third subsection we validate our analytical findings through simulation.

5.7.1 Simulation environment

The simulation environment is mostly similar to that of [64, 27, 66, 93]. We consider a 250 m radius circular cell with one base station placed in the center of it. We consider DL and CL are uniformly placed within the circular region. For each DL, transmitter and receiver are placed randomly within 1 m to 25 m away from each other. Minimum power allocated to a link is considered to be -60 dBm and the maximum power $\eta_i(t)$ is considered to be 24 dBm for CL and 21 dBm for DL. We consider a snapshot based approach, where users are considered to be static during execution time of the algorithm. For generating different snapshots, we considered that users move following random way-point mobility model with average velocity 20 m/s (i.e $\mathbf{v} = 40$). Snapshots are taken every 1 s interval. The important simulation parameters are summarized in Table 5.1. We have written our own C++ custom code and run them on a GNU 5.5.0 compiler on Intel core i7 Linux machine.

Parameters	Description
Cell radius	250 m
$\delta_i(t)$	24 dBm for CL & 21 dBm for DL
Minimum power allocated to a link	-60 dBm
σ^2	-174 dBm
$\gamma_i(t)$	25 dB
$h_{ij}^{slow}(t)$	log-normal distribution with mean 0 dB and standard deviation 8 dB
$h_{ij}^{fast}(t)$	exponential distribution with $\mu = 1$
Pathloss exponent β	4
Packet size	100 $byte$
Average speed of users	20 ms^{-1}
Bandwidth	5 MHz
$\|S_{CL}\|$	25
$\|S_{DL}\|$	$\in \{1, 2, \cdots, 75\}$

Table 5.1: Simulation Environment

5.7.2 Comparison with other algorithms

In this section through simulation we will compare our algorithm with that of the two-step solution proposed in [64, 10]. In [64], a matching based algorithms for energy efficiency (EE) maximization is presented. In the first step, the optimal power that is required to maximize EE for each DL-CL pair is found and then in the second step, it finds a stable matching of DL and CL using Gale-Shapley algorithm [65]. It is important to note that the performance of the second step is highly dependent on the performance of the first step. In [10] authors propose a two step approach for EE minimization. Given DL-CL pairing, they show that the energy minimization problem is conditionally convex. If the problem is convex, they solve it using standard techniques else they propose an iterative approach for it. To efficiently form DL-CL pair they then propose a random switch-based iterative (RSBI) algorithm. RSBI starts with a random DL-CL pairing and then taking random steps tries to minimize the energy consumption. It is evident that both approaches consider that only one DL can be paired with one CL. Hence in both [64, 10] $|S_{DL}| \leq |S_{CL}|$. This scenario is called *resource-abundant* scenario and

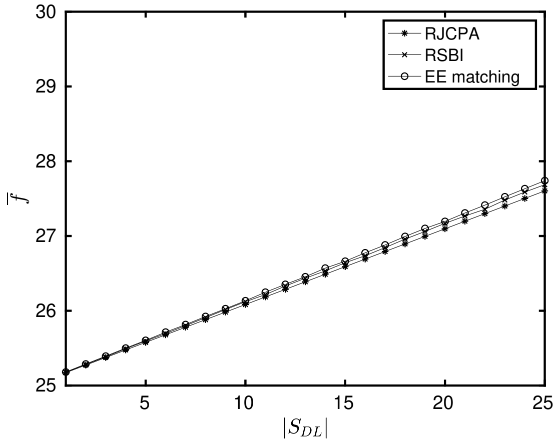

Figure 5.1: $\overline{f} = \overline{Y} + \alpha\overline{P}$ in dB vs $|S_{DL}|$

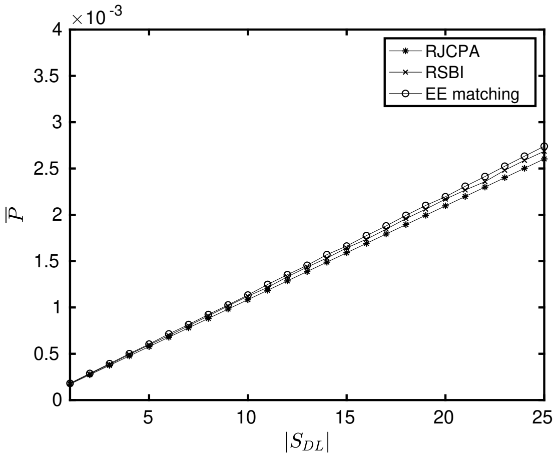

Figure 5.2: $\overline{P}$ in mJ vs $|S_{DL}|$

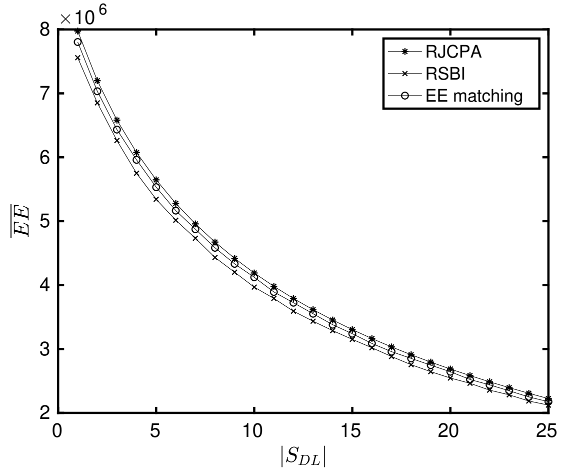

Figure 5.3: $\overline{EE}$ in $bitmJ^{-1}Hz^{-1}$ vs $|S_{DL}|$

considered by several authors [64, 94, 10]. In contrast to this, RJCPA has no such bound on $|S_{DL}|$ and $|S_{CL}|$. Also, RJCPA allows multiple DL to pair with a CL. In addition to that RJCPA deals with the channel and power allocation problems jointly and by making use of the GCPA it evaluates a large number of orders by searching only a few. There are also some cases for which RJCPA solve the JPCAP in expected polynomial time and for other cases it solves it in slowly growing exponential time with very high probability.

In Figures 5.1, 5.2 and 5.3 we compare the cost $\overline{f} = \overline{Y} + \alpha\overline{P}$, $\overline{P}$ in mJ and $\overline{EE}$ in

$bitmJ^{-1}Hz^{-1}$ obtained by RJCPA with that obtained by EE matching based [64] and RSBI [10] algorithms respectively, where $10\log_{10}(\alpha) = 30$, $|S_{CL}| = 25$ and $|S_{DL}| \in \{1, 2, \cdots, 25\}$. To make these algorithms comparable, we fix the number of channels available as $|S_c| = 25$ for all of them. We run our algorithm with $\rho = 10$ and report the average result of 10 runs. We observe that RJCPA produces lesser $\overline{f}$, $\overline{P}$ and higher $\overline{EE}$ than that of both EE matching based and RSBI algorithms. By efficiently searching the orders, RJCPA eventually reduces the total power requirement of the links and thus achieves higher $\overline{EE}$.

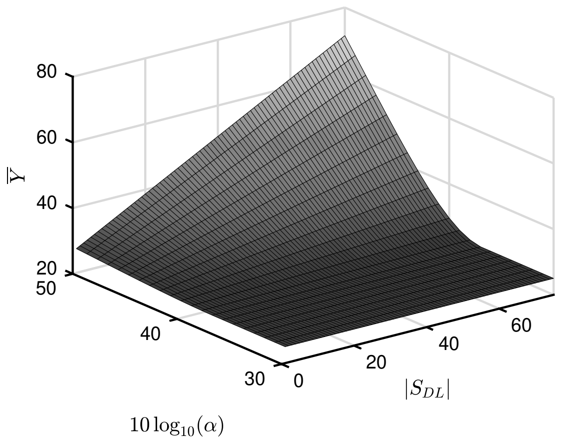

Figure 5.4: $\overline{Y}$ vs $10log_{10}(\alpha)$ and $|S_{DL}|$

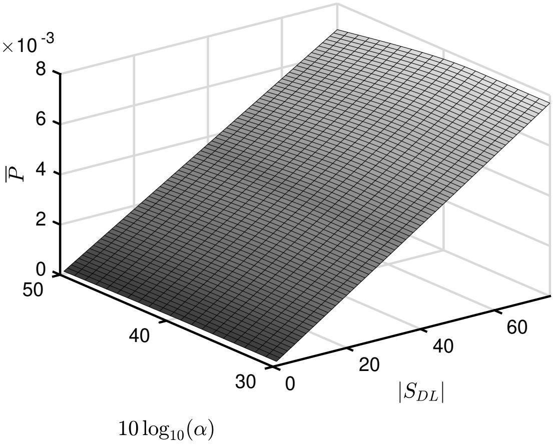

Figure 5.5: $\overline{P}$ vs $10log_{10}(\alpha)$ and $|S_{DL}|$

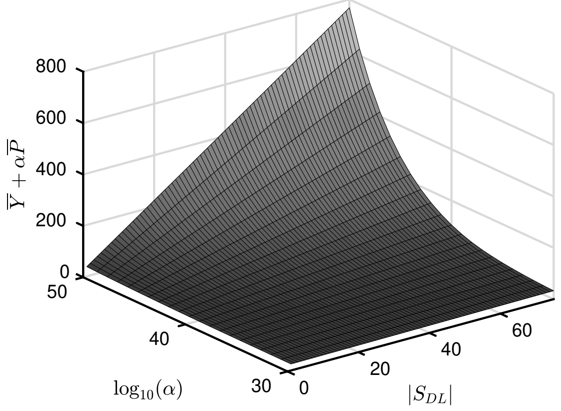

Figure 5.6: $\overline{f} = \overline{Y} + \alpha\overline{A}$ vs $10log_{10}(\alpha)$ and $|S_{DL}|$

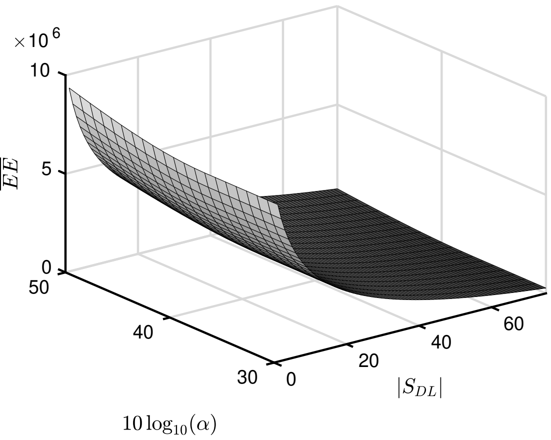

Figure 5.7: $\overline{EE}$ vs $10log_{10}(\alpha)$ and $|S_{DL}|$

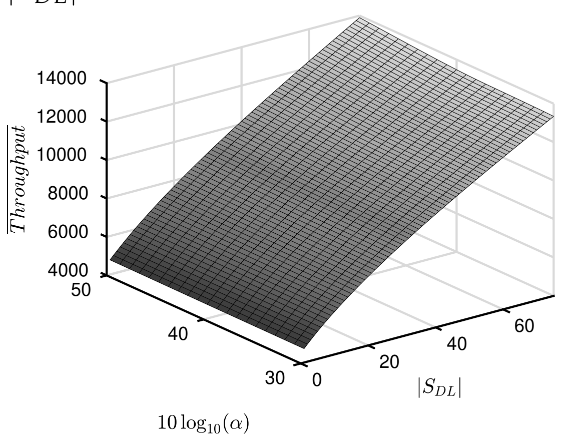

Figure 5.8: $\overline{Throughput}$ vs $10log_{10}(\alpha)$ and $|S_{DL}|$

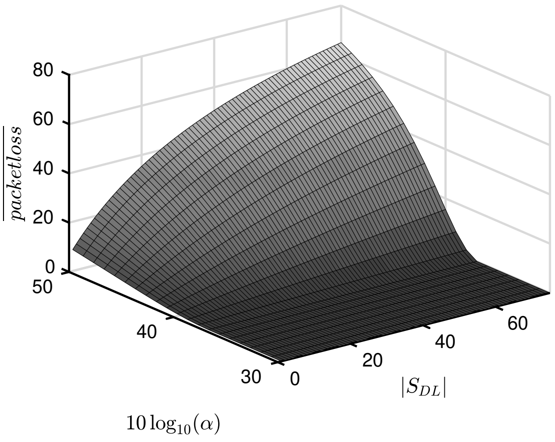

Figure 5.9: $\overline{packetloss}$ vs $10log_{10}(\alpha)$ and $|S_{DL}|$

5.7.3 Validations of expected cost and energy efficiency

In this subsection, we will validate our theoretical expressions with practical findings. In Figures 5.4-5.9 we plot $\overline{Y}$, $\overline{P}$ in mJ^{-1}, $\overline{Y} + \alpha\overline{P}$, $\overline{EE}$ in $bitmJ^{-1}Hz^{-1}$, average throughput $\overline{Throughput}$ in $Mbps$ and average packet loss $\overline{packetloss}$ in % against different values of $10\log_{10}(\alpha)$ and $|S_{DL}|$ respectively. We fix $|S_{CL}| = 25$ and vary $|S_{DL}| \in \{1, 2, \cdots, 75\}$ and $10\log_{10}(\alpha) \in \{30, 31, \cdots, 50\}$. We have calculated throughput using Shannon's capacity formula and for computation of packet loss, we have considered maximum number of channels available as 25. For each value of $|S_{DL}|$ and α, we simulate 100 instances and run RJCPA with $\rho = 100$ and report the average.

In Figures 5.4, 5.5 and 5.6 we observe that $\overline{Y}$, $\overline{P}$ and $\overline{f}$ increase with increasing $|S_{DL}|$. With increasing α, both $\overline{Y}$ and $\overline{f}$ increase but $\overline{P}$ decreases. These behaviors of $\overline{Y}$, $\overline{P}$ and $\overline{f}$ reflect the theoretical facts mentioned in Equations (5.41), (5.42) and (5.43) respectively. In Figure 5.7 we observe that with increasing $|S_{DL}|$, $\overline{EE}$ decreases. Since every CL gets different channel, when $|S_{DL}| \to 0$, $\overline{EE} \to \dfrac{\log(1+\Gamma)}{\sigma^2}$ which is indeed a extremely large value. As $|S_{DL}|$ increases, the data rate increases slowly (due to the logarithmic relation with SINR), but total power increases very fast. Hence $\overline{EE}$ decreases with increasing $|S_{DL}|$. This also follows from Equation (5.44), where we had shown that $\overline{EE}$ is a non-increasing function of κ, i.e., $|S_{DL}|$. We also observe that with increasing α, $\overline{EE}$ increases. This is also in accordance with the theoretical result presented in Equation (5.44).

We further see from Figure 5.8 that $\overline{Throughput}$ increases with both $|S_{DL}|$ and α. With increasing $|S_{DL}|$, total number of links increases and our algorithm allocates more channels to accommodate all of them subject to the maximum number of channels available (which is here the total number of links). Hence $\overline{Throughput}$ increases with $|S_{DL}|$. With increasing α, relative weight on $\overline{P}$ increases. As a result of that $\overline{Y}$ increases (i.e., more channels are allocated) and hence $\overline{Throughput}$ increases. On the other hand, if we restrict maximum number of channels that could be allocated, then our algorithm may not be able to allocate all the links. In Figure 5.9 we fix maximum number of channels that could be allocated to 25 and plot $\overline{packetloss}$ vs $|S_{DL}|$ and $10\log_{10}(\alpha)$. We observe that $\overline{packetloss}$ increases with both $|S_{DL}|$ and α. With increasing $|S_{DL}|$, more links are failing to be accommodated within 25 channels, resulting $\overline{packetloss}$ to increase. With increasing α, the relative weight on P increases and hence multiple DL cannot be shared with one CL due to increased interference. Hence $\overline{packetloss}$ increases.

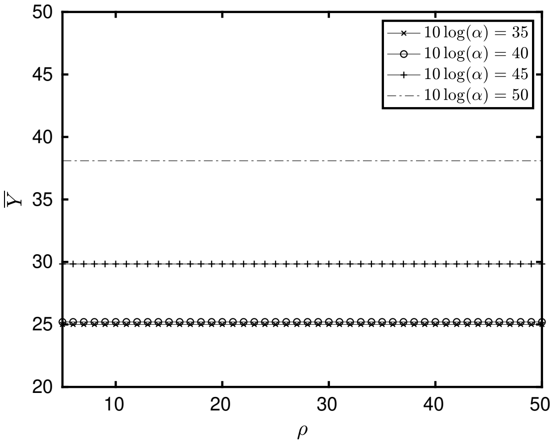

Figure 5.10: $\overline{Y}$ vs ρ

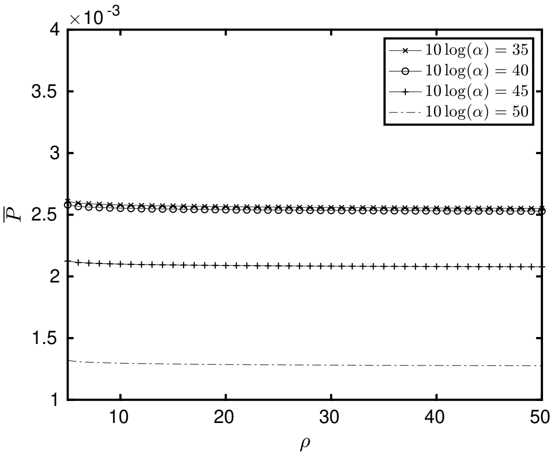

Figure 5.11: $\overline{P}$ vs ρ

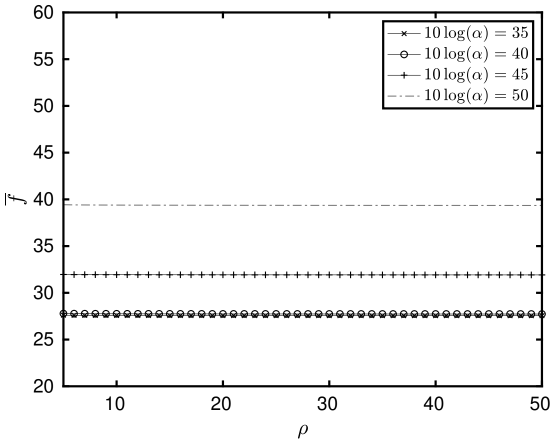

Figure 5.12: $\overline{f} = \overline{Y} + \alpha\overline{P}$ vs ρ

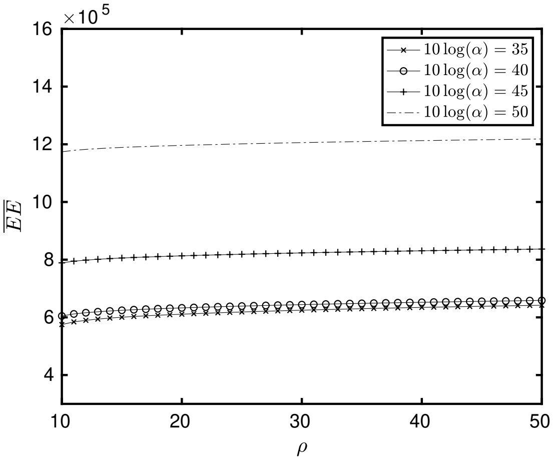

Figure 5.13: $\overline{EE}$ vs ρ

In Figures 5.10-5.13 we plot $\overline{Y}, \overline{P}, \overline{f} = \overline{Y} + \alpha\overline{P}$ and $\overline{EE}$ vs ρ for fixed $|S_{CL}| =$

$|S_{DL}| = 25$ and varying $10\log_{10}(\alpha) \in \{35, 40, 45, 50\}$ respectively. We observe that $\overline{Y}$, $\overline{P}$ and $\overline{f}$ decrease while $\overline{EE}$ increases with increasing ρ. So we can conclude that with increasing ρ, both the expected cost and expected energy efficiency improve, which also reflects the theoretical behavior discussed in Subsection 5.5.4.

5.8 Conclusion

In this chapter, we have formulated the joint power and channel allocation problem (JPCAP) in D2D underlaid cellular network as a cost minimization problem, where cost is defined as a linear combination of Y and P. We first showed that JPCAP is NP-hard and even providing a $n^{1/\epsilon}$ approximation for it is also NP-hard. Then we proposed a MILP formulation of this problem. As solving MILP is also NP-hard we proposed a GCPA algorithm which runs on an order of links. We proved that there exists an order for which this GCPA produces optimum solution. Then we proposed an IA algorithm which by actually evaluating less number of orders achieves the effect of evaluating large number of orders. As IA depends on the initial order hence we proposed a RJCPA algorithm to solve JPCAP. We showed that for some cases RJCPA solves JPCAP in expected polynomial time and for other cases it solves with slowly growing expected exponential time with high probability. We calculated the expected cost and energy efficiency produced by RJCPA. Through simulation, we have shown that RJCPA outperforms two existing approaches in terms of cost and energy efficiency. Finally we have validated our theoretical findings through simulation.

Chapter 6

Conclusion and Future Scope

In this thesis, we modeled the resource allocation problem in D2D communication as a cost minimization problem. We aimed to minimize the maximum channels, total number of perturbations, and total power. Since there is a natural trade-off between these three quantities we consider our minimization objective as 1) Maximum channel, 2) Total perturbations, 3) A linear combination of maximum channel and total perturbation, and 4) A linear combination of maximum channel and total power respectively. For first three cases, we simplified the problem in terms of interference graph. To minimize maximum channel, we proposed an ISH and a PISH technique. Since minimizing channel is essentially solving the graph coloring problem, we showed that ISH can solve the graph coloring problem in expected polynomial time. By definition, NP-complete class is a subset of NP class such that each problem inside NP can be reduced to each member of NP-complete class in polynomial time and graph coloring problem is a well-known NP-complete problem [11]. Since ISH is a randomized algorithm which solves the graph coloring problem in expected polynomial time, this implies that ”For each problem in NP there is a randomized algorithm which can solve it in expected polynomial time”. To minimize total perturbation, we proposed DC, RC and DDC algorithms. To minimize a linear combination of maximum channel and total perturbation, we proposed GP and GU. Finally we minimized a linear combination of maximum channel and total power. After showing the hardness, we developed a RJCPA algorithm to solve it. For each of these approaches, we theoretically analyse the performance of our algorithms, then compare and verify through simulation.

6.1 Future Scope

In this thesis we indeed tried to develop a novel paradigm of solving optimization problems in NP. This paradigm could be enhanced for other problems too. A proper formalisation of this paradigm might be an option for future study. Also the decentralization of each centralized approach mentioned here is a challenging task. Since interference graph is indeed a geometric graph, so it may be that each links need to communicate to only its neighbors, to assign its color and power. It is also possible to split the whole graph into small clusters in decentralized manner and then apply those algorithms only on it. GU can also be decentralized by considering k as a small value and each node has to store the information of its neighbors only at a time. Since the current movement of a link, depends on previous moves, further improvement could also be done by applying reinforcement learning and taking Markov model into consideration.

Bibliography

[1] Subhankar Ghosal and Sasthi C Ghosh. A randomized algorithm for joint power and channel allocation in 5g d2d communication. In *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, pages 1–5. IEEE, 2019.

[2] Ajay Pratap and Rajiv Misra. Firefly inspired improved distributed proximity algorithm for d2d communication. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pages 323–328. IEEE, 2015.

[3] Hesham ElSawy, Ekram Hossain, and Mohamed-Slim Alouini. Analytical modeling of mode selection and power control for underlay d2d communication in cellular networks. *IEEE Transactions on Communications*, 62(11):4147–4161, 2014.

[4] Ahmed Hamdi Sakr and Ekram Hossain. Cognitive and energy harvesting-based d2d communication in cellular networks: Stochastic geometry modeling and analysis. *IEEE Transactions on Communications*, 63(5):1867–1880, 2015.

[5] Mohsen Nader Tehrani, Murat Uysal, and Halim Yanikomeroglu. Device-to-device communication in 5g cellular networks: challenges, solutions, and future directions. *IEEE Communications Magazine*, 52(5):86–92, 2014.

[6] Zhesheng Lin, Yuancao Li, Si Wen, Yuehong Gao, Xin Zhang, and Dacheng Yang. Stochastic geometry analysis of achievable transmission capacity for

relay-assisted device-to-device networks. In *2014 IEEE international conference on communications (ICC)*, pages 2251–2256. IEEE, 2014.

[7] Hongliang Zhang, Yun Liao, and Lingyang Song. D2d-u: Device-to-device communications in unlicensed bands for 5g system. *IEEE Transactions on Wireless Communications*, 16(6):3507–3519, 2017.

[8] Jiajia Liu, Nei Kato, Jianfeng Ma, and Naoto Kadowaki. Device-to-device communication in lte-advanced networks: A survey. *IEEE Communications Surveys & Tutorials*, 17(4):1923–1940, 2014.

[9] Jiajia Liu, Hiroki Nishiyama, Nei Kato, and Jun Guo. On the outage probability of device-to-device-communication-enabled multichannel cellular networks: An rss-threshold-based perspective. *IEEE Journal on Selected Areas in Communications*, 34(1):163–175, 2015.

[10] Shijun Lin, Haichuan Ding, Yuguang Fang, and Jianghong Shi. Energy-efficient d2d communications with dynamic time-resource allocation. *IEEE Transactions on Vehicular Technology*, 68(12):11985–11999, 2019.

[11] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[12] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690, 2006.

[13] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[14] Min Sheng, Yuzhou Li, Xijun Wang, Jiandong Li, and Yan Shi. Energy efficiency and delay tradeoff in device-to-device communications underlaying cellular networks. *IEEE Journal on Selected Areas in Communications*, 34(1):92–106, 2015.

[15] Daquan Feng, Guanding Yu, Cong Xiong, Yi Yuan-Wu, Geoffrey Ye Li, Gang Feng, and Shaoqian Li. Mode switching for energy-efficient device-to-device communications in cellular networks. *IEEE Transactions on Wireless Communications*, 14(12):6993–7003, 2015.

[16] Dan Wu, Jinlong Wang, Rose Qingyang Hu, Yueming Cai, and Liang Zhou. Energy-efficient resource sharing for mobile device-to-device multimedia communications. *IEEE Transactions on Vehicular Technology*, 63(5):2093–2103, 2014.

[17] Tuong Duc Hoang, Long Bao Le, and Tho Le-Ngoc. Energy-efficient resource allocation for d2d communications in cellular networks. *IEEE Transactions on Vehicular Technology*, 65(9):6972–6986, 2015.

[18] Zhenyu Zhou, Mianxiong Dong, Kaoru Ota, Guojun Wang, and Laurence T Yang. Energy-efficient resource allocation for d2d communications underlaying cloud-ran-based lte-a networks. *IEEE Internet of Things Journal*, 3(3):428–438, 2015.

[19] Guanding Yu, Lukai Xu, Daquan Feng, Rui Yin, Geoffrey Ye Li, and Yuhuan Jiang. Joint mode selection and resource allocation for device-to-device communications. *IEEE transactions on communications*, 62(11):3814–3824, 2014.

[20] Jiahao Dai, Jiajia Liu, Yongpeng Shi, Shubin Zhang, and Jianfeng Ma. Analytical modeling of resource allocation in d2d overlaying multihop multichannel uplink cellular networks. *IEEE Transactions on Vehicular Technology*, 66(8):6633–6644, 2017.

[21] Demia Della Penda, Liqun Fu, and Mikael Johansson. Energy efficient d2d communications in dynamic tdd systems. *IEEE Transactions on Communications*, 65(3):1260–1273, 2016.

[22] Shijun Lin, Liqun Fu, and Yong Li. Energy saving with network coding design

over rayleigh fading channel. *IEEE Transactions on Wireless Communications*, 16(7):4503–4518, 2017.

[23] Xiaoming Tao, Xiao Xiao, and Jianhua Lu. A qos-aware power optimization scheme in ofdma systems with integrated device-to-device (d2d) communications. *Engine*, 2012.

[24] Lei Lei, Zhangdui Zhong, Chuang Lin, and Xuemin Shen. Operator controlled device-to-device communications in lte-advanced networks. *IEEE Wireless Communications*, 19(3):96–104, 2012.

[25] Gabor Fodor, Demia Della Penda, Marco Belleschi, Mikael Johansson, and Andrea Abrardo. A comparative study of power control approaches for device-to-device communications. In *2013 IEEE International Conference on Communications (ICC)*, pages 6008–6013. IEEE, 2013.

[26] Chia-Hao Yu, Klaus Doppler, Cassio B Ribeiro, and Olav Tirkkonen. Resource sharing optimization for device-to-device communication underlaying cellular networks. *IEEE Transactions on Wireless communications*, 10(8):2752–2763, 2011.

[27] Rui Yin, Caijun Zhong, Guanding Yu, Zhaoyang Zhang, Kai Kit Wong, and Xiaoming Chen. Joint spectrum and power allocation for d2d communications underlaying cellular networks. *IEEE Transactions on Vehicular Technology*, 65(4):2182–2195, 2015.

[28] Pekka Janis, Visa Koivunen, Cassio Ribeiro, Juha Korhonen, Klaus Doppler, and Klaus Hugl. Interference-aware resource allocation for device-to-device radio underlaying cellular networks. In *VTC Spring 2009-IEEE 69th Vehicular Technology Conference*, pages 1–5. IEEE, 2009.

[29] Mohammad Zulhasnine, Changcheng Huang, and Anand Srinivasan. Efficient resource allocation for device-to-device communication underlaying lte net-

work. In *2010 IEEE 6th International conference on wireless and mobile computing, networking and communications*, pages 368–375. IEEE, 2010.

[30] Hyunkee Min, Jemin Lee, Sungsoo Park, and Daesik Hong. Capacity enhancement using an interference limited area for device-to-device uplink underlaying cellular networks. *IEEE Transactions on Wireless Communications*, 10(12):3995–4000, 2011.

[31] Tinghan Yang, Rongqing Zhang, Xiang Cheng, and Liuqing Yang. Graph coloring based resource sharing (gcrs) scheme for d2d communications underlaying full-duplex cellular networks. *IEEE Transactions on Vehicular Technology*, 66(8):7506–7517, 2017.

[32] Dimitris Tsolkas, Eirini Liotou, Nikos Passas, and Lazaros Merakos. A graph-coloring secondary resource allocation for d2d communications in lte networks. In *2012 IEEE 17th international workshop on computer aided modeling and design of communication links and networks (CAMAD)*, pages 56–60. IEEE, 2012.

[33] Xuejia Cai, Jun Zheng, and Yuan Zhang. A graph-coloring based resource allocation algorithm for d2d communication in cellular networks. In *2015 IEEE International Conference on Communications (ICC)*, pages 5429–5434. IEEE, 2015.

[34] Philippe Galinie and Alain Hertz. A survey of local search methods for graph coloring. *Computers and Operations Research*, 33(9):2547–2562, 2006.

[35] Pardalos Panos, M, Mavridou Thelma, and Xue Jue. The graph coloring problem: A bibliographic survey. *Handbook of combinatorial optimization. Springer*, pages 1077–1141, 1998.

[36] Zhou Yangming, Hao Jin-Kao, and Duvala Béatrice. Reinforcement learning based local search for grouping problems: A case study on graph coloring. *Expert Systems with Applications*, 64:412–422, 2016.

[37] Shadi Mahmoudi and Shahriar Lotfi. Modified cuckoo optimization algorithm (mcoa) to solve graph coloring problem. *Applied soft computing*, 33:48–64, 2015.

[38] Laurent Moalic and Alexandre Gondran. Variations on memetic algorithms for graph coloring problems. *Journal of Heuristics*, 24(1):1–24, 2018.

[39] Wen Sun, Jin-Kao Hao, Xiangjing Lai, and Qinghua Wu. On feasible and infeasible search for equitable graph coloring. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 369–376, 2017.

[40] Hasenplaugh William, Kaler Tim, Schardl Tao, B, and Leiserson Charles, E. Ordering heuristics for parallel graph coloring. In *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures*, pages 166–177, 2014.

[41] Dominic JA Welsh and Martin B Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967.

[42] Alfred B Kempe. On the geographical problem of the four colours. *American journal of mathematics*, 2(3):193–200, 1879.

[43] Daniel Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[44] David S Johnson, Cecilia R Aragon, Lyle A McGeoch, and Catherine Schevon. Optimization by simulated annealing: An experimental evaluation; part ii. *Graph Coloring and Number Partitioning. Operations Research*, 39(3):378–406, 1991.

[45] John H Holland. Adaptation in natural and artificial systems. *Ann Arbor, MI: University of Michigan Press and*, 1992.

[46] Fred Glover. Tabu search and adaptive memory programming—advances, applications and challenges. In *Interfaces in computer science and operations research*, pages 1–75. Springer, 1997.

[47] Alain Hertz and Dominique de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 1987.

[48] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Publications, 1998.

[49] Matti Åstrand and Jukka Suomela. 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*, pages 294–302, 2010.

[50] Christos Koufogiannakis and Neal 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*, pages 171–179, 2009.

[51] Fabrizio Grandoni, Jochen Könemann, and Alessandro Panconesi. Distributed weighted vertex cover via maximal matchings. *ACM Transactions on Algorithms (TALG)*, 5(1):1–12, 2008.

[52] Changhao Sun, Wei Sun, Xiaochu Wang, and Qingrui Zhou. Potential game theoretic learning for the minimal weighted vertex cover in distributed networking systems. *IEEE transactions on cybernetics*, 49(5):1968–1978, 2018.

[53] Luis Barba, Jean Cardinal, Matias Korman, Stefan Langerman, André Van Renssen, Marcel Roeloffzen, and Sander Verdonschot. Dynamic graph coloring. In *Workshop on Algorithms and Data Structures*, pages 97–108. Springer, 2017.

[54] Shay Solomon and Nicole Wein. Improved dynamic graph coloring. *arXiv preprint arXiv:1904.12427*, 2019.

[55] Sayan Bhattacharya, Deeparnab Chakrabarty, Monika Henzinger, and Danupon Nanongkai. Dynamic algorithms for graph coloring. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–20. SIAM, 2018.

[56] Piotr Borowiecki and Elżbieta Sidorowicz. Dynamic coloring of graphs. *Fundamenta Informaticae*, 114(2):105–128, 2012.

[57] Davy Preuveneers and Yolande Berbers. Acodygra: an agent algorithm for coloring dynamic graphs. *Symbolic and Numeric Algorithms for Scientific Computing (September 2004)*, 6:381–390, 2004.

[58] Linda Ouerfelli and Hend Bouziri. Greedy algorithms for dynamic graph coloring. In *2011 International Conference on Communications, Computing and Control Applications (CCCA)*, pages 1–5. IEEE, 2011.

[59] Feng Yu, Amotz Bar-Noy, Prithwish Basu, and Ram Ramanathan. Algorithms for channel assignment in mobile wireless networks using temporal coloring. In *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems*, pages 49–58, 2013.

[60] Feng Shu, XiaoHu You, Mao Wang, YuBing Han, Yang Li, and WeiXing Sheng. Hybrid interference alignment and power allocation for multi-user interference mimo channels. *Science China Information Sciences*, 56(4):1–9, 2013.

[61] Minchae Jung, Kyuho Hwang, and Sooyong Choi. Joint mode selection and power allocation scheme for power-efficient device-to-device (d2d) communication. In *2012 IEEE 75th vehicular technology conference (VTC Spring)*, pages 1–5. IEEE, 2012.

[62] Yanxiang Jiang, Qiang Liu, Fuchun Zheng, Xiqi Gao, and Xiaohu You. Energy-efficient joint resource allocation and power control for d2d communications. *IEEE Transactions on Vehicular Technology*, 65(8):6119–6127, 2015.

[63] Zhenyu Zhou, Kaoru Ota, Mianxiong Dong, and Chen Xu. Energy-efficient matching for resource allocation in d2d enabled cellular networks. *IEEE Transactions on Vehicular Technology*, 66(6):5256–5268, 2016.

[64] Sihan Liu, Yucheng Wu, Liang Li, Xiaocui Liu, and Weiyang Xu. A two-stage energy-efficient approach for joint power control and channel allocation in d2d communication. *IEEE Access*, 7:16940–16951, 2019.

[65] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[66] Wei-Kuang Lai, You-Chiun Wang, He-Cian Lin, and Jian-Wen Li. Efficient resource allocation and power control for lte-a d2d communication with pure d2d model. *IEEE Transactions on Vehicular Technology*, 69(3):3202–3216, 2020.

[67] Javad Akbari Torkestani and Mohammad Reza Meybodi. A new vertex coloring algorithm based on variable action-set learning automata. *Computing and Informatics*, 29(3):447–466, 2012.

[68] Sidi Mohamed Douiri and Souad Elbernoussi. Solving the graph coloring problem via hybrid genetic algorithms. *Journal of King Saud University-Engineering Sciences*, 27(1):114–118, 2015.

[69] Francisco J Aragón Artacho, Rubén Campoy, and Veit Elser. An enhanced formulation for solving graph coloring problems with the douglas–rachford algorithm. *Journal of Global Optimization*, pages 1–21, 2018.

[70] Karim Baiche, Yassine Meraihi, Manolo Dulva Hina, Amar Ramdane-Cherif, and Mohammed Mahseur. Solving graph coloring problem using an enhanced binary dragonfly algorithm. *International Journal of Swarm Intelligence Research (IJSIR)*, 10(3):23–45, 2019.

[71] Subhankar Ghosal (https://mathoverflow.net/users/91159/subhankar ghosal). Prove that $\frac{(1-x)^{y(1-x)}}{(1-xy)^{1-xy}} \leq 1$, when $0 \leq x \leq 1$, $y \geq 1$ and $xy \leq 1$. MathOverflow. URL:https://mathoverflow.net/q/383250 (version: 2021-02-05).

[72] Kazunori Mizunoa and Seiichi Nishihara. Constructive generation of very hard 3-colorability instances. *Discrete Applied Mathematics*, 156(2):218–229, 2008.

[73] Zymolka Adrian, Koster Arie, M.C.A., and Wessaly Roland. Transparent optical network design with sparse wavelength conversion. In *Proceedings of the 7th IFIP Working Conference on Optical Network Design and Modelling*, pages 61–80, 2003.

[74] Carla Gomes and David Shmoys. Completing quasigroups or latin squares: A structured graph coloring problem. In *In D. S. Johnson, A. Mehrotra, M. Trick (eds.), Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, pages 22–39, 2002.

[75] Shahadat Hossain and Trond Steihaug. Graph coloring in the estimation of mathematical derivatives. In *In D. S. Johnson, A. Mehrotra, M. Trick (eds.), Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, pages 9–16, 2002.

[76] Kazunori Mizuno and Seiichi Nishihara. Toward ordered generation of exceptionally hard instances for graph 3-colorability. In *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, pages 1–8, 2002.

[77] Massimiliano Caramia and Paolo Dell'Olmo. Coloring graphs by iterated local search traversing feasible and infeasible solutions. *Discrete Applied Mathematics*, 156(2):201–217, 2008.

[78] Gary Lewandowski and Anne Condon. Experiments with parallel graph coloring heuristics and applications of graph coloring. *DIMACS Series in Discrete Mathematics and Theoretical Computer Sciences*, 26:309–334, 1996.

[79] Anuj Mehrotra and Micheal A Trick. A column generation approach for graph coloring. *INFORMS Journal On Computing*, 8(4):344–354, 1996.

[80] Culberson Joseph, Beacham Adam, and Papp Denis. Hiding our colors. In *In Proceedings of the CP'95 Workshop on Studying and Solving Really Hard Problems*, pages 31–42, 1995.

[81] Adalat Jabrayilov and Petra Mutzel. New integer linear programming models for the vertex coloring problem. In *Latin American Symposium on Theoretical Informatics*, pages 640–652. Springer, 2018.

[82] Yassine Meraihi, Amar Ramdane-Cherif, Mohammed Mahseur, and Dalila Achelia. A chaotic binary salp swarm algorithm for solving the graph coloring problem. In *International Symposium on Modelling and Implementation of Complex Systems*, pages 106–118. Springer, 2018.

[83] Taha Mostafaie, Farzin Modarres Khiyabani, and Nima Jafari Navimipour. A systematic study on meta-heuristic approaches for solving the graph coloring problem. *Computers & Operations Research*, page 104850, 2019.

[84] Yangming Zhou, Jin-Kao Hao, and Béatrice Duval. Reinforcement learning based local search for grouping problems: A case study on graph coloring. *Expert Systems with Applications*, 64:412–422, 2016.

[85] Yangming Zhou, Béatrice Duval, and Jin-Kao Hao. Improving probability learning based local search for graph coloring. *Applied Soft Computing*, 65:542–553, 2018.

[86] Colin JH McDiarmid. Colouring random graphs badly. *Graph Theory and Combinatorics. RJ Wilson, Ed. Pitman Advanced Publishing Program, San Francisco*, 1979.

[87] Geoffrey R Grimmett and Colin JH McDiarmid. On colouring random graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 77, pages 313–324. Cambridge University Press, 1975.

[88] Dieter Mitsche, Giovanni Resta, and Paolo Santi. The random waypoint mobility model with uniform node spatial distribution. *Wireless networks*, 20(5):1053–1066, 2014.

[89] Subhankar Ghosal and Sasthi C Ghosh. An incremental search heuristic for coloring vertices of a graph. *Graphs and Combinatorial Optimization: from Theory to Applications: CTW2020 Proceedings*, pages 39–52, 2021.

[90] Klaus Doppler, Mika Rinne, Carl Wijting, Cassio B Ribeiro, and Klaus Hugl. Device-to-device communication as an underlay to lte-advanced networks. *IEEE Communications Magazine*, 47(12):42–49, 2009.

[91] Gábor Fodor, Erik Dahlman, Gunnar Mildh, Stefan Parkvall, Norbert Reider, György Miklós, and Zoltán Turányi. Design aspects of network assisted device-to-device communications. *IEEE Communications Magazine*, 50(3):170–177, 2012.

[92] Hongseok Kim and Gustavo De Veciana. Leveraging dynamic spare capacity in wireless systems to conserve mobile terminals' energy. *IEEE/ACM transactions on networking*, 18(3):802–815, 2009.

[93] Durgesh Singh and Sasthi C Ghosh. Mobility-aware relay selection in 5g d2d communication using stochastic model. *IEEE Transactions on Vehicular Technology*, 68(3):2837–2849, 2019.

[94] Li Wang, Huan Tang, Huaqing Wu, and Gordon L Stüber. Resource allocation for d2d communications underlay in rayleigh fading channels. *IEEE Transactions on Vehicular Technology*, 66(2):1159–1170, 2016.