Distributed Resource Allocation in SINR Models

A Project Report

submitted by

VINOD S

in partial fulfilment of the requirements

for the award of the degree of

MASTER OF TECHNOLOGY

DEPARTMENT OF Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2014

THESIS CERTIFICATE

This is to certify that the thesis titled Distributed Resource Allocation in SINR Models, submitted by **Vinod S**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr Srikrishna Bashyam
Research Guide
Associate Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Dr.Krishna Jagganathan
Research Guide
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 4th June 2014

ACKNOWLEDGEMENTS

I would like to express my gratitude to my project advisers, Dr. Srikrishna Bhashyam and Dr. Krishna Jagannathan, for this wonderful opportunity of working with them for the project. It is their guidance, support and encouragement that motivated me to work and come up with solutions to the problems we faced during the course of the research work. The confidence I gained over the last year by working with them has been tremendous and I would treasure the same.

I also thank Dr. Venkatesh Ramaiyan for the course "Communication Networks" and Dr. Radha Krishna Ganti for the course "Convex Optimization". These two courses were very relevant for the project. I also thank our faculty adviser Dr. T.G. Venkatesh for helping us with all the administrative formalities during the last two years.

I thank my friends Akhil C, Aseem, Debayani, Geetha C, Gopal, Jose T, Ravi Kiran, Ravi Kolla, Reshma K B, Sudarshan P and Swamy P S who have helped me a lot during my stay at IIT.

ABSTRACT

KEYWORDS: SINR Model; Throughput Optimal Resource Allocation

We study the problem of distributed resource allocation in the SINR model. Our model consists of a fixed number of transmitters and their distinct receivers. In the models that we consider, the data rate a transmitter gets depends on the SINR at the receiver. The objective is to allocate the resource in such a way that the queue lengths at the transmitter remains bounded in a probabilistic sense. We consider two SINR models. The first model we consider is called SINR threshold model in which the data rate is 1 or 0 depending upon the SINR being above a predefined threshold or not. The second model we consider is the SINR adaptation model in which the data rate is a concave function of SINR. By distributed we mean that every transmitter must take decisions only by using local information of SINR at the receiver. In this thesis, we propose distributed algorithms that are throughput optimal for the first model under joint power control and scheduling and for the second model under scheduling case.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AVIP	Activation Vector Indexing Phase
AVTPIP	Activation Vector and Transmit Power Identification Phase
DFP	Data Frame Phase
FM-PCA	Foschini Miljani - Power Control Algorithm
IITM	Indian Institute of Technology, Madras
IP	Initialization Phase
LP	Linear Program
SINR	Signal to Interference and Noise Ratio

NOTATION

N	Number of links in the network.
S_i	Time average service rate of transmitter i.
r_{ij}	Distance between i^{th} and j^{th} receiver.
α	Constant attenuation loss.
η	Path loss exponent.
β	SINR threshold.
β_0	Maximum balanced SINR.
g_{ij}	Gain between i^{th} receiver and j^{th} transmitter.
z_{ij}	Normalized gain between i^{th} receiver and j^{th} transmitter.
λ_i	Arrival rate of i^{th} transmitter.
P_i	Transmission power of i^{th} transmitter.
Q_i	Queue length of transmitter i.
S_i	Time average service rate of transmitter i.
e	Activation vector.
$E(f)$	Set of all feasible activation vectors.
κ_Z^i	Eigen value of matrix Z.
κ_A^i	Eigen value of matrix A.
J_Z^i	Eigen vector of matrix Z.
J_A^i	Eigen vector of matrix A.
Θ	Penalty factor.
$E(f)$	Set of all feasible activation vectors.

CHAPTER 1

INTRODUCTION

Effective resource allocation among a set of interfering users is important so as to achieve fairness amongst the users. There are different notions of fairness found in literature. A particular notion of fairness dictates what the objective must be for all the users. Once the notion of fairness is decided, a resource allocation algorithm will decide how exactly the users go about sharing the resource so as to achieve the objective set forth by the fairness criteria. One such notion of fairness is to share the resource in such a way so that the users will be able to support an arrival process meaning that the queue lengths of all the users remain bounded in some probabilistic sense. A resource allocation algorithm that achieves this objective whenever at all this is possible is called a throughput optimal algorithm.

But what exactly is the resource and what do we mean by interfering users? The resource could refer to the transmission data rate in a wireless channel and the users interfere because increasing data rate of a user could decrease the date rate of other users. Since a wireless channel in it's full generality is too complicated to study, researchers make simplifying assumptions for the wireless channel and come up with resource allocation algorithms for such simplified models. Different assumptions leads to different network models. The most commonly encountered network model is the independent set model in which the independent sets of the network are specified explicitly. An independent set is a set of users that do not interfere with each other meaning that if the set of users that are transmitting at some time form an independent set, then all of them will be having a data rate of 1 unit. So the throughput optimal resource allocation problem reduces to a problem of sharing of the channel in time (scheduling) between the inde-

pendent sets. Further simplification of this independent set model leads to a conflict graph interference model. For this model, the interference relations can be represented using a graph called conflict graph. In a conflict graph the users form the vertices and an undirected edge between the users indicate that these two users cannot transmit successfully at the same time. So from a conflict graph interference model, we can deduce the independent sets by selecting users that do not share an edge in the conflict graph.

There are other simple network models in which the network connectivity is expressed as a graph. In such models the users can transmit directly only to their neighbors subject to other interference constraints. For example, in a primary interference model, the only interference constraint is that the user cannot transmit or receive at the same time. Similarly in a K-Hop interference model, the transmission is free of interference if all the users in K-Hop neighborhood are silent.

In this report, we consider two interference models in which the transmission data rate is a function of the Signal to Interference and Noise Ratio (SINR). A user in this model refers to a transmitter and its distinct receiver. The transmitter transmits at some power level to its corresponding receiver and this power decays as a function of distance. At the receiver, the transmission of all other transmitters other than its corresponding transmitter will be perceived as additive receiver noise. The first model we consider is called SINR threshold model in which the transmission rate is 1 if the SINR is greater than or equal to a predefined threshold and 0 otherwise. The second model we consider is called SINR adaptation model in which the transmission rate is a concave function of the SINR. Depending on the parameters taken into account, we can study these models under three different contexts. They are (a) Power control, (b) Scheduling and (c) Joint Power Control and Scheduling. In power control case, the users share the resource only by limiting their transmission power. In scheduling case, the users share the channel in time between different subsets of users with the users transmitting at same common

power level whenever they are active. In the most general case of joint power control and scheduling, the users while sharing the resource in time between different subsets of users, they are also allowed to adapt their transmission power depending on the subset of active users.

A resource allocation algorithm decides how at each instant of time, the resource is shared among the interfering users to meet the network objective. The Maxweight algorithm introduced in (Tassiulas and Ephremides, 1992) is a throughput optimal algorithm for the independent set model. However as the Maxweight algorithm computes the schedule by comparing the sum of queue length of independent sets, it does not lend itself easily to a distributed implementation. A distributed algorithm would be preferable as it eliminates the need to have a centralized controller and the associated overheads.

In this report we propose distributed algorithms for the two SINR models mentioned above. The distributed algorithm we propose is throughput optimal in the following cases, (a) SINR threshold model under joint power control and scheduling and (b) SINR adaptation model under scheduling.

1.1 Related Work

In Jiang and Walrand (2010*a*), a continuous time distributed throughput optimal algorithm was proposed for a conflict graph based interference model. In Ni *et al.* (2012), a discrete time distributed throughput optimal algorithm for the same model was proposed which had better delay performance and also was able to account for collisions and other issues such as hidden nodes, exposed nodes. In Lee *et al.* (2012) an algorithm was proposed which was shown to be throughput optimal for the SINR adaptation model under joint power control and scheduling. However this algorithm required the users to exchange control information between them and hence cannot be called distributed in a

strict sense. In (Chaporkar and Proutiere, 2013) a distributed algorithm was proposed for the SINR adaptation model under scheduling. In Yi and Veciana (2007) a distributed algorithm was proposed for the SINR threshold model under scheduling.

1.2 Our Contributions

In this report, we present throughput optimal algorithms for the SINR threshold model under joint power control and scheduling and also for the SINR adaptation model under scheduling. We first describe the capacity region for the models which specify the conditions on the arrival process so that there exists some resource allocation policy that will stabilize all the queues. Our approach is to invert the capacity region conditions to find out the unknown parameters that come up in these conditions that will make the queues stable.

1.3 Organization of Thesis

This thesis is organized as follows. In chapter 2, we describe the network model that we consider. In chapter 3 we describe the stability criterion and also present the capacity region for the models we consider. In chapter 4, we describe the distributed algorithm for the SINR threshold model under joint power control and scheduling. In chapter 5 we briefly describe how the approach used in chapter 4 can be used for obtaining a distributed throughput optimal algorithm for the SINR adaptation model. In chapter 6 we present the simulation results of our algorithm and in chapter 7 we conclude the thesis.

CHAPTER 2

NETWORK MODEL

The network we consists of N transmitter nodes and N receiver nodes whose locations are fixed. Each transmitter node intends to communicate to a distinct receiver node. A link will denote a transmitter and its corresponding receiver. The term *network configuration* will denote the location of the different transmitters and receivers. We assume an ergodic arrival process for the network with arrival rate vector $\lambda = [\lambda_1,\ \lambda_2,\ ...,\ \lambda_N]^T$ units, where λ_i is the arrival rate to the transmitter of link i. For the `SINR` threshold model the unit of arrival rate is $packets/secs$ and for the `SINR` adaptation model it will be $bits/sec$. The time average service rate of link i is denoted as S_i. The queue length of transmitter i will be denoted as Q_i. P_i will denote the transmission power of i^{th} transmitter. We define a set $\mathcal{P}$ called the power constraint set such that the transmission power of link i, P_i satisfies the condition $P_i \in \mathcal{P}, \forall i \in \{1, 2, ..., N\}$. $P = [P_1,\ P_2,\ ...,\ P_N]^T$ will be called the power vector. $g_{ij},\ i, j \in \{1, 2, ..., N\}$ will denote the constant gain coefficient between j^{th} transmitter and i^{th} receiver. Let I_i denote the interference power at the receiver of link i. Then $I_i = \sum_{j=1,\, j\neq i}^{N} P_j\ g_{ij}$. N_0 is the additive receiver noise power. So the $SINR$ at the receiver of link i, with a power vector of P denoted as $SINR_i[P]$ is given by

$$SINR_i[P] = \frac{P_i\ g_{ii}}{N_0 + \sum_{j=1,\, j\neq i}^{N} P_j\ g_{ij}} \tag{2.1}$$

Define an $N \times N$ matrix G such that $G(i, j)$, the $(i, j)^{th}$ entry in G, is given by $G(i, j) = g_{ij}, \forall i, j \in \{1, 2, ..., N\}$. G will be called link gain matrix. We will assume that the matrix G is a full rank matrix. Also define another $N \times N$ matrix Z such that $Z(i, j)$,

the $(i,j)^{th}$ entry in Z is z_{ij}, where z_{ij} is given by $z_{ij} = \frac{g_{ij}}{g_{ii}}$, $\forall i,j \in \{1,2,...,N\}$. Z is called the normalized link gain matrix. Note that both Z and G are positive matrices.

2.1 SINR Threshold Model

In the SINR threshold model, the transmission rate can take on only two values depending upon the SINR at the receiver. The transmission rate for a link i, using a power vector P denoted as $Rate_i[P]$ is given by

$$Rate_i[P] = \begin{cases} 1 & \textbf{if } SINR_i[P] \geq \beta \\ 0 & \textbf{if } SINR_i[P] < \beta \end{cases} \tag{2.2}$$

β in the above expression is called the threshold SINR. The transmission rate is normalized to the packet size.

2.1.1 Feasibility of SINR Constraint

To maintain successful transmission for all the links in the network, we should be able to find a power vector that satisfies the SINR constraint for all the links simultaneously. We will say that a threshold SINR of β is feasible for the network if there exists a power vector that can simultaneously achieve the SINR constraint for all the links. The question of finding such a power vector that satisfies the SINR constraints is addressed in this section.

Let $\kappa_Z^1, \kappa_Z^2, ..., \kappa_Z^N$, be the N eigen values of the matrix Z. In particular, let κ_Z^1 be the dominant eigen vector of the matrix Z, i.e., $|\kappa_Z^1| > |\kappa_Z^i|$, $\forall i \in \{2,3,...,N\}$. The dominant eigen value κ_Z^1 satisfies $\kappa_Z^1 \in \mathbb{R}^+$ and using eigen value bounds for positive

matrices, ($\kappa_Z^1 \geq \underset{i\in\{1,2,...,N\}}{min} \sum_{j\in\{1,2,...,N\}} z_{ij}$) we have the result $\kappa_Z^1 > 1$. Let $J_Z^1, J_Z^2, ..., J_Z^N$ be the N eigen vectors corresponding to the eigen values $\kappa_Z^1, \kappa_Z^2, ..., \kappa_Z^N$ respectively. For this network of interfering links, there exits a parameter called maximum balanced SINR denoted as β_0, which is the maximum common value of SINR for all links that can be attained by using any power vector P. β_0 is given by the following expression (Zander, 1992*b*)

$$\beta_0 = \frac{1}{\kappa_Z^1 - 1} \tag{2.3}$$

The power vector that achieves this maximum balanced SINR (β_0) is the eigen vector J_Z^1 corresponding to the eigen value κ_Z^1. The result in Eq: 2.3 is derived using the assumption of receiver noise power being zero ($N_0 = 0$). So this quantity would serve as an upper bound for the achievable threshold SINR (β) values. In the rest of the paper, we will assume that the transmitter can use any positive power value or in other words that the power constraint set $\mathcal{P} = (0, \infty)$. We remark that this assumption would only mean that the additive receiver noise (N_0) is sufficiently low when compared with the maximum transmission power limit for the links.

For maintaining successful transmission for all the links, we need to find a power vector P that satisfies

$$SINR_i[P] \geq \beta, \ \forall i \in \{1, 2, ..., N\} \tag{2.4}$$

$$\implies \ P_i \, g_{ii} - \sum_{j=1,\, j\neq i}^{N} P_j \, \beta \, g_{ij} \geq \beta N_0, \ \forall i \in \{1, 2, ..., N\} \tag{2.5}$$

The above N constraints can be expressed in matrix form as

$$\begin{bmatrix} g_{11} & -\beta g_{12} & ... & -\beta g_{1N} \\ -\beta g_{21} & g_{22} & ... & -\beta g_{2N} \\ ... & ... & ... & ... \\ -\beta g_{N1} & -\beta g_{N2} & ... & g_{NN} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ ... \\ P_N \end{bmatrix} \geq \begin{bmatrix} \beta N_o \\ \beta N_o \\ ... \\ \beta N_o \end{bmatrix}$$

Define a new parameter for the network, an $N \times N$ matrix $\tilde{G}$. The entries of $\tilde{G}$ are as follows $\tilde{G}(i,i) = g_{ii}, \forall i \in \{1, 2, ..., N\}$ and $\tilde{G}(i,j) = -\beta\, g_{ij}, \forall i, j \in \{1, 2, ..., N\}, i \neq j$. Also define an $N \times 1$ matrix $B = [\beta N_0,\ \beta N_0,\ ..., \beta N_0]^T$. Then the above $SINR$ constraints on the power vector can be compactly expressed as $\tilde{G}P \geq B$.

Among the possible power vectors that satisfy the `SINR` constraint, we define an optimal power vector as the one with least l_1 norm. Then the optimal power vector is the optimal point of the following linear program (LP).

$$minimize\ \ L(P) = \sum_{i=1}^{N} P_i$$

$$subject\ to\ \ \tilde{G}P \geq B\ ,\quad P \geq 0$$

Theorem 1 *The optimal power vector P^* satisfies $\tilde{G}P^* = B$. Hence the threshold $SINR$ is attainable if and only if the power vector P^* satisfying $\tilde{G}P^* = B$ also satisfies $P^* > 0$.*

Proof 1 *Following terminology in (Luenberger and Ye, 2008), we first convert the LP into standard form by adding N surplus variables $\beta' = [\beta'_1,\ \beta'_2,\ ...\beta'_N]^T$. Now the LP can be written as*

$$minimize\ \ L(P) = \sum_{i=1}^{N} P_i$$

$$subject\ to\ \ [\tilde{G}\ -I] \begin{bmatrix} P \\ \beta' \end{bmatrix} = B\ ,\quad [P\ \ \beta'] \geq 0$$

To find the optimal power vector, it is enough to search over basic feasible solutions. The

absence of any basic feasible solution will indicate that the LP is infeasible (Luenberger and Ye, 2008, Section 2.4). From the $2N$ variables, a basic solution is obtained by setting N variables to be zero and solving the equality constraints for the rest N variables. Note that since G was assumed to be a full rank matrix, the matrix $[\tilde{G} \ -I]$ is also of rank N. Now observe that any basic solution having $P_i = 0$, for some $i \in \{1, 2, ..., N\}$ cannot be feasible as the SINR *constraint for the i^{th} link will fail otherwise. So the only basic solution that can be feasible corresponds to $\beta_i' = 0, \forall i = 1, 2, ..., N$. Hence the optimal power vector P^* satisfies $\tilde{G}P^* = B$. This optimal power vector is feasible if and only if $P^* > 0$. Hence the result.*

With an optimal power vector of P^*, the SINR for all the links is exactly equal to β. In general it may not be possible for the network to maintain successful transmission for all the links in the network. This situation is characterized by using the two equivalent conditions ($\beta > \beta_0$) or ($P^* \not> 0$).

2.1.2 FM-PCA in the feasible case

When the SINR threshold of β is feasible, the Foschini Miljanic-Power Control Algorithm (FM-PCA) (Foschini and Miljani, 1993) can be used to find the optimal power vector of P^* in a distributed manner. In FM-PCA, the power P_i of a link i is updated every discrete time slots as follows

$$P_i \leftarrow (1 - \epsilon)P_i + \epsilon \, P_i \, \frac{\beta}{SINR_i[P]} \, , \epsilon \in (0, 1] \tag{2.6}$$

Starting from any power vector, the above iteration when executed for all the links, was proved to make the power vector converge to the optimal power vector P^* in case the SINR threshold of β is feasible.

2.1.3 FM-PCA in the infeasible case

In this section, we analyze the FM-PCA in the infeasible scenario ($\beta > \beta_0$), to study how the power and SINR evolves in this case. We first prove that the transmission power for all links diverges in this case as the algorithm evolves. As a consequence of diverging powers, the interference power at the receiver for all links also diverges and then we will analyze the algorithm after neglecting the finite receiver noise term N_0 to establish a useful result for FM-PCA in the infeasible case. Here we will consider the step size $\epsilon \in (0, 1)$. The case of $\epsilon = 1$ can be worked out similarly.

Plug Eq: (2.1) into Eq: (2.6) and the evolution of power of link i according to FM-PCA can be expressed as

$$P_i \leftarrow (1-\epsilon)P_i + \epsilon\, P_i \frac{\beta}{\dfrac{g_{ii}P_i}{N_0 + \sum\limits_{j=1, j\neq i}^{N} g_{ij}P_j}} \tag{2.7}$$

$$P_i \leftarrow (1-\epsilon)\left(P_i + \frac{\epsilon\beta}{1-\epsilon}\sum_{j=1, j\neq i}^{N} z_{ij}P_j\right) + \frac{\epsilon\beta N_0}{g_{ii}} \tag{2.8}$$

So the evolution of power vector P can be expressed as

$$P \leftarrow (1-\epsilon)\, A\, P + C \tag{2.9}$$

where A is an $N \times N$ matrix such that $A(i,i) = 1, \forall i \in \{1, 2, ..., N\}$ and $A(i,j) = c\, Z(i,j)\ ,\ \forall i, j \in \{1, 2, ..., N\},\ i \neq j$, and $c = \dfrac{\epsilon\beta}{1-\epsilon}$. C is an $N \times 1$ matrix with entries $C(i,1) = \dfrac{\epsilon\beta N_0}{g_{ii}}\ ,\ i = 1, 2, ..., N$

We first prove two lemmas relating the eigen pairs of the matrices Z and A.

Lemma 1 *The matrices Z and A have the same set of eigen vectors.*

Proof 1 *Let κ_Z^1, κ_Z^2,...,κ_Z^N be the eigen values of the positive matrix Z and let $J_Z^1, J_Z^2, ..., J_Z^N$ be the corresponding eigen vectors. Let $\kappa_A^1, \kappa_A^2, ..., \kappa_A^N$ be the eigen values of the positive matrix A and let $J_A^1, J_A^2, ..., J_A^N$ be the corresponding eigen vectors. Let us express the matrix Z as $Z = I + Z'$, where I is the $N \times N$ Identity matrix and Z' is an $N \times N$ matrix such that $Z'(i,i) = 0, \forall i$ and $Z'(i,j) = Z(i,j)$, $\forall i, j$, $i \neq j$.*

The matrix A can be expressed as $A = I + c\,Z'$, where I is the $N \times N$ Identity matrix. For some $i \in \{1, 2, ..., N\}$, we have $Z J_Z^i = \kappa_Z^i J_Z^i \implies (I + Z')J_Z^i = \kappa_Z^i J_Z^i$
$\implies J_Z^i + Z' J_Z^i = \kappa_Z^i J_Z^i \implies Z' J_Z^i = (\kappa_Z^i - 1) J_Z^i$.

Thus J_Z^i is an eigen vector of Z' also. We have,

$$A J_Z^i = (I + c\,Z')J_Z^i = (1 + c\,(\kappa_Z^i - 1))J_Z^i \tag{2.10}$$

So J_Z^i is an eigen vector of matrix A also with the eigen value $(1 + c\,(\kappa_Z^i - 1))$. Similarly for every eigen vector J_A^i of A with eigen value κ_A^i, J_A^i is also an eigen vector of matrix Z with eigen value $(1 + \frac{(\kappa_A^i - 1)}{c})$. Thus we can conclude that the matrices A and Z have the same set of eigen vectors. Hence the lemma.

Henceforth, J_Z^i, $i \in \{1, 2, ..., N\}$ will denote the eigen vector of matrices Z and A with eigen values κ_Z^i and κ_A^i respectively.

Lemma 2 *$\kappa_A^1 = 1 + c\,(\kappa_Z^1 - 1)$ is the dominant eigen vector for the matrix A. i.e. $|\kappa_A^1| > |\kappa_A^i|\,, \forall i \in \{2, 3, ..., N\}$.*

Proof 2 *Assume that for some $j \in \{2, 3, ..., N\}$, κ_A^j is the dominant eigen value of the positive matrix A. So κ_A^j satisfies $|\kappa_A^j| > |\kappa_A^i| \forall i \in \{1, 2, ..., N\} \backslash \{j\}$. As κ_A^j is the dominant eigen value of A, κ_A^j satisfies $\kappa_A^j \in \mathbb{R}^+$ and $\kappa_A^j > 1$. Also as κ_Z^1 is the dominant eigen value of Z, κ_Z^1 satisfies $\kappa_Z^1 \in \mathbb{R}^+$ and $\kappa_Z^1 > 1$.*

Consider $\kappa_A^1 = 1 + c(\kappa_Z^1 - 1)$. *We have* $\kappa_A^1 \in \mathbb{R}^+$ *and* $\kappa_A^1 > 1$.

Now we have, $|\kappa_A^j| > |\kappa_A^1| \implies \kappa_A^j > \kappa_A^1$

$\implies 1 + c(\kappa_Z^j - 1) > 1 + c(\kappa_Z^1 - 1)$

$\implies \kappa_Z^j > \kappa_Z^1 \implies |\kappa_Z^j| > |\kappa_Z^1|$

But this is not possible as κ_Z^1 *is the dominant eigen value of positive matrix* Z. *Hence our assumption is wrong and* κ_A^1 *indeed is the dominant eigen value of the positive matrix* A.

Let us ignore the matrix C in the iteration given by Eq: (2.9) and consider the evolution of power according to the following update rule

$$P \leftarrow (1 - \epsilon)\ A\ P \tag{2.11}$$

Say the update occurs at time $t = 0, 1, 2, ...$ Let $P(0), P(1), P(2), ...$ denote the power vector at times $t = 0, 1, 2, ...$ due to iteration given by (2.9), where $P(0)$ is the initial power vector. Starting from the same power vector $P(0)$ as above let $\widetilde{P}(1), \widetilde{P}(2), ...$ be the sequence of power vectors due to iteration given by Eq: (2.11). As C contains entries that are strictly positive, it is clear that for all $t = 1, 2, ...$ we have $\widetilde{P}(t) < P(t)$.

Let us decompose the initial power vector $P(0)$ as $P(0) = \sum_{i=1}^{N} p_0^i\ J_Z^i$.

Assume that the starting power vector $P(0)$ has entries that are strictly positive. Then from Zander (1992*a*), we have

$$\lim_{t \to \infty}\ \widetilde{P}(t) = p_0^1\ (1 - \epsilon)^t\ (\kappa_A^1)^t\ J_Z^1\ ,\ \text{where}\ p_0^1 > 0. \tag{2.12}$$

Now $(1 - \epsilon)\kappa_A^1 = (1 - \epsilon)(1 + \frac{\epsilon\beta}{1 - \epsilon}(\kappa_Z^1\text{-}1)) = 1 - \epsilon + \frac{\epsilon\beta}{\beta_0} > 1$ as $\beta > \beta_0$ due to infeasibility.

So as $t \rightarrow \infty$, $\widetilde{P}_i(t) \rightarrow \infty$, $\forall i \in \{1, 2, ..., N\}$. Now since $\widetilde{P}_i(t) \rightarrow \infty$, we have $P_i(t) \rightarrow \infty$, $\forall i \in \{1, 2, ..., N\}$ Hence in the infeasble scenario, the interference power at the receiver of all links grows without bounds and hence we can ignore the finite receiver noise term of N_0.

Let us say that at some finite time $T \geq 1$, we ignore the term N_0. For any starting power vector $P(0)$ with non negative components, note that the power vector at time T, $P(T)$ will have entries that are strictly positive. Let us decompose the power vector $P(T)$ as $P(T) = \sum_{i=1}^{N} p_T^i \; J_Z^i$. With the term N_0 ignored, the power evolves according to FM-PCA as

$$P \; \leftarrow \; (1 - \epsilon) \; A \; P \tag{2.13}$$

Then similar to above section, we have

$$\lim_{t \rightarrow \infty} \; P(t) = p_T^1 (1 - \epsilon)^t \; (\kappa_A^1)^t \; J_Z^1, \text{ where } p_T^1 > 0 \tag{2.14}$$

Hence from (Zander, 1992*a*), we have as $t \rightarrow \infty$, $\texttt{SINR}_\texttt{i}[P(t)] \rightarrow \beta_0$. Thus when the SINR threshold of β is not feasible for the network, FM-PCA will make the SINR for all the links converges to β_0 given by Eq: (2.3).

We illustrate these ideas using simulations of FM-PCA done in MATLAB. Figure: 2.1 shows the location of links for the network we consider. For this network, we first execute FM-PCA using a threshold SINR β of 1, which is infeasible. From Figure: 2.2 we see that the power of all the users diverges and from Figure: 2.3 we see that the SINR for all the links converges a constant (β_0). We then execute FM-PCA using a threshold SINR β of 0.1 which is feasible. From Figure: 2.4 we see that the power vector converges to a constant (P^*) and from Figure: 2.5 we see that the SINR for all the links converges to the threshold SINR.

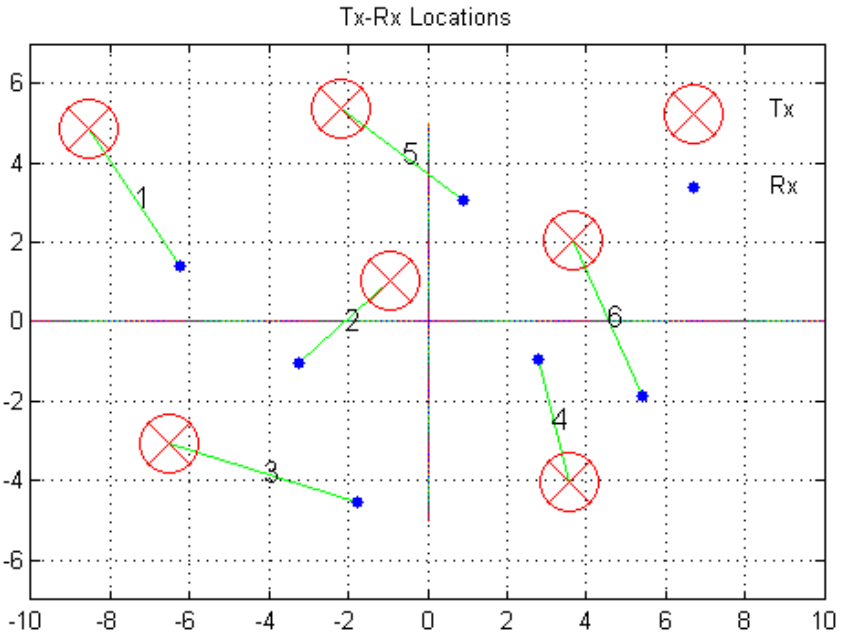

Figure 2.1: Tx-Rx Locations

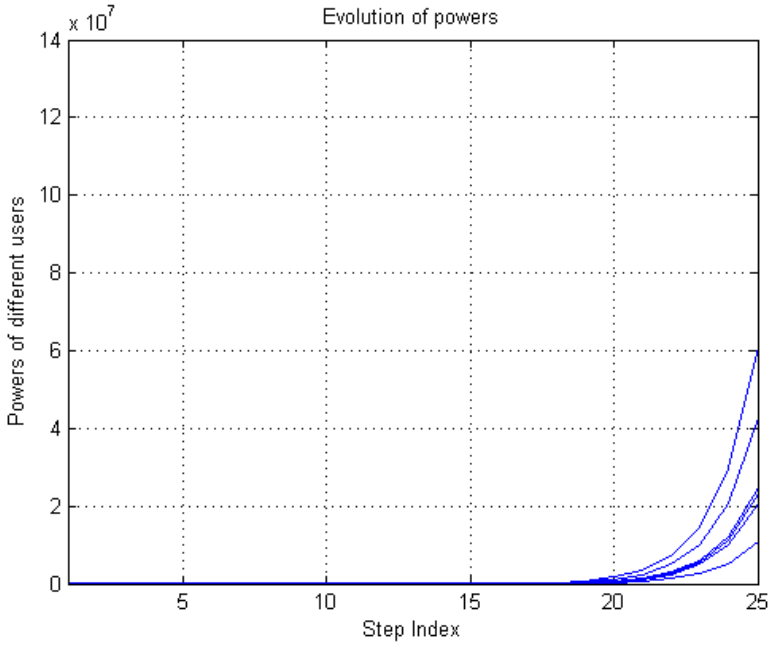

Figure 2.2: Threshold SINR infeasible scenario

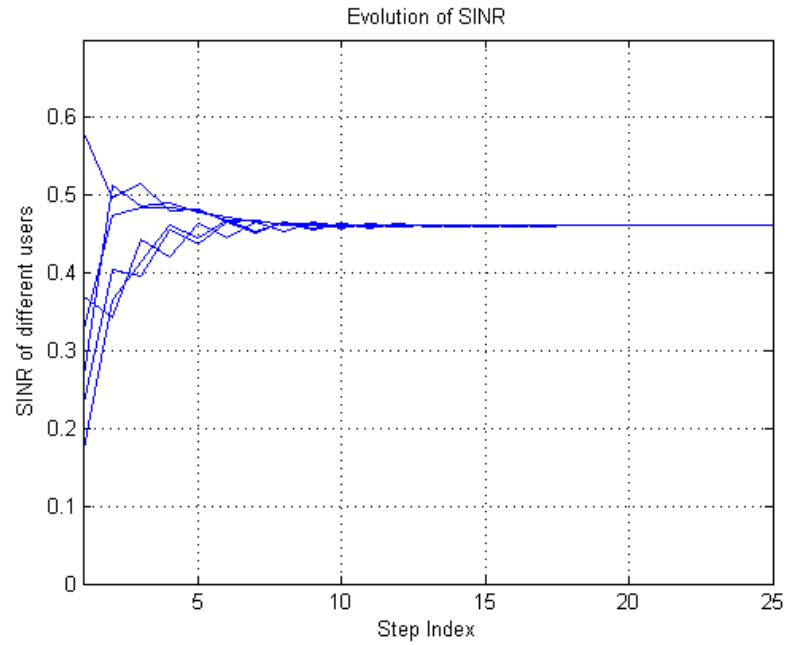

Figure 2.3: Threshold SINR infeasible scenario

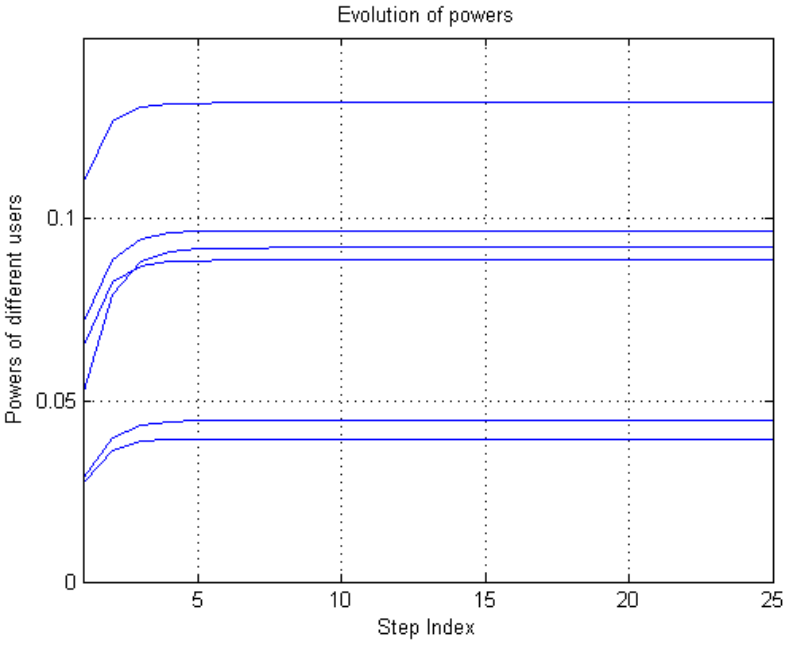

Figure 2.4: Threshold SINR feasible Scenario

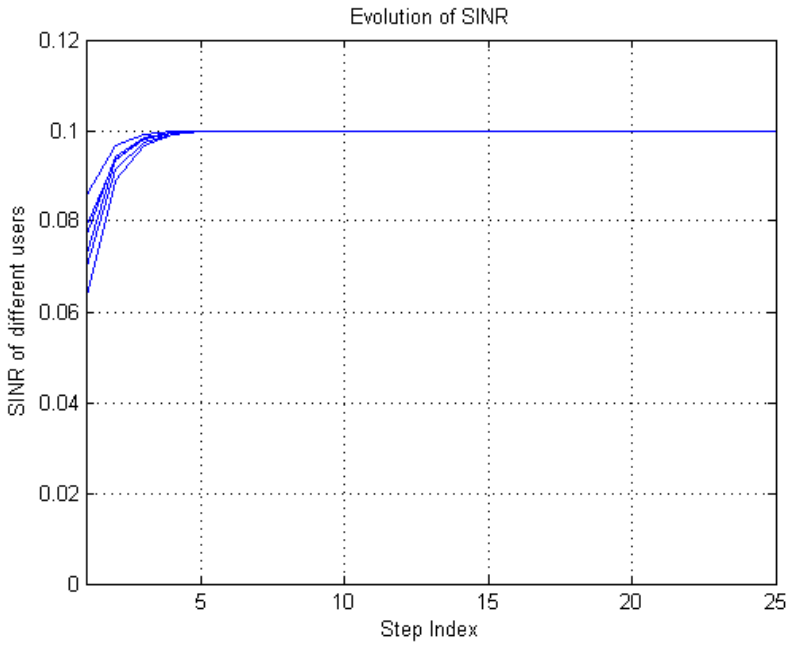

Figure 2.5: Threshold SINR feasible scenario

2.2 SINR Adaptation Model

In this channel model, the transmission rate is a concave function of `SINR`. The transmission rate for link i is given by the following expression

$$Rate_i[P] = C\ log\left(1 + SINR_i[P]\right), \text{where } C > 0 \tag{2.15}$$

2.3 Activation Vector

For both the channel models sharing of channel in time between the different users could be necessary so as to achieve data rates that are not possible otherwise. For a

network of N links we have $2^N - 1$ possible non-idle transmission states. Define a set $E = \{0,1\}^N \backslash (0,\ 0,\ ...,0)$ that will be used to denote the set of all non-idle transmission states. An element $e \in E$ will be called an activation vector and is a $0-1$ vector of length N such that the i^{th} component of e, e_i is 1 if link i is transmitting in this state and 0 otherwise. We will also refer the activation vector e as a set e such that $i \in e$ iff $e_i = 1$. The power vector corresponding to an activation vector will be denoted as $P^e = [P_1^e\ , P_2^e\ ....,\ P_N^e]$ where P_i^e is the transmission power of link i while in activation vector e. The transmission power P_i^e satisfies the following constraint $P_i^e \in \mathcal{P}$ if $i \in e$ and $P_i^e = 0$ if $i \notin e$.

So an activation vector partitions a network into many sub-networks. We can extend the ideas related to the SINR threshold model for a network to these sub-networks as follows

2.3.1 Activation Vector in SINR Threshold Model

We define a feasible activation vector as an activation vector $e \in E$ such that there exists a power vector P^e that satisfies $SINR_i[P^e] \geq \beta$ and $P_i^e \in \mathcal{P}\ \forall i \in e$ and $P_i^e = 0\ \forall i \notin e$. Then similar to the above section, we define P^{e*} as the optimal power vector for activation vector e. So $\texttt{SINR}_i[P^{e*}] = \beta\ \forall i \in e$. Then we have the condition that the activation vector e is feasible iff $P_i^{e*} > 0\ \forall i \in e$. Also β_0^e will denote the maximum achievable maximum common value of SINR for all links in e that can be attained by using any power vector P^e.

CHAPTER 3

RATE STABILITY AND CAPACITY REGIONS

In this chapter, we discuss the notion of stability that we consider and also the capacity region for the SINR models that under the three different resource allocation schemes.

3.1 Stability of Queues

The criteria for stability we consider is rate stability. According to this criterion of stability, the queue of link i is said to be stable if the time average service rate of link i denoted as S_i satisfies the condition, $S_i > \lambda_i$. In this report we will consider the fluid model for queues, meaning that the data can be split into packets of arbitrarily small units send across the channel. We will also assume that the arrival process is finitely bounded. Using these assumptions, it was shown in (Chaporkar and Proutiere, 2013) that the queue length of user i, $Q(i)$ will hit zero infinitely often if link i is rate stable.

3.2 Capacity Regions

In this section we present the capacity region for the two SINR models that we consider under the three different resource allocation schemes. The proof of these arguments follows from the condition of rate stability that requires us to be able to find parameters that make the time average service rate for a link strictly greater than its arrival rate.

3.2.1 SINR Threshold Model

Power Control

The condition on the *network configuration*, *arrival rate vector* λ and the power constraint set $\mathcal{P}$ is the following

$$\exists\ P \text{ such that}$$

$$SINR_i[P] \geq \beta,\ \forall i \in \{1, 2, ..., N\} \tag{3.1}$$

$$P_i \in \mathcal{P}, \forall i \in \{1, 2, ..., N\} \tag{3.2}$$

$$\lambda_i < 1,\ \forall i \in \{1,\ 2,\ ...,\ N\} \tag{3.3}$$

Scheduling

Then the conditions on the *network configuration*, *arrival rate vector* λ and the power constraint set $\mathcal{P}$ is the following

$$\exists\ E(f) = \{e^1, e^2, ..., e^K\} \subseteq E$$

$$\text{such that } \forall e^k \in E(f) \text{ and with } P^{e^k} = P_t\ [I_{\{e_1^k=1\}}, I_{\{e_2^k=1\}}, ..., I_{\{e_N^k=1\}}]^T$$

$$\text{we have ,}$$

$$SINR_i[P^{e^k}] \geq \beta, \forall i \in e^k \tag{3.4}$$

$$P_t \in \mathcal{P} \tag{3.5}$$

$$\exists \mu = [\mu_1, \mu_2, ..., \mu_K]^T$$

such that

$$\lambda_i = \sum_{k=1}^{K} \mu_k \, I_{\{e_i^k = 1\}}, \; \forall i \in \{1, 2, ..., N\} \tag{3.6}$$

$$0 \leq \mu_k < 1, \; \forall k \in \{1, 2, ..., K\} \tag{3.7}$$

$$\sum_{k=1}^{K} \mu_k < 1 \tag{3.8}$$

μ_k is the fraction of time activation vector $e^k \in E(f)$ will be used by the network.

Joint Power Control and Scheduling

The the conditions on the *network configuration*, *arrival rate vector* and the power constraint set $\mathcal{P}$ is the following

$$\exists \; E(f) = \{e^1, e^2, ..., e^K\} \subseteq E$$

such that $\forall e^k \in E(f)$, $\exists \; P^{e^k}$ such that

$$SINR_i[P^{e^k}] \geq \beta, \forall i \in e^k \tag{3.9}$$

$$P_i^{e^k} \in \mathcal{P} \;\; \forall e^k \in E(f) \text{ and } \forall i \in e^k \tag{3.10}$$

$$\exists \; \mu = [\mu_1, \mu_2, ..., \mu_K]^T$$

such that

$$\lambda_i = \sum_{k=1}^{K} \mu_k \, I_{\{e_i^k = 1\}}, \; \forall i \in \{1, 2, ..., N\} \tag{3.11}$$

$$0 \leq \mu_k < 1, \forall k \in \{1, 2, ..., K\} \tag{3.12}$$

$$\sum_{k=1}^{K} \mu_k < 1 \tag{3.13}$$

3.2.2 SINR Adaptation Model

Power Control

$$\exists\ P \text{ such that}$$

$$\lambda_i < Rate_i[P], i = 1, 2, ..., N \tag{3.14}$$

$$P_i \in \mathcal{P}, \forall i \in \{1, 2, ..., N\} \tag{3.15}$$

Scheduling

$$\exists\ E(f) = \{e^1, e^2, ..., e^K\} \subseteq E$$

$$\text{and } \exists \mu = [\mu_1, \mu_2, ..., \mu_K]^T$$

such that

$$\forall e^k \in E(f) \text{ and with } P^{e^k} = P_t\ [I_{e_1^k=1}, I_{e_2^k=1}, ..., I_{e_N^k=1}]$$

we have

$$\lambda_i = \sum_{k=1}^{K} \mu_k\ Rate_i[P^{e^k}],\ i = 1, 2, ..., N \tag{3.16}$$

$$0 \leq \mu_k < 1, \forall k \in \{1, 2, ..., K\} \tag{3.17}$$

$$\sum_{k=1}^{K} \mu_k < 1 \tag{3.18}$$

μ_k is the fraction of time activation vector $e^k \in E(f)$ will be used by the network.

Joint Power Control and Scheduling

$$\exists\ E(f) = \{e^1, e^2, ..., e^K\}$$

$$\text{and } \exists\ P^{e^k}, \text{ and } \exists \mu = [\mu_1, \mu_2, ..., \mu_K]^T$$

$$\text{such that}$$

$$\lambda_i = \sum_{k=1}^{K} \mu_k\ Rate_i[P^{e^k}],\ i = 1, 2, ..., N \tag{3.19}$$

$$P_i^{e^k} \in \mathcal{P}\ \forall i \in e^k and \forall e^k \in E(f) \tag{3.20}$$

$$0 \leq \mu_k < 1,\ \forall k \in \{1, 2, ..., K\} \tag{3.21}$$

$$\sum_{k=1}^{K} \mu_k < 1 \tag{3.22}$$

μ_k is the fraction of time activation vector $e^k \in E(f)$ will be used by the network.

CHAPTER 4

DISTRIBUTED ALGORITHM FOR SINR THRESHOLD MODEL UNDER JOINT POWER CONTROL AND SCHEDULING

4.1 Throughput Optimal Resource Allocation

In this section, we discuss how, from the capacity region conditions, a distributed throughput optimal algorithm can be obtained for the SINR threshold model under joint power control and scheduling. From the capacity region, we see that to be able to support an arrival process with arrival rate vector λ, we are required to solve a *power problem* and a *scheduling problem*. In this section we describe how these two problems can be solved.

4.1.1 Power Problem

Power problem constitutes of finding the set of all feasible activation vectors $E(f) = \{e^1, e^2, ..., e^K\}$ and a power vector P^{e^k} that satisfies the SINR constraints for all the users in activation vector e^k simultaneously. However using Theorem:1 the feasibility or infeasibility of activation vector is decided by the *optimal power vector* P^{e^k*}. Hence by executing FM-PCA for a link i in activation vector $e \in E$, we can determine if the activation vector is feasible or not as follows

- Power P_i^e converges to a constant P_i^{e*} and SINR converges to β. In this case activation vector e is feasible.

- Power P_i^e diverges and SINR converges to $\beta_0^e < \beta$. In this case activation vector e is infeasible.

4.1.2 Scheduling Problem

The scheduling problem constitutes finding a parameters μ that will make the time average service rate for a link i greater than its time average arrival rate. We first formulate a Linear Program whose optimal point will be shown to satisfy this service rate requirement for any arrival rate vector in the capacity region.

Consider the following *Linear Program (LP)* in variables $\nu = [\nu_1,\ \nu_2,\ ...,\ \nu_k]^T$

$minimize$

$$L(\nu) = \sum_{k=1}^{K} \nu_k$$

$subject\ to$

$$s_i \geq \lambda_i\ ,\ i = 1, 2, ..., N$$

$$\nu_k \geq 0\ ,\ k = 1, 2, ..., K$$

where, $s_i = \sum_{k=1}^{K} \nu_k\, I_{\{e_i^k = 1\}}\ ,\ i = 1, 2, ..., N$

Let ν^* be the optimal point for the above *LP*.

In the above formulation ν_k is the duration of time feasible activation vector $e^k \in E(f)$ will be used by the network exactly once before being taken up again. If the channel is

never left idle, then we have the following relation between the variable μ_k and ν_k

$$\mu_k = \frac{\nu_k}{\sum_{l=1}^{K} \nu_l} \tag{4.1}$$

The term s_i will be called *virtual service rate* of link i and it is related to the time average service rate S_i as

$$S_i = \frac{s_i}{\sum_{l=1}^{K} \nu_l} \tag{4.2}$$

Theorem 2 *If the arrival rate vector λ lies in the capacity region, then at ν^* we have $S_i > \lambda_i \, , \forall i \in \{1, 2, ..., N\}$.*

Proof 2 *Since the arrival rate vector λ lies in the capacity region, we have from capacity region equation a point $\mu' = [\mu'_1 \, , \mu'_2 \, , ..., \mu'_K]$ that satisfies ,*

$\lambda_i = \sum_{k=1}^{K} \mu'_k \, I_{\{e_i^k = 1\}}, \; \forall i \in \{1, 2, ..., N\} \implies s_i = \lambda_i \, , \forall i \in \{1, 2, ..., N\}$

$0 \leq \mu'_k < 1, \; \forall k \in \{1, 2, ..., K\} \implies \mu'_k \geq 0 \, , \forall k \in \{1, 2, ..., K\}$

From the above two conditions, we see that the point μ' lies in the feasible set of the Linear Program.

Now since ν^* is the optimal point of the LP, we have

$\sum_{k=1}^{K} \nu_k^* \leq \sum_{k=1}^{K} \mu'_k < 1 \implies \sum_{k=1}^{K} \nu_k^* < 1$

Also we have $\sum_{k=1}^{K} \nu_k^* > 0$, or else $s_i \geq \lambda_i$ condition cannot be met for any link i with a strictly positive arrival rate λ_i.

Now we have at ν^*

$s_i \geq \lambda_i, \; \forall i \in \{1, 2, ..., N\}$

$$\implies S_i = \frac{s_i}{\sum_{l=1}^{K} \nu_l} > s_i \geq \lambda_i.$$

Hence to solve the *LP*, every link tries to minimize their channel usage while maintaining their virtual service rate greater than or equal to their arrival rate.

Theorem 3 *Any arrival rate λ in the capacity region can be supported by using at most N activation vectors.*

Proof 3 *Similar to Theorem: 1 we can convert the LP into standard form and then the result follows from (Luenberger and Ye, 2008, Section 2.4)*

It turns out that the above LP cannot be solved directly in a distributed manner. Hence, to find the optimal point ν^* in a distributed manner, we formulate the following unconstrained convex optimization problem parameterized by a scalar Θ. This convex program is an approximation to the above linear program. $\Theta > 0$ is called the penalty factor and the approximation gets better as $\Theta \downarrow 0$.

Barrier Program

$minimize$

$$L(\nu, \Theta) = \sum_{k=1}^{K} \nu_k - \Theta \sum_{i=1}^{N} log(s_i - \lambda_i) - \Theta \sum_{k=1}^{K} log(\nu_k)$$

where, $s_i = \sum_{k=1}^{K} \nu_k \, I_{\{e_i^k = 1\}}$, $i = 1, 2, ..., N$ and $\Theta > 0$.

Let $\nu^*(\Theta)$ be the solution to *Barrier Program*. From (Boyd and Vandenberghe, 2004, Section 11.2.1), we have the relation,

$$\nu^* = \lim_{\Theta \downarrow 0} \nu^*(\Theta) \tag{4.3}$$

We show later that the point $\nu^*(\Theta)$ can be reached in a distributed manner as the gradient of the objective function $L(\nu, \Theta)$ can be computed in a distributed manner thus enabling us to solve the scheduling problem in a distributed manner by choosing an appropriately small enough value for penalty factor Θ.

Solving barrier program using gradient descent

As the objective function in barrier program, $L(\nu, \Theta)$ is a convex function, its optimal point can be reached using gradient descent. The gradient for the objective function $L(\nu, \Theta)$ is given by

$$\nabla_\nu L(\nu, \Theta) = \left[\frac{\partial L(\nu, \Theta)}{\partial \nu_1}, \frac{\partial L(\nu, \Theta)}{\partial \nu_2}, ..., \frac{\partial L(\nu, \Theta)}{\partial \nu_K}\right]^T \tag{4.4}$$

where

$$\frac{\partial L(\nu, \Theta)}{\partial \nu_k} = 1 - \Theta\frac{1}{\nu_k} - \Theta\sum_{i=1}^{N}\frac{I_{\{e_i^k=1\}}}{s_i - \lambda_i}, \;\; k = 1, 2, ..., K \tag{4.5}$$

The gradient descent algorithm for finding the parameters ν works as follows. Every discrete time slot indexed by t, update the parameter ν as

$$\nu(t+1) \leftarrow \nu(t) - \alpha(t)\, \nabla_{\nu(t)}\, L(\nu(t), \Theta) \tag{4.6}$$

Where $\alpha(t),\ t = 1, 2, ...$ is a sequence such that $\alpha(t) > 0$, $\sum_{t=1}^{\infty}\alpha(t) = \infty$, $\sum_{k=1}^{\infty}\alpha(t)^2 < \infty$.

4.2 Distributed Algorithm

In this section we present our distributed algorithm for the SINR threshold model under *joint power control and scheduling*. We assume that there is a feedback channel from

the receiver to the transmitter that reports the SINR at the receiver and that this feedback is instantaneous. We also assume that the receiver will be able to detect any infinitesimal change in interference in the network. We also assume that the propagation delay, sensing delay etc is zero. We also assume that the transmitter i knows its own arrival rate λ_i.

We describe our algorithm below

The Distributed algorithm has the following phases.

A Initialization Phase (IP).

B Activation Vector and Transmit Power Identification Phase (AVTPIP).

C Activation Vector Indexing Phase (AVIP).

D Data Frame Phase (DFP).

4.2.1 Initialization Phase

In this phase, the objective is to establish a unique index for all the links in the network and also to identify the total number of links in the network. We present an algorithm that every link will execute in the beginning. We assume that time is divided into slots of length σ. The odd slots will be used for capturing the channel in order to obtain an index for the link. A link that has not been indexed so far, will choose to transmit a message to all the links in the network with some probability $p > 0$. The other links in the network will listen to this message and increment the number of indexed links. The index of the link will be the order in which it's timer fired with respect to other links. In case of a collision in an odd slot, the links that are involved in the collision will use the next even slot and transmit a message to inform other links in the network about the collision so that they can decrement the number of indexed links. We also keep an upper limit on the number of consecutive idle odd slots W, so that all the unindexed links will

transmit message in the W^{th} consecutive odd slot with probability 1. So W consecutive idle slots would indicate the links to exit the IP.

Assume that the update occurs at discrete time slots indexed by $t = 1, 2, ...$ Link i executes the following algorithm.

IP ALGORITHM

Link i initializes the following

- The number of Links that hasn't been indexed so far, $index = 0$
- Link i indexed status, $indexed_i = No$
- The no. of consecutive odd idle slots before the current odd slot, $idle\_slots = 0$
- The variable that indicates if Link i has suffered a collision in the odd slot, $collide = 0$

every time slot t, do

1) **if** t is an *odd* slot

1.1) **if** $indexed_i = No$. Link i hasn't been indexed so far and will try to use the channel in this slot with probability p given by

$$p = \frac{idle\_slots + 1}{W}$$

1.1.1) Choose $U_i = unif(0, 1)$. Link i chooses the value of random variable U_i uniformly at random from $(0, 1)$ interval.

1.1.2) **if** $U_i \leq p$. Link i will then use the channel in the current time slot to get an index.

1.1.2.1) Transmit *Intent*

1.1.2.2) Update the following

$$idle\_slots = 0\ ,\quad index = index + 1$$

1.1.2.3) **if** *collision* occurs. Two or more links tried to use the channel.

1.1.2.3.1) Update

$$collide = 1\ ,\quad index = index - 1$$

1.1.2.4) **if** *collision* did not occur. Indexing is done in the current time slot.

1.1.2.4.1) Update

$$Link\_index_i = index\ , \quad indexed = Yes$$

1.1.3) **if** $U_i > p$. Link i will not use the channel in the current time slot to get an index.

1.1.3.1) **if** channel is *Busy*. At least one link in the network tried to get an index in the current time slot.

1.1.3.1.1) Update the following

$$index = index + 1\ , \quad idle\_slots = 0$$

1.1.3.2) **if** channel is *Idle*. None of the links that are unindexed so far tried to get an index in the current time slot.

$$idle\_slots = idle\_slots + 1$$

1.2) **if** $indexed_i = Yes$. Link i has been indexed.

1.2.1) **if** channel is *Busy*. At least one link in the network tried to get an index in the current time slot.

1.2.1.1) Update the following

$$idle\_slots = 0\ , \quad index = index + 1$$

1.2.2) **if** channel is *Idle*. None of the links that are unindexed so far tried to get an index in the current time slot.

1.2.2.1) Update the following

$$idle\_slots = idle\_slots + 1$$

1.2.2.2) **if** $idle\_slots = W$

1.2.2.2.1) Set

$$N = index$$

1.2.2.2.2) Exit **IP**

2) **if** t is an *even* slot

2.1) **if** $collide = 1$. Link i encountered a collision in the previous odd slot.

2.1.1) Transmit *Intent*

2.1.2) Update

$$collide = 0$$

2.2) **if** $collide = 0$. Link i did not encounter a collision in the previous odd slot.

2.2.1) **if** channel is *Busy*

2.2.1.1) Update

$$index = index - 1$$

After the execution of this algorithm, link i will have a unique index given by $Link\_index_i$ and all links will infer the value of N. They would then execute the next phase in the distributed algorithm.

4.2.2 Activation Vector and Transmit Power Identification Phase

For $E(f) \subseteq E$ define for $i \in \{1, 2, ..., N\}$ a set $E_i(f) \subseteq E(f)$ such that $E_i(f) = \{e \in E(f) | e_i = 1\}$. So $E_i(f)$ is the set of all feasible *activation vectors* that link i is contained in. In this phase, every link i in the network tries to figure out the set $E_i(f)$ and also the transmission power P_i^e, $\forall e \in E_i(f)$. For $m \in \{0, 1, ..., 2^N - 1\}$, $\{m\}_2$ will denote the N length binary equivalent of decimal number m. Assume that the update occurs at discrete time slots indexed by $t = 1, 2, ...$ We will abuse the notation and let $SINR_i[t]$ and $I_i[t]$ to denote $SINR_i[P(t)]$ and $I_i[P(t)]$ where P(t) is the power vector at time t. Link i executes the following algorithm.

AVTPIP ALGORITHM

Link i initializes the following

- *activation vector* $e = \{m\}_2$, where $m = 2^N - 1$
- The transmission power of Link i at the beginning of every slot, $p_i = 0$

- The initial value of $SINR$ at the receiver of link i, $SINR_i[0] = 0$

every time slot t, do

1) **if** t is an odd slot

1.1) **if** link i is contained in *activation vector* e

1.1.1) Update power p_i as

$$p_i \leftarrow (1-\epsilon)p_i + \epsilon\, p_i \frac{\beta}{SINR_i[t-1]}$$

1.1.2) Transmit with power p_{ei}

1.1.3) Update $I_i[t]$, $SINR_i[t]$ from channel

1.1.4) **if** $\{I_i[t]\}$ has converged. *Activation vector* e is feasible

1.1.4.1) Update the following

$$exit_i = 1$$

$$P_i^e = p_i\ , \quad p_i = 0\ , \quad E_i(f) = E_i(f) \cup e$$

1.1.5) **if** $\{I_i[t]\}$ has not converged but $\{SINR_i[t]\}$ has converged. *Activation vector* e is infeasible

1.1.5.1) Update the following

$$p_i = 0\ , \quad exit_i = 1$$

1.2) **if** Link i is not contained in *activation vector* e

1.2.1) Update $I_i[t]$ from channel

2) **if** t is an even slot

2.1) **if** Link i is contained in *activation vector* e

2.1.1) Transmit with power p_i

2.1.2) **if** $exit_i = 0$

2.1.2.1) Update $I_i[t]$ from channel

2.1.2.2) **if** $I_i[t]$ is less than $I_i[t-1]$

2.1.2.2.1) **if** $SINR_i[t-1] \geq \beta$. *Activation vector* e is feasible

2.1.2.2.1.1) Update the following

$$P_i^e = p_i \ , \quad p_i = 0 \ , \quad E_i(f) = E_i(f) \cup e$$

2.1.2.2.2) **if** $SINR_i[t-1] < \beta$. *Activation vector* e is infeasible

2.1.2.2.2.1) Update the following

$$p_i = 0$$

2.1.2.3) Update the following

$$m = m - 1 \ , \quad e = \{m\}_2 \ , \quad SINR_i[t] = 0$$

2.1.3) if $exit_i = 1$

2.1.3.1) Update the following

$$exit_i = 0$$

$$m = m - 1 \ , \quad e = \{m\}_2 \ , \quad SINR_i[t] = 0$$

2.2) **if** Link i is not contained in *activation vector* e

2.2.1) Update $I_i[t]$ from the channel

2.2.2) **if** $I_i[t]$ is less than $I_i[t-1]$

2.2.2.1) Update the following

$$m = m - 1 \ , \quad e = \{m\}_2 \ , \quad SINR_i[t] = 0$$

2.3) **if** $m = 0$

2.3.1) Exit **AVTPIP**

4.2.3 Activation Vector Indexing Phase

In this phase, the objective is to figure out the set $E(f)$ from the knowledge of the set $E_i(f), \ \forall i \in \{1, 2, ..., N\}$. Assume that the update occurs at discrete time slots indexed by $t = 1, 2, ...$ Link i executes the following algorithm.

AVIP ALGORITHM

Link i initializes the following

- *activation vector* $e = \{m\}_2$, where $m = 2^N - 1$
- set of all feasible activation vectors $E(f) = \phi$

every time slot t**, do**

1) **if** Link i is contained in *activation vector* e
 1.1) **if** $e \in E_i(f)$. *Activation vector* e is feasible, and hence Channel will taken up by the Link.
  1.1.1) Transmit using Power P_i^e
  1.1.2) Update the following as

$$k = k + 1\ , \quad e^k = e\ , \quad E(f) = E(f) \cup \{e^k\}$$

2) **if** Link i is not contained in *activation vector* e
 2.1) **if** channel is *Busy*. This indicates that *activation vector* e is feasible.
  2.1.1) Update the following as

$$k = k + 1\ , \quad e^k = e\ , \quad E(f) = E(f) \cup \{e^k\}$$

3) Update the following as

$$m = m - 1\ , \quad e = \{m\}_2$$

4) **if** $m = 0$
 4.1) set $K = k$
 4.2) Exit **AVIP**

4.2.4 Data Frame Phase

After executing the Initialization Phase, Activation Vector and Transmit Power Identification Phase, Activation Vector Indexing Phase, link i becomes aware of the set $E(f)$

and their corresponding optimal Power P_i^{e*} corresponding to each of these activation vectors $e \in E(f)$. Now they must figure out the parameters $\nu_k^*, \forall k \in \{1, 2, ..., K\}$. We will call *data frame* as the period of time during which all the activation vectors $e \in E(f)$ will be executed once sequentially. In a *data frame*, one link will update the parameters $\nu_k, \forall k \in \{1, 2, ..., K\}$ which will be observed from the channel by other links and stored. After N *data frames*, the links update the parameters based on the updated parameters in previous N *data frames*.

Link i executes the following algorithm

DFP ALGORITHM

Link i initializes the following parameters

$$m = 0\ , \quad Big\_\mu = [1, 1, ..., 1]\ , \quad s_i = |E_i(f)|$$

$$scale\_1 = 1\ , \quad scale\_2 = 1\ , \quad \Theta = \Theta_0$$

for each *data frame* m**, do**

1) Update the following

$$m = m + 1$$

2) Deterministically choose a Link to update the parameters in the current *data frame* whose index is idx, given by

$$idx = 1 + mod(m - 1, N)$$

3) Link i now proceeds to transmit data in the *data frame* according to the following rule

3.1) **if** $idx = Link\_index_i$. Link i is responsible for updating the parameter ν in the current *data frame*

for each $e^1, e^2, ..., e^K$ in $E(f)$

- Transmit Data with Power $P_i^{e^k}$ for duration ν_k, where

$$\nu_k = Big\_u_k - \frac{\epsilon_1}{scale\_1}(1 - \frac{\Theta}{Big\_u_k} - \frac{\Theta\, N}{(s_i - \lambda_i)}\, I_{\{e_i^k=1\}}) \tag{4.7}$$

- Wait for σ amount of time
- Store

$$u_k^{idx} = \nu_k$$

end for

3.2) **if** $idx \neq Link\_index_i$. Link i is not responsible for updating the parameter ν in the current *data frame*

for each $e^1, e^2, ..., e^K$ in $E(f)$

- Transmit Data with Power $P_i^{e^k}$ for duration $\hat{\nu}_k$, where

$$\hat{\nu_k} = Big\_u_k - \frac{\epsilon_1}{scale\_1}(1 - \frac{\Theta}{Big\_u_k}) \tag{4.8}$$

- Observe ν_k from the Channel
- Wait for σ amount of time
- Store

$$u_k^{idx} = \nu_k$$

end for

4) **if** $idx = N$. After every N *data frames*, the parameter Big_u is updated as follows

$$Big\_u = (1 - \epsilon_2)\, Big\_u + \epsilon_2\, \frac{1}{N}(\sum_{i=1}^{N} u^i) \tag{4.9}$$

$$s_i = \sum_{k=1}^{K} Big\_u_k\, I_{\{e_i^k=1\}}$$

$$scale\_1 = scale\_1 + 1$$

5) **if** $mod(m-1, MN) = MN-1$. After every MN *data frames* check if convergence has occurred.

if Big_u has converged. The penalty factor Θ is updated as follows

$$scale\_1 = 1$$

$$scale\_2 = scale\_2 + 1$$

$$\Theta = \frac{\Theta_0}{scale\_2}$$

Theorem 4 *The distributed algorithm makes the parameters ν converge to ν^*.*

Proof 4 *Plug Eq:* (4.7) *into Eq:* (4.9) *through Eq:* (4.8) *and we have, after* N data frames*, the k^{th} component of Big_u, Big_u_k updated as*

$$Big\_u_k \leftarrow (1-\epsilon_2)\, Big\_u_k +$$

$$\frac{\epsilon_2}{N}(\sum_{i=1}^{N} Big\_u_k - \frac{\epsilon_1}{scale\_1}(1 - \frac{\Theta}{Big\_u_k} - \frac{\Theta\, N}{(s_i - \lambda_i)}\, I_{\{e_i^k=1\}})) \tag{4.10}$$

$$\implies Big\_u_k \leftarrow Big\_u_k -$$

$$\frac{\epsilon_2\ \epsilon_1}{scale\_1}\,(1 - \frac{\Theta}{Big\_u_k} - \sum_{i=1}^{N}\frac{\Theta}{(s_i - \lambda_i)}\, I_{\{e_i^k=1\}}) \tag{4.11}$$

From the above update equation, we can see that the distributed algorithm behaves similar to gradient descent algorithm that minimizes the Loss function $L(\nu, \Theta)$. So the distributed algorithm drives the parameter Big_u to $\nu^*(\Theta)$. After MN data frames *the*

parameter Θ is decremented if the variable Big_u has converged. Hence for appropriately small step size parameters for the gradient descent algorithm, we have as $\Theta \downarrow 0$, $\nu^*(\Theta) \rightarrow \nu^*$ as desired.

CHAPTER 5

DISTRIBUTED ALGORITHM FOR SINR ADAPTATION MODEL UNDER SCHEDULING

In this chapter we show how the framework proposed developed for the SINR threshold model can be used for the SINR adaptation model under scheduling.

In this physical layer model, as the rate is a continuous function of $SINR$, we will be able to get a non-zero transmission rate for any *activation vector* $e \in E$. Hence we will consider each of the *activation vectors* $e \in E$ to be a feasible activation vector. Let $e^1, e^2, .., e^K$ be the elements of E where $K = 2^N - 1$. For this channel model, as the users will be transmitting at fixed power levels, we only have to solve the scheduling problem. Similar to the SINR threshold model, we formulate the problem of finding a point μ that makes all the queues rate stable from the knowledge of the arrival rate vector λ as an LP. A barrier problem is then formulated which is then solved using gradient descent algorithm in a distributed manner. We highlight only the key steps in this section.

5.1 Scheduling Problem

The problem of finding a parameter μ that makes all the links from the knowledge of the arrival rate vector λ can be posed as an optimization problem given below. Then a barrier problem can be formulated which is then solved in a distributed manner.

$minimize$

$$L(\nu) = \sum_{k=1}^{K} \nu_k$$

$subject\ to$

$s_i \geq \lambda_i\ ,\ i = 1, 2, ..., N$

$\nu_k \geq 0\ ,\ k = 1, 2, ..., K$

where, $s_i = \sum_{k=1}^{K} \nu_k\ Rate_i[P^{e^k}]$ where $P^{e^k} = P_t\ [I_{\{e_1^k=1\}}, I_{\{e_2^k=1\}}, ..., I_{\{e_N^k=1\}}]^T$ and $Rate_i[P]$ is computed according to Eq: (2.15).

5.2 Distributed Algorithm

We know describe the distributed algorithm below

5.2.1 Initialization Phase

The links execute the IP algorithm.

5.2.2 Transmission Rate Identification Phase

In this phase the objective is to find the transmission rate $Rate_i[P^{e^k}]$ a link gets while in activation vector e^k. In this phase we have $2^N - 1$ time slots, one corresponding to each of the activation vectors $e^k \in E$. In the k^{th} time slot, links in activation vector

e^k will transmit with power P_t and other links will remain silent. So in the k^{th} time slot, each transmitter can figure out the parameter $Rate_i[P^{e^k}]$ by knowing the value of $SINR_i[P^{e^k}]$ which is fed back by its corresponding receiver.

5.2.3 Data frame phase

The links execute the DFP algorithm similar to the `SINR` threshold case but here the difference will be that the virtual service rate for link i, s_i will be calculated as $s_i = \sum_{k=1}^{K} \nu_k \; Rate_i[P^{e^k}]$. The rest of the algorithm and proof remains same.

CHAPTER 6

SIMULATIONS

Simulations were done in MATLAB for the above distributed algorithm for the SINR threshold model. We consider a network of 10 links. Figure: 6.1 shows the location of links. The gain coefficients were computed using the standard path loss function $g_{ij} = \alpha\, {r_{ij}}^{-\eta}\ \forall\ i, j \in \{1, 2, ..., N\}$. Queue lengths plot vs time for the links is shown in Figure: 6.2. Note that $t = 0$ in the plot corresponds to the starting point for the *DFP*, with the initial queue lengths being non-zero due to the data arrivals during the first three phases of the algorithm. A loading factor of 0.99 was used for the simulation and the arrival rate was chosen randomly using this loading factor. Table: 6.1 lists the different parameters and results of the simulation.

Table 6.1: Simulation Parameters

Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
N	10	W	12	M	10
α	1	η	3	N_0	10^{-9}
Θ_0	5	ϵ_1	0.001	ϵ_2	1
σ	10^{-4}	β	1	ϵ	0.50
Parameter	**Value**				
$E(f)$	$\{6,8\}$ $\{6,7\}$ $\{5,10\}$ $\{5,8\}$ $\{5,7\}$ $\{3,5\}$ $\{1,10\}$ $\{1,9\}$ $\{1,8\}$ $\{1,7\}$ $\{1,4\}$ $\{1,3\}$ $\{1,2\}$ $\{1\}$ $\{2\}$ $\{3\}$ $\{4\}$ $\{5\}$ $\{6\}$ $\{7\}$ $\{8\}$ $\{9\}$ $\{10\}$				

From Figure: 6.2 we see that the queue length of users initially increases till the algorithm reaches the stopping criteria. From that point onwards the queue lengths stabilizes. This illustrates that our algorithm makes the queues bounded.

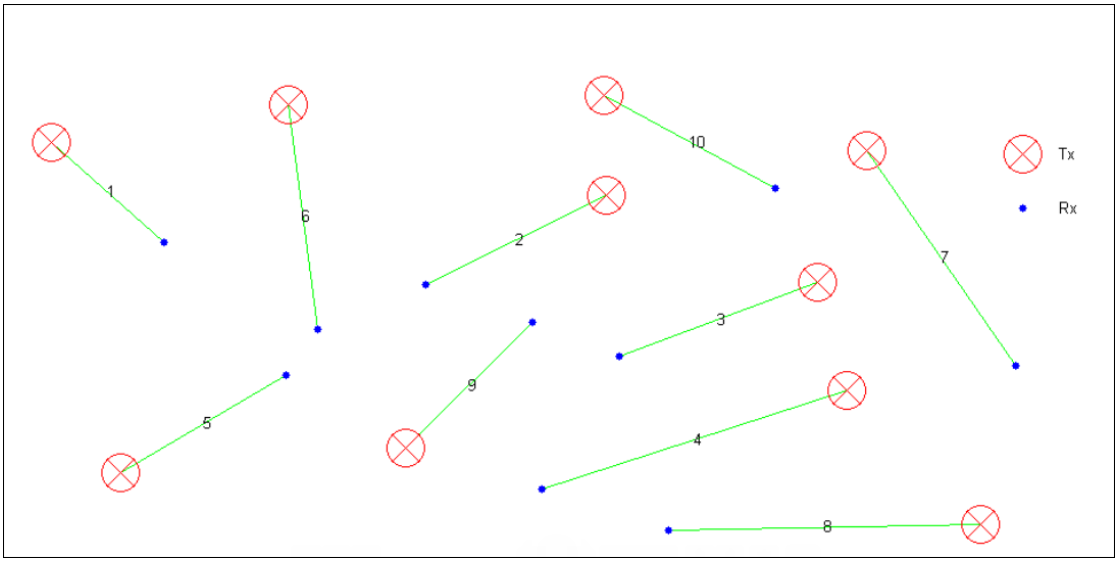

Figure 6.1: Link locations on a 20 $\times$ 12 grid

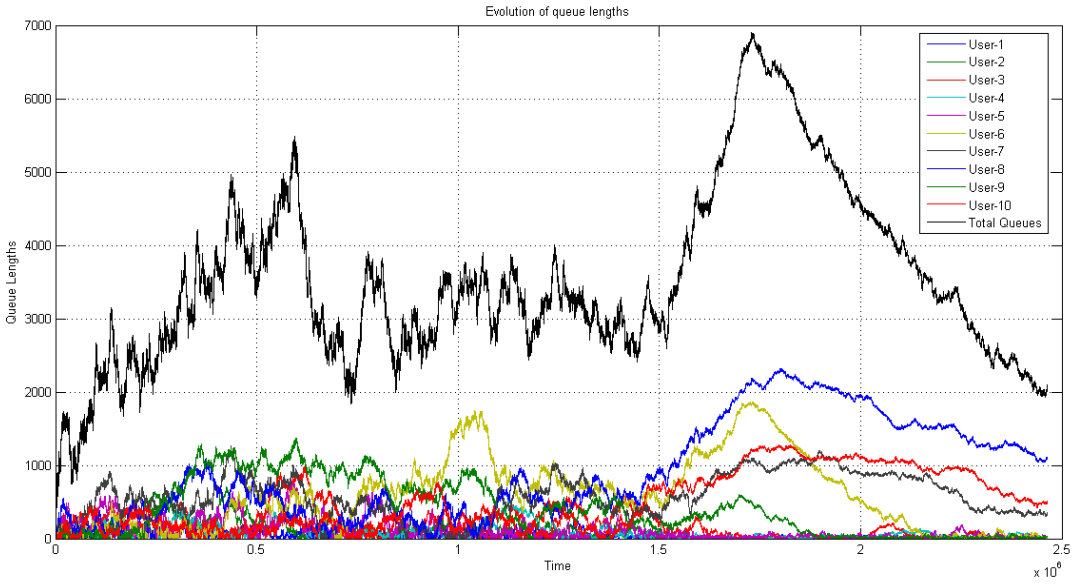

Figure 6.2: Queue length evolution

CHAPTER 7

CONCLUSION

In this thesis we proposed distributed throughput optimal algorithms for two SINR models. The first model we studied is the SINR threshold model. The proposed algorithm was proved to be throughput optimal under joint power control and scheduling scheme. We first formulated the capacity region and showed that to support an arrival process, we need to know quantities such as the set of all feasible activation vectors, the power vector (optimal) for the feasible activation vectors and the fraction of time each of the activation vectors must be used. We decoupled such a problem in that we first solved the problem of finding the optimal power vectors for each of the feasible activation vectors and then solved the problem of finding the fraction of time each of these activation vectors has to be used. A similar approach was used to propose a throughput optimal algorithm for second channel model under scheduling.

Although our algorithms require an exponential amount of time in number of users for convergence, we emphasize that throughput optimal algorithms are always NP-Hard in general and hence this complexity would be reflected in the queue lengths or in the computational complexity. One of the drawbacks of this work is that the algorithm needs to be restarted whenever a user joins/leaves the network. However if the timescale in which changes occur in the network is slow compared to the timescale required for convergence this will not be a major problem.

REFERENCES

1. **Boyd, S.** and **L. Vandenberghe**, *Convex Optimization*. Cambridge University Press, 2004.

2. **Chaporkar, P.** and **A. Proutiere** (2013). Optimal distributed scheduling in wireless networks under the sinr interference model. *http://arxiv.org/abs/1305.0384*.

3. **Foschini, G.** and **Z. Miljani** (1993). A simple distributed autonomous power control algorithm and its convergence. *IEEE Transactions on Vehicular Technology*, **42**(4), 641–646.

4. **Jiang, L. B.** and **J. Walrand** (2010*a*). A distributed csma algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Transaction on Networking*, **18**(3), 960–972.

5. **Jiang, L. B.** and **J. Walrand**, *Scheduling and Congestion Control for Wireless and Processing Networks*. Morgan & Claypool Publishers, 2010*b*.

6. **Lee, H. W.**, **E. Modiano**, and **L. B. Le** (2012). Distributed throughput maximization in wireless networks via random power allocation. *IEEE Transactions on Mobile Computing*, **11**(4), 577–590.

7. **Luenberger, D. G.** and **Y. Ye**, *Linear and Nonlinear Programming*. Springer, 2008.

8. **Ni, J.**, **B. Tan**, and **R. Srikant** (2012). Q-csma: Queue-length-based csma/ca algorithms for achieving maximum throughput and low delay in wireless networks. *IEEE/ACM Transaction on Networking*, **20**(3), 825–836.

9. **Tassiulas, L.** and **A. Ephremides** (1992). Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transaction on Automatic Control*, **37**(12), 1936–1948.

10. **Yi, Y.** and **G. D. Veciana** (2007). On optimal mac scheduling with physical interference. *in Proc. of 26th IEEE INFOCOM*.

11. **Zander, J.** (1992*a*). Distributed co-channel interference control in cellular radio systems. *IEEE Transactions on Vehicular Technology*, **41**.

12. **Zander, J.** (1992*b*). Performance of optimum transmitter power control in cellular radio systems. *IEEE Transactions on Vehicular Technology*, **41**(1), 57–62.