A Scalable Block-Proportional Fair Scheduling Scheme For Multi-Carrier Transmission Systems

A Project Report

submitted by

K. BHARATI

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **A Scalable Block-Proportional Fair Scheduling Scheme For Multi-Carrier Transmission Systems**, submitted by **K. Bharati (EE19E008), an exchange student from IIITDM Kanchipuram**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bonafide record of the research work done by her under my 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.

Prof. Krishna Jagannathan
Research Guide
Associate Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

First and foremost, I am extremely grateful to my Guru, Prof. Krishna Jagannathan for providing me this enjoyable platform to learn. His guidance and careful monitoring throughout, has definitely helped me understand and learn better. I am really fortunate that I had the kind association as well as supervision of Dr. Palani Maheswaran. I would like to sincerely thank my mother and father for everything that they have done for me. Without their constant encouragement and affection, everything would have been a distant dream. I take this opportunity to express my gratitude towards my grandfather for always believing in me and praying for my well-being. Last but not the least, I thank the God Almighty.

ABSTRACT

KEYWORDS: Wireless Communication, Multi-carrier Transmission, Dynamic Fair Proportional Scheduling, Scalability, Software Defined Network

One of the key features of a robust multi-carrier wireless transmission system is its ability to support a large number of users that communicate in the network through various devices and applications. The growing demand of the users need to be satisfied by ensuring good user experience. The data traffic patterns and channel conditions drastically affect the data rates of these users on multiple carriers. In order to guarantee uniform user experience, the distribution of resources by the scheduler among users must be proportionally fair. A review of prior works reveal that the solution to the **proportional fair scheduling** problem is computationally intensive and also, can not be extrapolated to account for an ever-increasing user base. Thus, the need is to obtain a scalable solution that will ensure fair proportionality among all its users. In this report, a scalable block-proportional fair scheduling algorithm and its variants for systems with multiple carriers are proposed. The proposed algorithm employs an optimal but time-consuming subroutine to build a close-to optimal scalable solution for the proportional fair scheduling problem. The software-driven solution can be implemented on a multi-core commodity hardware. The scalability has been demonstrated empirically for a multiple user-multiple carrier system by developing a parameterised MATLAB framework.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

PF	Proportional Fair
OFDMA	Orthogonal Frequency Division Multiple Access
BS	Base Station
HDR	High Data Rate
CSI	Channel State Information
SIR	Signal to Interference Ratio
QoS	Quality of Service
CU	Carrier-User Matrix
RR	Round Robin
SCPF	Single Carrier Proportional Fair
MCPF	Multiple Carrier Proportional Fair
ISP	Internet Service Provider

CHAPTER 1

INTRODUCTION

1.1 Challenges in Wireless Communication today

The past few decades have witnessed a tremendous bloom in the field of wireless communication systems. The trend is rapidly shifting towards developing systems that are reliable, resilient and robust. A multitude of challenges are encountered while attempting to build and design such systems.

Interference management is one primary challenge. Interference refers to the addition of unwanted signals to a user transmitted signal that tend to disrupt or modify the properties of the original signal. This affects the reliability and limits the throughput of a system to a great extent. Attempts have been made to study the nature and/or patterns of interfering signals to improve the performance of the wireless communication system. A wide variety of receiver designs like multi-user detection have been developed to try to suppress the interference. In addition to this, other methods including (but not limited to) increase in the transmit power, orthogonal frequency division multiple access have also been proposed.

Fading or the time variations in the signal strengths poses a big challenge and plays a very crucial role when it comes to link design. Fading is caused because of path loss due to attenuation, reflection from objects, presence of obstacles in the path and many more. The fading could also be small-scale, that is, over distances that are much less compared to the wavelength of the signal transmitted at some frequency, or large scale, that is, over distances that are much larger compared to the wavelength of the signal transmitted at some frequency. The links in both directions (transmitter to receiver and vice-versa) must be able to ensure reliable transmission of information.

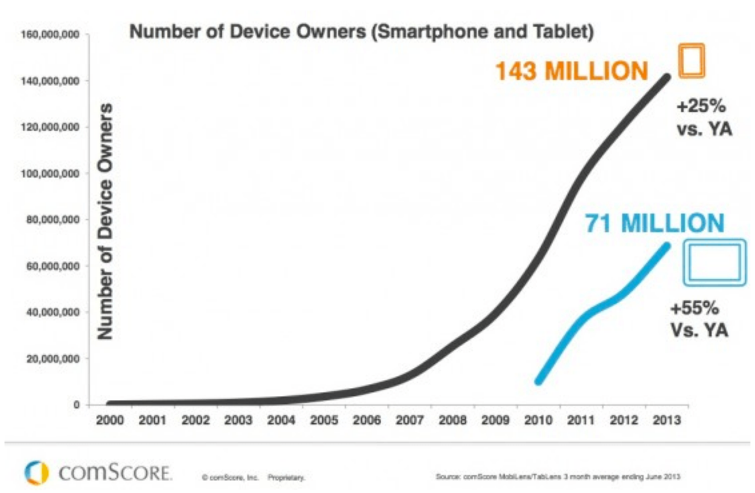


Figure 1.1: The image shows how the smartphone and tablet ownership in the US has rapidly increased over the years. Source: `https://heidicohen.com/67-mobile-facts-from-2013-research-charts`

The expansion of the user base and range of applications used, poses the third challenge. The number of people actively using wireless communication systems have drastically increased over the years. A wide range of devices, starting from radio and television to mobile radio to mobile telephone, satellite communication systems, global positioning systems make use of the electromagnetic spectrum which is licensed to specific telecommunication companies or government divisions.

With the ever-growing number of users and applications as seen in Fig. 1.1, the service provider is required to furnish the resource demanded by each other. Ideally, purchasing more spectrum would solve the problem but it would turn out to be exhaustive since the spectrum is not available in abundance. Therefore, scalability of the core network, given the spectrum based resource constraints, is a necessary feature.

Reducing the impact of mobility of the user device in the system is the fourth challenge. The random movement of the user device in the cellular environment causes variations in the strengths of the signal that it receives. This could affect the user experience to a great extent. To combat the same, some mobility models have been proposed. These models attempt to characterise parameters like cell residence time of new and handover calls and have been designed in such a way that it is applicable to most cellular environments.

The service provider also faces a number of challenges. As seen in Fig. 1.2, there is a steep increase in the use of internet leading to possible network congestion. The service

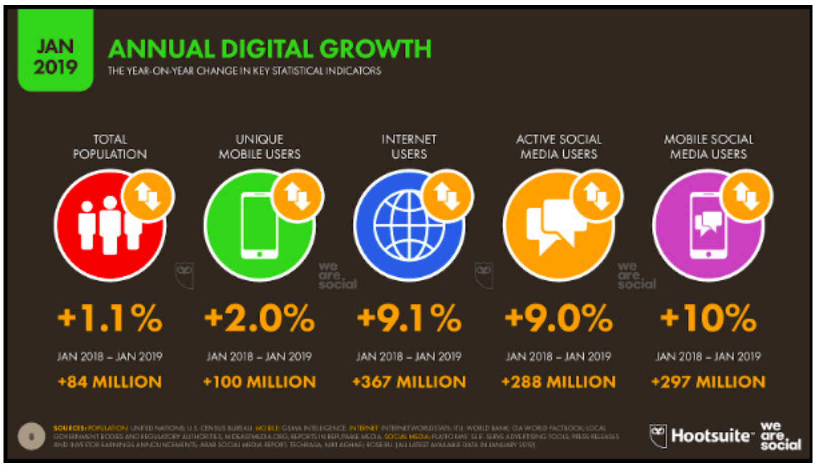


Figure 1.2: The image shows how the internet growth has accelerated in the year 2018. Source: `https://wearesocial.com/blog/2019/01/digital-2019-global-internet-use-accelerates`

providers are solely responsible for alleviating the network congestion. In addition, network security needs to be guaranteed and service providers can ensure the same by providing some services like mobile phone protection through implementation of standard encryption algorithms and by deploying stringent data privacy policies.

To sum up in brief, the overall performance of the network depends on how well the services have been offered to all its users in addition to being able to cater to the diversity of the data traffic conditions, growing number of users, channel conditions and the applications used and how well the design is able to utilise the entire spectrum. The study on block-proportional fair scheduling will further focus on how system scaling can be used to simplify the computational costs.

1.2 Carrier Assignment to Users

Many agencies are responsible for providing portions of the spectrum to their respective subscribers. It is to be noted that allocation and assignment are distinct. The allocation of frequencies is done depending on the type of service requested for, while assignment of frequencies to a user is to determine who is permitted to use what range of frequencies.

A vast number of licenses regarding the operation of wireless devices in such systems are assigned. Allocation of frequencies have also been done adhering to historic rules. On the other hand, the trend in frequency assignment in recent years has seen a radical shift from assignment by comparative hearing, to auctions, to influence of market dynamics directly affecting assignment strategies.

At this point, we will divert our attention to carrier assignment for multiple users in a multi-carrier system. This assignment should be done in such a way that every user is served **fairly** or **proportionally**. To model this particular routine in a proportional fashion, we need to have a quantitative notion of resources that every user has received on an average during every time step, and further assign carriers to the users depending on these numbers. To ensure proportionality among all users, the users with lower average rates until the previous time slot as compared to the other users in the network, will be given the necessary resources. Specifically, users with lower average rates will be **assigned carriers** with priority, thereby, causing those users with higher average rates to wait till the former are served. This ensures fairness in the scheduling.

Users can be classified into premium and non-premium class depending on how much they are able to pay for procuring services. Let us look at a simple example to understand the same. A non-premium user pays an amount ’X’ to procure services that are valid for a month. A premium user pays twice the amount to procure the same services for the same tenure. It seems only fair that the premium user gets the resources, specifically twice as much as what the non-premium user would get during this tenure. The premium users must receive the services in proportion to what they are paying for.

All these build up to the core problem of interest that is addressed through this thesis. The thesis presents both algorithmic and analytical insights on proportional fair (PF) scheduling. The proportional fair scheduling algorithm is widely used in multi-carrier systems such as those which employ orthogonal frequency division multiple access (OFDMA). The scheduler assigns carrier(s) to all users with the objective of maximizing the sum of logarithmic transmission rate Kim *et al.* (2004). The details of the algorithm are presented in Chapter 2.

1.3 Organisation of the report

The objective of this project is to implement and verify the working of a conventional PF scheme. To achieve the same, the following schemes are implemented on software and are simulated to obtain the results. The details of the same are presented in this report. The report is organised as follows:

Chapter 2 elaborates on the related works and concepts that are closely tied to the PF problem. The formal problem definition and contribution of the thesis are also included in this chapter.

Chapter 3 describes the details of the building blocks that are necessary to realise the proposed scalable PF scheme.

Chapter 4 presents the subtle details of the implementation of the proposed PF scheme and its variants.

Chapter 5 presents the experimental setup and results. It further summarizes the inferences arrived at, based on the same.

Chapter 6 contains the conclusion of the thesis and lists some open issues and arenas for future research and development.

CHAPTER 2

RELATED WORKS

2.1 Software Defined Networks

The majority of the network functionalities as on date, accept a stream of digital input data, process the same and deliver the output. This is exactly the way in which any digital program works. Many of the network functionalities are achieved by executing software on commodity hardware. Thus, modern day networking is more defined by software than hardware leading to the term Software Defined Networking. The main functionality implemented by these software driven systems is resource/carrier scheduling. With the ever-growing number of users and the population density, wireless networks need to scale on a continuous basis. This requires efficient scheduling algorithms for carrier assignment. The efficiency is measured not only in terms of balanced carrier association but also in terms of scalability with increasing load. With the advent of multi-core network processors, the scalability issue is comprehensively addressed. The challenge now remains in developing scheduling algorithms that shall effectively use the concurrency made available through the multi-core environment. Specifically, the algorithms must have multiple modules that could execute concurrently in the multiple cores provided with minimal data/control dependency between them. This thesis presents one such scheduling algorithm which has the potential to exploit the concurrency made available through multi-cores for scaling.

2.2 Multi-Carrier Network Resource Scheduling

In a practical wireless communication setup, the wireless channel's strength varies due to fading. If the base station (BS) and the users are separated by a large distance, we can assume that the multiple paths fade independently, meaning that the channel conditions for all the users in the system are not symmetric. Some users may have strong channels while the others may have relatively weaker channels. All the users may not be served uniformly. This arises the need for PF scheduling routine that ensures that all the wireless users are served fairly, that is, the services are being offered in proportion to the amount of money that the user invests Tse and Viswanath (2005).

The goal of PF scheduling is two-fold. On one hand, the users must be fairly served and at the same time, the routine must ensure that the system's overall throughput is high. One way of ensuring high system throughput is to schedule the users depending upon the strength of the channel conditions. That is, the routine could assign time slots to users that have good channel strength. By doing this, the overall throughput of the system can be maximised. This scheduling routine is called opportunistic scheduling.

Opportunistic scheduling should guarantee fairness among users Liu *et al.* (2001). At times, the rate at which a signal is transmitted is widely affected due to interference and other noises. Cell partitioning and frequency planning strategies are adopted in order to combat interference. The wireless users are widely dispersed within a cell and could be at disparate distances from the base station. In this case, users that are close to the BS are allowed to transmit since the channel conditions are better. On the other hand, those users that are far away from the BS have relatively weaker channel conditions and do not transmit and thereby, do not contribute to the system throughput. Here, the **level of satisfaction** among users is not uniform and the scheduler must ensure this. Better frequency planning methods can be employed in order to use the spectrum efficiently. While using realtime applications like call services, the user has a set of service constraints or requirements like low-latency, higher data rate and many more. If the channel is in a deep fade, the user barely gets any data service. If the channel fades slowly, the instantaneous rate of the user is relatively less and the user waits for a long time to transmit. The scheduling routine must ensure that user requirements are satisfied.

The PF scheduler is a classic case of the opportunistic scheduler and has been implemented in Qualcomm's HDR system (also known as 1xEV-DO) Huang *et al.* (2003) Andrews and Zhang (2011). This is a single carrier system that uses a common shared downlink to schedule user transmissions. Every time slot has a fixed duration of 1.67 ms. At every time slot, exactly one user is allowed to access the link. The scheduler needs to make a decision about which user to schedule for that time slot. This decision is based on the instantaneous data rate and the historic average user rate of the wireless users. If the user with the highest data rate were to be chosen at each time step, the other users that have lower rates will be starved. Qualcomm adopts a fair-sharing strategy where the current data rate for each user at that time step is compared with the users' historic average data rate and that user with the highest ratio is chosen. This algorithm is closely related to the fairness criterion proposed in Kelly *et al.* (1998), that is described below.

Definition Kelly *et al.* (1998): *An allocation of rates $\overrightarrow{x}$ is proportionally fair if and only if, for any other feasible allocation $\overrightarrow{y}$, we have,*

$$\sum_{s=1}^{S} \frac{y_s - x_s}{y_s} \leq 0 \tag{2.1}$$

The above equation is indicative of the fact that the change in the allocation must have a **non-positive average change**. A unique solution to the proportional fair assignment problem has been proposed. The proof of the understated theorem can be found in the Appendix of the Kim *et al.* (2004) paper.

Theorem Kelly *et al.* (1998): *There exists one unique proportionally fair allocation. It is obtained by maximizing $J(\overrightarrow{x}) = \sum \ln(x_s)$ over the set of feasible allocations.*

Multipath fading causes signal strength variations and that affects the throughput of the system to a great extent. A broadband transmission scheme called Orthogonal Frequency Division Multiplexing (OFDM) has been found to be extremely useful in reducing the effects of multipath fading. There are multiple carriers in these multiuser OFDM systems. Many scheduling schemes have been studied but the PF scheduling scheme appears as the best bandwidth management scheme. The allocation of resources to a user is done when the user's current channel conditions are the most favourable with respect to its historic time average Wang Anchun *et al.* (2003).

2.3 Symbol-to-Interference ratio - The Significance

As discussed in the previous section, fading causes time variations in the signal strengths, thereby leading to asymmetric channel conditions for users in the wireless network. At a given time, different users experience different channel strengths. An opportunistic scheduling scheme allows users with good channel strength to transmit with the key goal of maximising the system throughput and effectively utilising the shared channel resources.

The opportunistic scheduling at the Link layer of the base station is done based on the knowledge about the users' channel strength. The channel state information (CSI) should be available at the BS. The nature of the wireless channel, be it slow or fast fading, is random. At every time step, the channel quality is bound to change, but not in a deterministic way. These changes need to be accounted for by the BS while making scheduling decisions. That is, there should be a mechanism that allows the user to send a channel quality feedback periodically to the BS.

The steady-state achievable data rate for each user is made available at the base station at all times. In addition to that, each user in the network tracks its channel signal-to-interefernce ratio (SIR), which determines the instantaneous user data rate and this piece of information is passed on to the BS. High data-rate (HDR) 3.5G systems already have these features embedded in their design. Most of these systems employ channel estimation techniques at the receiver. In such cases, the CSI available to the BS is not exactly accurate because this information could be delayed at times and could have been tampered with because of channel estimation error. If one were to assume that the CSI is perfectly known to the BS, it means that, at every time step, the scheduler knows the instantanous user data rates of all the users in the network. In case the CSI is known only partially or not known to the BS, the steady state data rate numbers are used.

2.4 Definition of the Problem and Relevance

The two biggest challenges in the next generation multi-user multi-carrier wireless communication systems are efficient resource utilisation among wireless users and maximising the system throughput. A wide range of services and utilities are provided to the users with an adequate Quality of Service (QoS). On one hand, the wireless resources need to be fairly distributed among the users while on the other hand, the system throughput needs to be maximised. The PF scheduling scheme incorporates a certain level of fairness while allocating resources to the users. Given that the size of the network is increasing rapidly, the existing PF scheduling routines for multi-carrier systems are not practically feasible due to their high computational costs. Thus, the challenge is to *d*evelop a more sophisticated and computationally less intensive scheduling routine for a multi-user multi-carrier wireless communication setup, that ensures a uniform experience among the users and maximal system throughput.

2.5 Contribution of the thesis

As stated in the previous section, the objective is to develop a scheduling scheme that ensures a certain level of fairness and also maximises the system throughput. According to Kim *et al.* (2004), the cost of scheduling a set of U users in a C carrier environment setup is $(|U|^{|C|})$ and this needs to be reduced. The following are the contribution of the thesis.

1. Proposes a scalable technique for resource scheduling on a multi-user multi-carrier wireless communication system;
2. The proposed scheduling algorithm can be run on commodity multi-core processor hardware;
3. Efficient partitioning strategies to convert the larger problem into sub-problems of significantly smaller size that could be concurrently solved using optimal compute intensive sub-routines to yield an overall efficient throughput; and,
4. Empirical verification of the claims on a MATLAB simulation framework.

CHAPTER 3

BASIC BUILDING UNITS OF SCHEDULING FRAMEWORK

This chapter focuses on describing the roles and functionalities of each of the functional blocks that comprise the whole scheduling framework of the multi-user multi-carrier wireless communication system. The blocks are software simulated and have been parameterized to the maximum extent possible. The process of scheduling can be broadly divided into two steps - firstly, allocating the resource, which is the carrier in this case, to a fit user, and secondly, go through a scheduling routine where the Link layer makes a decision about when the user will receive the service. We will focus on these two steps. To carry out the same, we will construct the following blocks:

- **Resource Allocation Block**: This block is responsible for generating all possible permutations of how the resources can be allocated to a user.
- **Proportional Fair (PF) Scheduler**: This block is responsible for making the scheduling decisions using the **best** possible allocation and ensuring that the users are served fairly.
- **The Carrier-User (CU) Matrix**: This matrix contains the instantaneous rates of different carriers for different users at every time step.

These blocks work in a sequential fashion. The resource allocation block provides the necessary information about which carriers are allocated to which users. The PF Scheduler block uses the **best** of many possible allocations and inputs from the CU matrix to serve all the users uniformly. This simulation aims to validate the analytical equations that define the performance of proportional fair scheduling. Designing a scalable solution using the basic building blocks along with a few variants of greedy scheduling schemes will be explained in greater depth in Chapter 4. The rest of this chapter will aim to explain the functionalities of each of the building blocks of this multi-user multi-carrier communication system.

3.1 Best Resource Allocation

The real question is to find out the best possible allocation strategy that the scheduler needs to adapt to, for a given particular time slot. The question involves generating the set of all possible allocations and then choosing the best among them. The following example attempts to explain the same.

Consider a wireless communication environment that has a user set with three users, U1, U2 and U3 and a carrier set with two carriers, C1 and C2. We want to find out in how many different ways two carriers can serve three users. It is to be stressed at this point of time that, **no two users are allowed to use the same carrier in a given time step, but the converse, that is, the allocation of multiple carriers to the same user is allowed.** Keeping this in mind, let us generate all the possible allocation matrices for this system. The interpretation of the matrix should be done as follows: If the $(i, j)^{th}$ entry in the matrix is 1, it means that the carrier i has been allocated to user j, else it is 0.

It is to be noted that the allocation matrix will have exactly one entry with value 1 in every row. There are $|U|$ columns and $|C|$ rows in the matrix. By the laws of combinatorics, the number of such possible matrices that can be generated is $|U|^{|C|}$, where $|U|$ is the cardinality of the user set and $|C|$ is the cardinality of the carrier set. The set of all possible allocations has been shown in Fig. 3.1

The scheduler now needs to pick the best allocation from the set of all possible allocations and follow the scheduling routine. The picking of the best allocation strategy depends upon a lot of factors, most importantly, the instantaneous rates of all the users. The basis for this assertion will be explained in subsequent sections of this report. The various permutation patterns can be generated by a simple piece of software code which can be found in the Appendix of this report. If one observes closely, Fig. 3.1 depicts a pattern for generating these different permutations. The functional block exploits this property and returns the next pattern in that list, given a particular pattern.

	U1	U2	U3
C1	1	0	0
C2	1	0	0

	U1	U2	U3
C1	1	0	0
C2	0	1	0

	U1	U2	U3
C1	1	0	0
C2	0	0	1

	U1	U2	U3
C1	0	1	0
C2	1	0	0

	U1	U2	U3
C1	0	1	0
C2	0	1	0

	U1	U2	U3
C1	0	1	0
C2	0	0	1

	U1	U2	U3
C1	0	0	1
C2	1	0	0

	U1	U2	U3
C1	0	0	1
C2	0	1	0

	U1	U2	U3
C1	0	0	1
C2	0	0	1

Figure 3.1: Resource Allocation Policy

3.2 Dynamic Carrier-User Matrix

The Carrier-User (CU) matrix plays an essential role in selecting the best resource allocation for the users at every time step. There are $|U|$ columns and $|C|$ rows in the matrix. The $(i, j)^{th}$ entry in the matrix is the instantaneous rate of user j for carrier i. The entries of the matrix change at every time step.

It becomes imperative to model this block as close to a realistic scenario since the subsequent routines rely on this. The type of channel model we choose, the SIR values, the overall bandwidth and basic system parameters like sub-carrier spacing, type of modulation used, all of these dictate to a great extent, how these rates turn out to be. The smaller the SIR values, higher the diversity in the rates of the users. These various simulation parameters can be varied to fine-tune the system as per the need of the system. For the sake of experimentation, the values used in Kim *et al.* (2004) have been taken.

We now need to understand how the CU matrix is necessary for picking the best allocation. Let us consider an example of the wireless communication environment that has a user set with three users, U1, U2 and U3 and a carrier set with two carriers, C1 and C2, as shown in Fig. 3.1. We have nine possible permutations of allocation patterns.

Let us consider the first block. Here, the carriers C1 and C2 are allocated to user U1. The users U2 and U3 are not served in this round. The CU matrix gives us the instantaneous rates of users in the user set. Given the CU matrix and the chosen allocation pattern, we can deduce that the user U1 can use both the carriers C1 and C2, that is, the instantaneous rates that are associated with the carriers. This means we add up the entries in the first column of the CU matrix and say that U1 has been allocated an aggregate rate equal to the sum of the entries. The users U2 and U3 are not served and hence, their aggregate rates are equal to zero. In the same way, we can compute the aggregate rates of the users for different allocation patterns. The computation of the aggregate rates of the users is formally a dot product operation between the CU matrix and the allocation matrix. The aggregate rates are obtained by adding up the entries of the resulting matrix along the columns.

3.3 PF scheduler

The scheduler uses the inputs of the CU matrix and the set of allocation matrices and assigns the carriers to the users based on the best allocation. To find the best allocation, we review the following theorem stated in Kim *et al.* (2004):

Theorem (PF scheduling for multi-carrier systems)Kim *et al.* (2004): *A scheduler is proportionally fair in a multi-carrier system if and only if it satisfies*

$$P = \arg\max_{S} \prod_{i \in U_s} \left(1 + \frac{\sum_{k \in C_i} r_{i,k}}{(T-1)R_i^{'}}\right) \tag{3.1}$$

where U_S is the set of selected users by S, C_i is the set of carriers allocated to user $i(\in U_S)$ and $r_{i,k}$ is the instantaneous transmittable rate of the carrier $k \in C_i$ when transmitted for user i at the current slot. R_i is the average rate of user i at the previous slot, and T is the average window size.

Here, we assume that the all the users are selected by the scheduler S. The average rates of these uses are updated from $R_i^{'}$ as cited in Kim *et al.* (2004),

$$R_i^{(S)} = \frac{(T-1)R_i^{'} + I_{i \in U_s} \sum_{k \in C_i} r_{i,k}}{T} \tag{3.2}$$

where $I_{(.)}$ is 1 if $(.)$ is true and 0 if false.

The product term P shown in Equation (3.1) is computed for all possible allocation and given CU matrix. The window size T is a running index that changes with time. As stated in Equation (3.1), we need to maximise this product and take the argument, which is the allocation, and declare it as the best allocation of all possible allocations for that time step.

Algorithm 1 Optimal PF Scheduler (OFPS)

INPUT: User set U, Carrier set C, Time step T, Resource allocation matrix A, Historic user data rate R
OUTPUT: Total user data rate allocation made to all users

```
Function $PFSchedule()$:
for each time step T do
    for all allocations A at time step T do,
        for every user in user set U  do
            Compute product term P (as per Equation (3.1)) using allocation A.
        end for
    end for
    Find the largest of the product terms computed above.
    Find the allocation matrix that gives the largest product and declare it as best
    allocation maxCU.
end for
Compute total data rate allocated to user using allocation maxCU.
Update historic data rate of every user using expression in Equation (3.2).
```

After finding the best allocation, the scheduling routine is completed by assigning the carriers to the users. The average rates of the users need to be updated at every time step. The aggregate rates of the users are computed using the best allocation and the average rates of the users are updated as per Equation (3.2). The same routine is repeated for multiple time slots where, in each time slot, the instantaneous rate keeps changing, resulting in a change in the best allocation pattern. At every time slot, the aggregate user rate is updated. At the end of the run, the average of the aggregate user rate for every user is taken and then plotted against user index. This gives a visual idea of the long-term throughput of the system and also verifies if the users are served fairly or not.

3.4 Special Cases of PF scheduler

A multi-user single carrier wireless communication system can be modelled by setting the cardinality of the carrier set to the value 1. The problem now narrows down to a **single carrier proportional fair scheduling** routine. The Equation (3.3) becomes,

$$P = \arg\max_{S} \frac{r_{i(\in U_s),1}}{R'_{i(\in U_s)}} \tag{3.3}$$

The same routine as described for multi-carrier systems is used for this. Interestingly, we observe that the round robin (RR) scheduler is also a special case of the PF scheduling problem. By far, round robin is the fairest of all schedulers. Every user gets an equal chance. This is as good as saying that the average rates of all the users are identical. When this is the case, the aggregate rates of all the users are identical and we get the long-term throughput as a flat uniform curve. The RR scheduler will serve as a base case for comparison. The routine discussed in this chapter is an implementation of the method proposed in Kim *et al.* (2004). It has a time complexity of $|U|^{|C|}$. Given a small set of users and carriers, say, 80 users and 16 carriers, the complexity of the algorithm would be 80^{16} at **every** time slot, which is prohibitively high. Thus, the routine as stated in this chapter is NOT practically scalable.

In the next chapter, a scalable PF scheduler has been described in detail. The scheduler uses the routine presented in this chapter as a basic building block and proposes a scalable solution using the same.

CHAPTER 4

SCALABLE BLOCK PROPORTIONAL FAIR SCHEDULING

From the previous chapter, we know that the PF scheduler assigns the carriers to the users based on the best allocation policy. As the number of users and/or carriers increases, the above best allocation strategy shall become computationally costly. This hinders the performance of the system that ideally, should be very adaptable, thereby arising the need for a **scalable** solution. A scalable solution in contrast to the original solution, delivers the same system throughput but with much lesser computational cost. In this thesis, a **block** proportional fair scheduling scheme has been proposed that comprehensively addresses the problems of scalability and fairness of the solution. The scheme has been explained to a great extent in the course of this chapter along with the implemention details for the functionalities of the fundamental building blocks in the scheduling framework of the multi-user multi-carrier wireless communication system.

The solution presented in the previous chapter had a computational complexity of $(|U|^{|C|})$. Thus, the routine can work for small values of U and C and will take prohibitively large amounts of time for reasonably large values of U and C. Hence, the solution presented in the previous chapter can not be used for large values of U and C. The basic intuition behind the proposed solution is to partition the given problem into blocks such that the optimal algorithm presented in the previous chapter can be used on each one of those blocks. Each block is assigned a set of carriers and users. The novelty of the proposed solution is to partition both the user and carrier sets across the blocks so as to ensure fairness in the overall scheduling.

Now the following question arises - what is a block and how is it related to the original set of $|U|$ users and $|C|$ carriers. The task of assigning $|C|$ carriers to $|U|$ users is broken down. What this means is that, instead of one PF scheduler unit performing this functionality, we shall introduce multiple such PF scheduler units that will each perform the functionality over a set of $|U_{block}|$ users and $|C_{block}|$ carriers **concurrently**, where $|U_{block}|$ and $|C_{block}|$ are the number of users and carriers in every block respectively. The

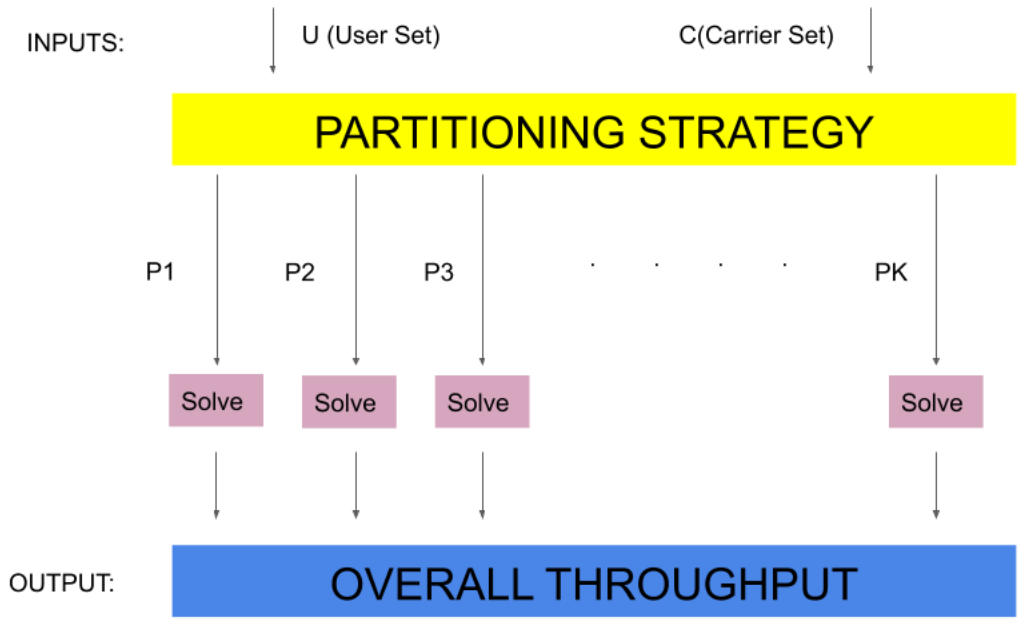


Figure 4.1: Schematic of the block PF scheduling framework

schematic for this routine can be found in Fig. 4.1. In this figure, the user and carrier sets U and C are partitioned into partitions P_1, P_2 and so on till P_K. Each of these partitions contain $|U_{block}|$ users and $|C_{block}|$ carriers and they are solved concurrently using Algorithm OPFS given in Chapter 3. The throughput obtained from each block contributes to the overall long-term throughput of the system.

For the sake of easier implementation, we have assumed that the values $|U_{block}|$ and $|C_{block}|$ are scalar multiples of $|U|$ and $|C|$ respectively. The implementation details throw some light on the numerical values that have been taken for simulation purposes, but for now, we have num_block number of PF scheduler blocks working concurrently. The computational complexity of this scheme reduces to $(|U_{block}|^{|C_{block}|})$. This is much less compared to the original PF scheme since the complexity has come down by a great factor.

4.1 The Partitioning Problem

The previous chapter discussed in detail about the functional blocks in the scheduling framework. As far as the design of Resource Allocation block and CU matrix formulation are concerned, the proposed scheme and the original PF scheduling scheme have no major changes. As described earlier, the proposed block PF scheduling scheme will use multiple PF scheduler units, all working concurrently.

Each PF scheduler unit will assign $|C_{block}|$ carriers to $|U_{block}|$ users based on the best allocation policy. The question is, how do we partition $|U|$ users and $|C|$ carriers into num_block clusters of $|U_{block}|$ users and $|C_{block}|$ carriers respectively. While dividing the carriers and users into blocks, we must keep in mind that there must be a *load balance* across all scheduling units. To understand the above statement, let us consider the following case. Let the number of carriers per block be 4. Let the carriers C1, C2, C3, C4 be such that they have the highest data rates among all carriers and C5, C6, C7, C8 be such that they have the lowest data rates among all carriers. Relatively, the throughput of the former block will be higher than that of the latter. The notion of "fairness" in serving the users is violated here.

To ensure that the above does not happen, we need to distribute the carriers into num_block sets of $|C_{block}|$ carriers to ensure that the carrier rates in each set is balanced. We can do the above using a simple load balancing technique. Consider the case demonstrated earlier. Assume that there are 8 carriers and number of carriers per block is 4. Firstly, the carriers are to be sorted in the increasing order of their respective data rates. Let the sequence after sorting be C2, C4, C5, C6, C1, C3, C7, C8. We now cluster these carriers into 2 blocks of (C2, C8, C4, C7) and (C5, C6, C1, C3). The overall available data rate is balanced across both these clusters. As could be inferred easily from the above example, a simple greedy technique is employed such that largest and the smallest, the second largest and the second smallest and so on are in the same cluster. Once the clusters for the carriers are made, the clusters for the users need to be made as well.

A similar load balancing technique needs to be adapted in order to divide the users into multiple clusters. It is to be noted that a random partitioning of the original user set into num_block sets of $|U_{block}|$ users is not entirely wrong. A more sophisticated measure that will help in balancing the load across all blocks is the average rates of the users. The numbers give some intuition about the amount of resources (rates, in this case) that have been assigned to every user. Thus, those users who have not been served *w*ell until that time step will be given a priority over the others in order to ensure that all users are served in a fair fashion.

Consider the following case: There are 12 users and number of users per block is 6. Firstly, the users are to be sorted in the increasing order of the inverse of their average data rates, as of that time step. Let the sequence after sorting be U11, U2, U4, U10, U5, U6, U9, U1, U3, U8, U7, U12. We now cluster these carriers into 2 blocks of (U11, U2, U4, U8, U7, U12) and (U10, U5, U6, U9, U1, U3). As could be inferred easily from the above example, a simple greedy technique is employed such that largest and the smallest, the second largest and the second smallest and so on are in the same cluster. The overall load is balanced across both these clusters.

It is to be noted that this partitioning of users and carriers takes place at every time step depending on the data rates of all the carriers. Once the partitioning is done on the user and carrier set, a sub-matrix of the CU matrix is generated. This sub-matrix will contain the data corresponding to the users and carriers that belong to the block of interest. Based on this matrix, the PF Scheduler does the scheduling routine and assigns carriers to users based on the best allocation policy and returns the aggregate rates of the users.

Once the allocations have been made for all the blocks, the average rates of the users are updated and at the next time step, the same routine is repeated. It is to be noted that once the partitioning is done, multiple PF Scheduler units can work concurrently. Given that the networking equipments that perform these scheduling operations are multi-core hardware, the proposed algorithm can be directly mapped onto these hardware such that it exploits the available concurrency. However, the MATLAB simulations performed as a part of the experimental analysis are carried out in a sequential fashion due to MATLAB compute constraints. In this simulation, post partitioning, the scheduling is done on the blocks one after the other, thereby mimicing the concurrency.

4.2 Variants of the Partitioning Problem

It is also interesting to note that the proposed proportional fair scheduling scheme can be modified to exhibit multiple schemes by varying the partitioning strategy. Two such variants of the partitioning problem are listed in the following section.

4.2.1 Greedy User and Balanced Carrier Partition

For this variant of the proposed scheme, the carriers are partitioned in a balanced fashion while the user partitioning is done in a greedy fashion. A greedy user is one who aims to get the best possible data rate. The carrier partition is done as cited in the previous section wherein we sort the carriers in the increasing order of their respective rates and cluster them by ensuring a load balance.

A greedy user partitioning can be done as follows: Consider the system with 8 carriers and 12 users. Let there be 4 carriers and 6 users per block. Assume that the carriers have been sorted as C2, C4, C5, C6, C1, C3, C7, C8. These carriers are partitioned into two blocks (C2, C8, C4, C7) and (C5, C6, C1, C3). Consider the first cluster (C2, C8, C4, C7). We take carrier C2 and find out the user for which the data rate is the highest, say for user U3. Similarly, for all carriers in a cluster, we find out the users for whom the data date is highest on the given set of carriers. The partitioning is done until the entire user set is exhausted. This kind of partition on the user set can lead to the following cases.

The same user might have highest data rate on two or more distinct carriers that belong to different blocks. Consider the cluster (C2, C8, C4, C7) and (C5, C6, C1, C3) in the example cited above. Suppose data rate is highest for user U6 on carriers C2 and C3. As far as the simulation is concerned, the routine of partitioning is done in a sequential fashion. Therefore, depending on which cluster is picked first, U6 is allotted to that particular block. But when the same is done in a parallel fashion, U6 is equally likely to land up in either block.

Some users may not have the highest data rate on any carrier. Let us say the data rate is highest for user U6 on carriers C2 and C3. Also assume that U6 has been assigned to the block that contains carrier C2. Now, U6 can not be allotted to the block containing carrier C3. Ideally, what we can do is, we can find out the user who has the second highest data rate on carrier C3 and assign that user to the block. If the following routine is carried out till all the users in the user set are exhausted, then some blocks will have users that have considerably low data rates on the carriers in that block. It is true that a 'greedy' way of partitioning users can land up in such a state. But the goal is to also ensure consistency in the throughput of each of these scheduling units. So, instead of finding out the user who has the second highest data rate on carrier C3, we can randomly pick a user that has not been assigned to any block and assign that user to the block containing carrier C3. This can ensure a balance among the various blocks to some level.

Multiple users may have the highest data rate on a single carrier. Let us consider the cluster (C2, C8, C4, C7) and let the users that are eligible to belong to the block be ((U4,U9,U11),(U1,U3),U6,U2). Here, users U4, U9 and U11 have the highest data rate on carrier C2; U1 and U3 have the highest rate on carrier C8 and so on. The block must have only six users. So, in this case, any six users of the total lot is chosen at random and assigned to the block. The results for the simulation of this scheme can be found in the next chapter. The software implementation code can be found in the Appendix.

4.2.2 Random User and Greedy Carrier Partition

For this variant of the proposed scheme, the carriers are partitioned in a greedy fashion while the user partitioning is done in a random fashion. A Greedy carrier aims to serve the user which has the highest data rate on that carrier. The carrier partition is done as cited in the previous section where we sort the carriers in the increasing order of their respective rates and cluster them by ensuring a load balance. The difference between the previous scheme and this scheme is in the way the users are partitioned into blocks. We find the user which has the highest data rate on that carrier and assign that user to that block.

A random user partitioning can be done as follows. Consider the system with 8 carriers and 12 users. Let there be 4 carriers and 6 users per block. Assuming we have sorted the carriers as C2, C4, C5, C6, C1, C3, C7, C8, we now cluster these carriers into 2 blocks of (C2, C8, C4, C7) and (C5, C6, C1, C3). The user set U1, U2,...U12 is randomly permuted. From this randomly permuted list, the users are picked sequentially. Let user U3 be picked first and let that user have the highest rate on carrier C2. Now the task is to assign user U3 to the block that contains carrier C2. The partitioning is done until the entire of the user set is exhausted. This kind of partition on the user set can lead to the following cases:

The selected user might have highest data rate on two or more distinct carriers

that belong to different blocks. Consider the cluster (C2, C8, C4, C7) and (C5, C6, C1, C3) in the example cited above. Suppose user U6 is picked and the data rate is highest on two carriers C2 and C3. As far as the simulation is concerned, the routine of partitioning is done in a sequential fashion. If carriers C2 and C3 belong to the same block, there is no issue. Unlikely, here, in our example C2 and C3 belong to different blocks. The routine of finding the index of the largest element in an array in MATLAB is such that, if there are duplicates of the largest element of the array, it returns the *s*mallest index of the largest element in that array. So, here, user U6 is assigned to the block containing carrier C2. Note, it can also belong to the block containing carrier C3, except a different routine must be incorporated.

Monitoring the number of non-full blocks is essential. Let us assume that users (U4,U9,U11,U1,U3,U6) are assigned to the block containing carriers (C2, C8, C4, C7). Say user U2 is picked. Let this user have the highest data rate on carrier C2. The number of users that can be assigned per block is 6, as stated. The block already has 6 users (U4,U9,U11,U1,U3,U6), which means, that the block is full and that no more users can be assigned to this block. When such a situation arises, the user U2 should be assigned to a different block. This case suggests that there must be a mechanism that monitors and keeps track of all the blocks that are non-full. If a user has the highest data rate on some carrier that belongs to a non-full block, then the user is assigned to that block, else it is randomly assigned to another non-full block and this routine is repeated till the entire of the user set is exhausted.

To sum up, this chapter elaborated on the different procedures for partitioning the carrier and user set among blocks. The ideas presented in this chapter are implemented using MATLAB and the simulation results are presented in the next chapter.

CHAPTER 5

INFERENCES AND EXPERIMENTAL RESULTS

The previous chapter primarily elaborated on the core idea of the block proportional fair scheduling scheme. This chapter focuses on the details of the software implementation (on MATLAB) of the same, along with simulation results. The basic experimental framework of the multi-user multi-carrier wireless communication system is described in detail in this chapter. We assume that the channel is bandlimited to W Hz. It has been modelled statistically to follow a Rayleigh fading pattern Tse and Viswanath (2005). The highest and most reliable rate of transmission on such a channel is given by $\frac{W}{C}log_2(1+|h|^2SIR)$, where $|h|$ represents the instantaneous channel condition and C, the total number of carriers Tse and Viswanath (2005).

5.1 The Basic Framework

This section presents the basic framework of the multi-user multi-carrier wireless communication system. The entire simulation is parameterised to explore multiple "what-if" scenarios. The following are the simulation parameters that can be varied to explore different scenarios.

1. `N`: Number of time steps. This value is set to 100 for simulation purposes.
2. num_block: Number of blocks in the scheduling framework. This parameter is varied.
3. U_block: Number of users per block in the scheduling framework. This parameter is varied.
4. C_block: Number of carriers per block in the scheduling framework. This parameter is varied.
5. SIR_dB: Average SIR in dB scale. This value is set to 10 for simulation purposes.
6. `TS`: Duration of one time slot (secs). This value is set to 0.001 for simulation purposes.

7. `SD`: Duraton of one modulated symbol (secs). This value is set to 0.0001 for simulation purposes.

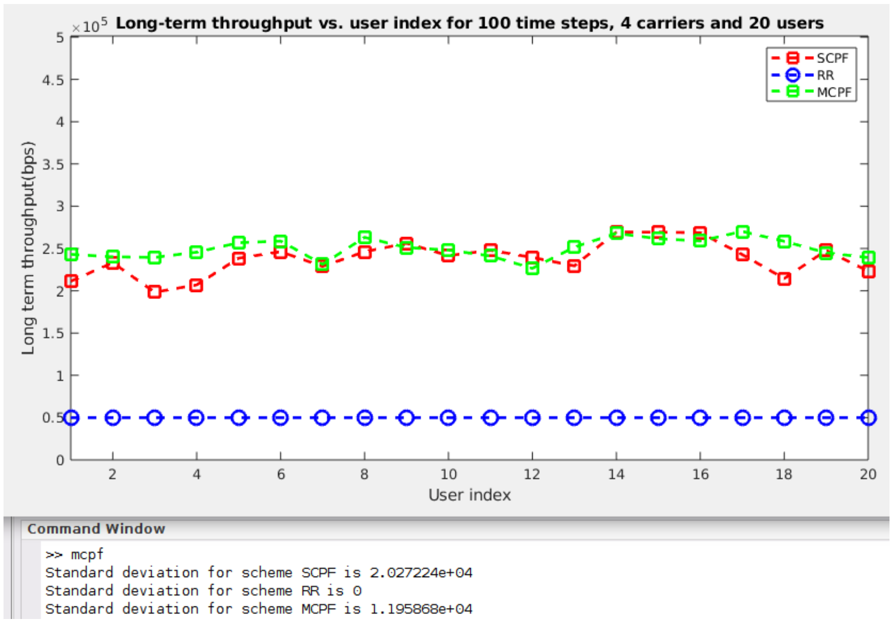
8. `BPS`: Number of bits per symbol (BPSK - 1, QPSK - 2, 16QAM - 4, 64QAM - 8). This value is set to 2 for simulation purposes.

9. `BW`: Bandwidth of the channel (Hz). This value is set to 1 MHz for simulation purposes.

The long-term throughput was obtained by simulating the scheduling routines for different values of the parameters stated above. For comparing different strategies, the Round Robin scheduler is used as a base case.

5.2 The Conventional PF scheduler approach

Empirical analysis of the conventional PF scheduler was taken and the results are presented in this section. Fig. 5.1 and Fig. 5.2 represent the long-term throughput for a multi-user multi-carrier wireless communication system with a scheduling routine that follows the conventional PF scheduler approach.

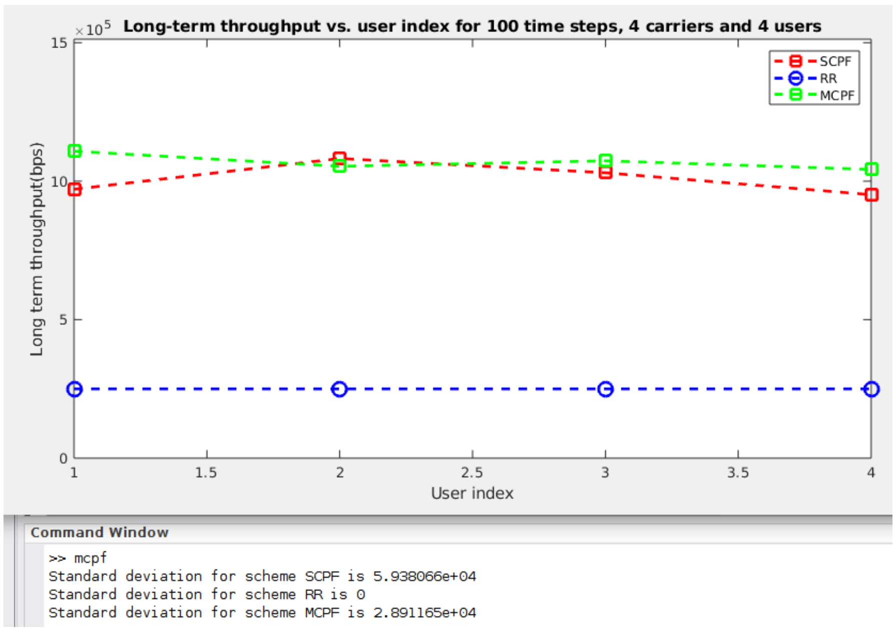


```
>> mcpf
Standard deviation for scheme SCPF is 2.027224e+04
Standard deviation for scheme RR is 0
Standard deviation for scheme MCPF is 1.195868e+04
```

Figure 5.1: Long-term throughput vs. user index for conventional PF scheduling

In Fig. 5.1, three different schedulers have been demonstrated, namely, a single-carrier PF (SCPF), Round robin (RR) and multi-carrier PF scheduler (MCPF). The SCPF is a special case of the MCPF scheme where the number of carriers is set to 1. When the instantaneous rates of all users across all carriers are identical, we arrive at the

RR scheme, which is by far, the fairest of the three. Fig. 5.1 represents the long-term throughput for a 20 user-4 carrier system and Fig. 5.2 represents the long-term throughput for a 4 user-4 carrier system. The idea of fairness can be closely associated with the deviation in the final average rates of the users. The outputs displayed under the curve in the figures are the values of the standard deviation in the total allocations that have been made to all the users in 100 time steps. It is worthwhile to note that the RR scheme gives constant throughput while the SCPF scheme and the MCPF scheme behave almost identically. As the number of users increases, the scheduler seems to be more fair since the variations in the user rates reduces. One can also observe that the deviation seen in the SCPF scheme is always greater than that of the MCPF scheme.

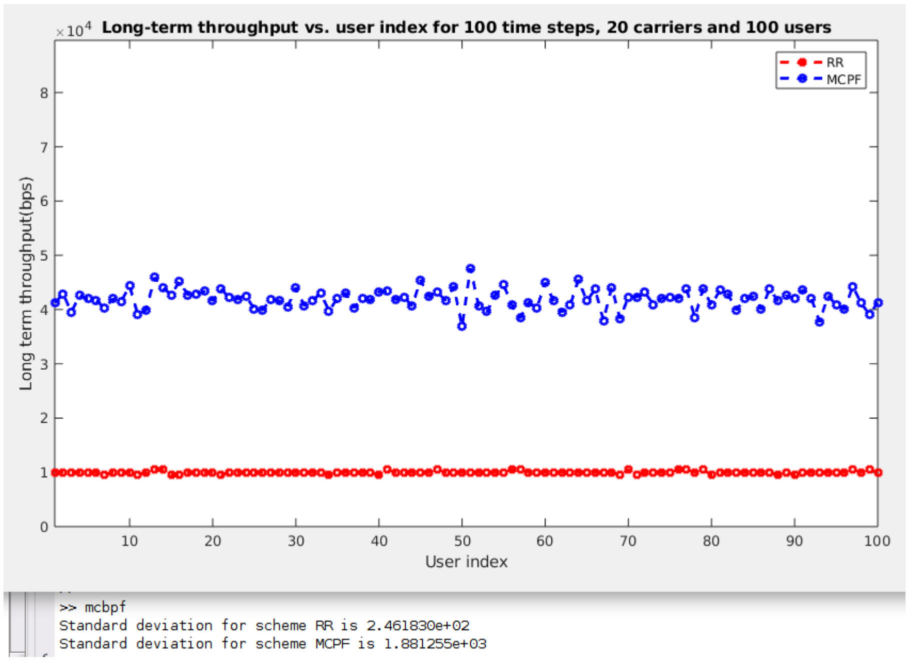


```
>> mcpf
Standard deviation for scheme SCPF is 5.938066e+04
Standard deviation for scheme RR is 0
Standard deviation for scheme MCPF is 2.891165e+04
```

Figure 5.2: Long-term throughput vs. user index for conventional PF scheduling

5.3 The Proposed Block PF scheduler approach

Fig. 5.3 and Fig. 5.4 represent the long-term throughput for a multi-user multi-carrier wireless communication system with a scheduling routine that follows the proposed block PF scheduler approach.

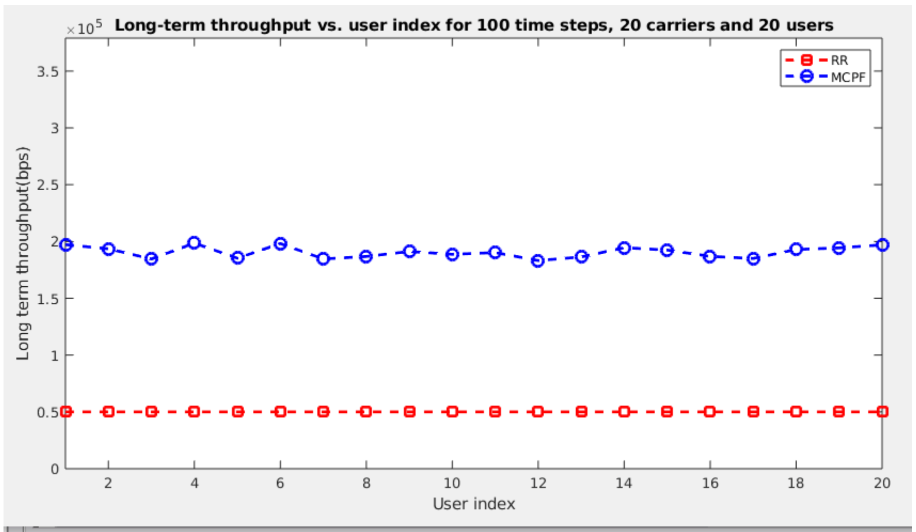


```
>> mcbpf
Standard deviation for scheme RR is 2.461830e+02
Standard deviation for scheme MCPF is 1.881255e+03
```

Figure 5.3: Long-term throughput vs. user index for block PF scheduling 20 carrier-100 user-5 block system

The proposed block proportional fair scheduling scheme has been implemented in a sequential fashion. Prior to the scheduling routine, the carrier and user sets are partitioned and placed in suitable blocks. In addition to MCPF scheme, the RR scheme has also been explored here. Fig. 5.3 represents the long-term throughput for a 20 carrier-100 user system and Fig. 5.4 represents the long-term throughput for a 20 user-20 carrier system. Here,the number of blocks is equal to 5. Fig. 5.5 represents the long-term throughput for a 20 carrier-100 user system that has **10 blocks**. Given a finite number of carriers in the system, a fair experience is guaranteed as long as there are not too many users. For the 20 user-20 carrier case, the RR scheme works perfectly as expected, but as the number of users increases, there is minimal deviation seen in the final average user rates. This is due to many users requesting for a resource that is not abundantly available. Thus, increasing the quantity of the deliverable resources for an ever-increasing user base seems to be a plausible solution here.

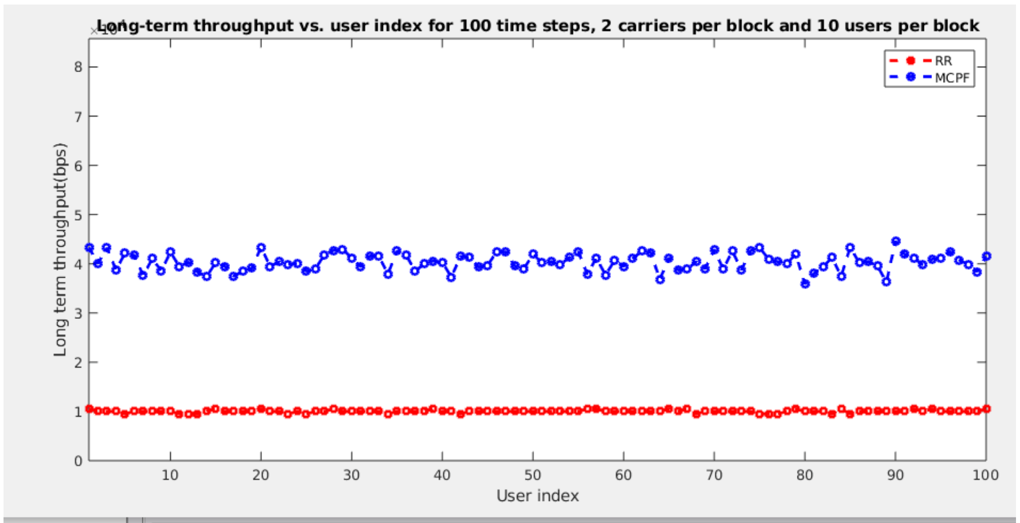
For the MCPF scheme, the increase in the number of users reduces the deviation



```
Command Window
>> mcpf
>> mcbpf
Standard deviation for scheme RR is 0
Standard deviation for scheme MCPF is 5.074571e+03
```

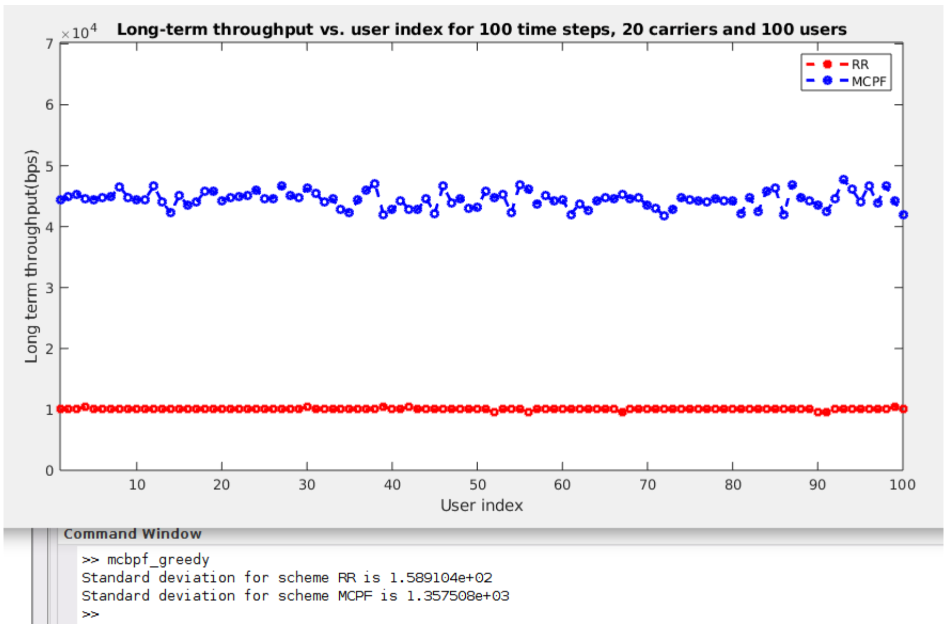
Figure 5.4: Long-term throughput vs. user index for block PF scheduling 20 user-20 carrier-5 block system

in the user rate. In both Fig. 5.3 and Fig. 5.4, the RR scheme is fairer than the MCPF scheme. Fig. 5.4 and Fig. 5.5 are both 20 carrier-100 user systems but have number of blocks equal to 5 and 10 respectively. The deviation seen in the final average user rates for both the schemes in Fig. 5.4 and Fig. 5.5 falls around the same range. **Increasing the number of blocks does not affect the throughput.** More number of blocks will require more number of processor cores (compute resources) which can be provided on a demand basis. This shows that the proposed technique is scalable.



```
Command Window
>> mcbpf
Standard deviation for scheme RR is 2.659080e+02
Standard deviation for scheme MCPF is 1.807505e+03
>>
```

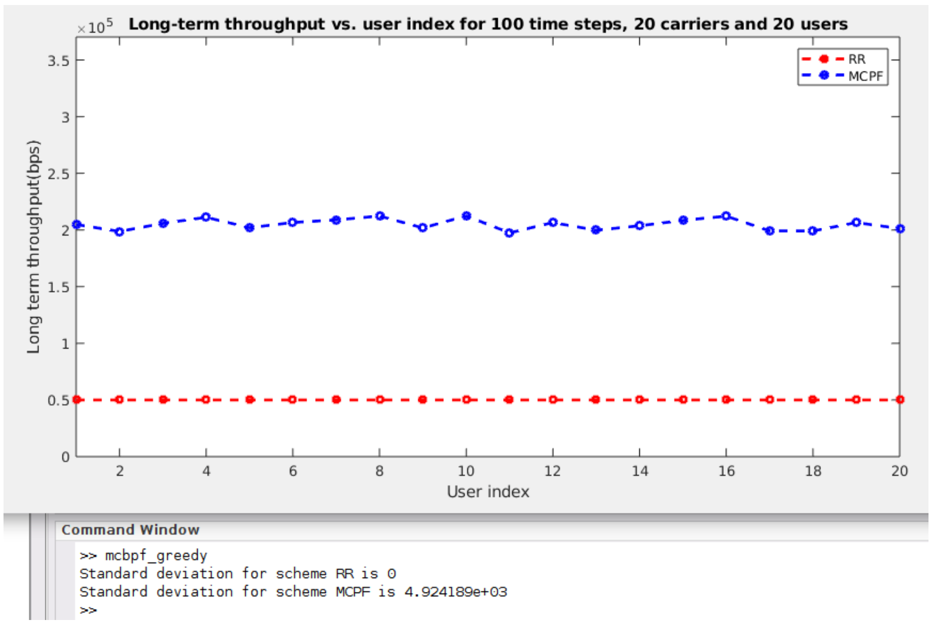
Figure 5.5: Long-term throughput vs. user index for block PF scheduling 20 carrier-100 user-10 block system



```
>> mcbpf_greedy
Standard deviation for scheme RR is 1.589104e+02
Standard deviation for scheme MCPF is 1.357508e+03
>>
```

Figure 5.6: Long-term throughput vs. user index for greedy user partitioning - block PF scheduling 20 carrier-100 user-5 block system

5.4 The Greedy User Partitioning - Block PF scheduler approach

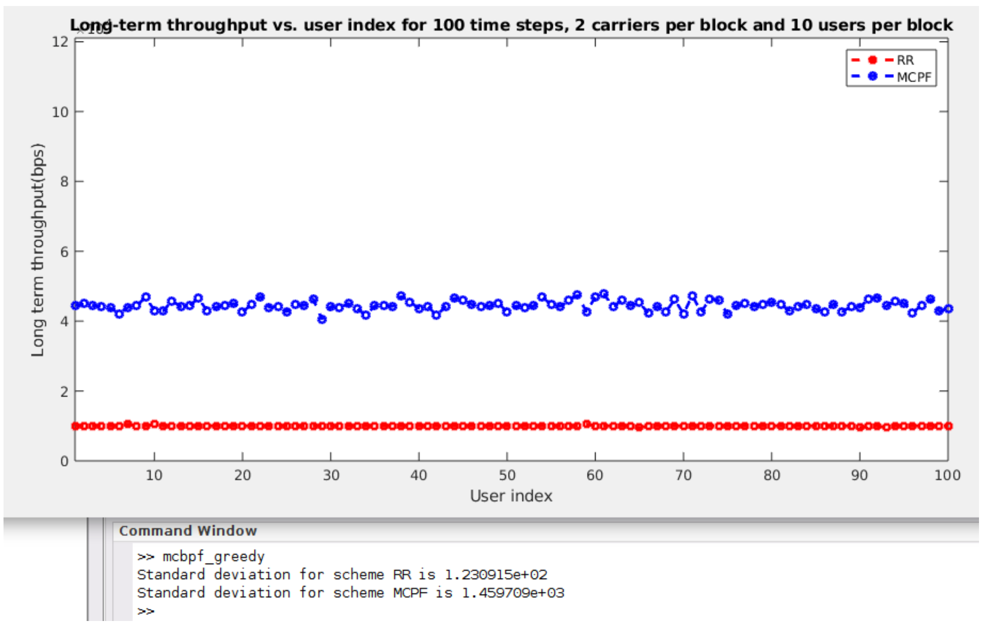


```
>> mcbpf_greedy
Standard deviation for scheme RR is 0
Standard deviation for scheme MCPF is 4.924189e+03
>>
```

Figure 5.7: Long-term throughput vs. user index for greedy user partitioning - block PF scheduling 20 user-20 carrier-5 block system

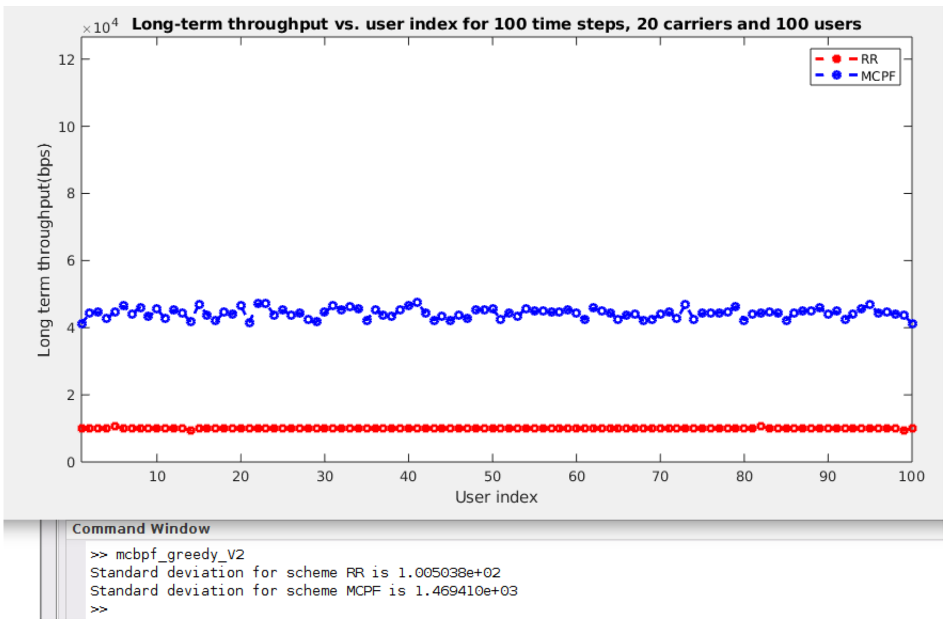
One of the variants of the block proportional fair scheduling schemes incorporates the notion of greediness while partitioning the user set. As described in Section 4.2.1, the Greedy User and Balanced Carrier Partition approach clusters the users based on how high their data rates are, on a given carrier. The carriers are partitioned in a way that ensures a load balance across all blocks. The MCPF and RR cases have been explored here as well. Fig. 5.6 represents the long-term throughput for a 20 carrier-100 user system and Fig. 5.7 represents the long-term throughput for a 20 user-20 carrier system. Here, the number of blocks is equal to 5.Fig. 5.8 represents the long-term throughput for a 20 carrier-100 user system that has **10 blocks**. Given a finite number of carriers in the system, a fair experience is guaranteed as long as there are not too many users. For the 20 user-20 carrier case, the RR scheme works perfectly as expected, but as the number of users increases, there is some finite deviation seen in the user rates. The deviation in the MCPF scheme decreases as the number of user increases.

From Fig. 5.3 and Fig. 5.6, we observe that the deviation obtained on greedy partitioning of users is lesser than the deviation obtained in the proposed Block PF scheduling approach (previous section). **Greedy scheduling seems to be fairer than a load-balancing divisive strategy.** Fig. 5.6 and Fig. 5.8 are both 20 carrier-100 user systems but have number of blocks equal to 5 and 10 respectively. The deviation seen in the final average user rates for both the schemes in Fig. 5.6 and Fig. 5.8 falls around the same range. **Increasing the number of blocks does not affect the throughput.**



```
>> mcbpf_greedy
Standard deviation for scheme RR is 1.230915e+02
Standard deviation for scheme MCPF is 1.459709e+03
>>
```

Figure 5.8: Long-term throughput vs. user index for greedy user partitioning - block PF scheduling 20 carrier-100 user-10 block system



```
>> mcbpf_greedy_V2
Standard deviation for scheme RR is 1.005038e+02
Standard deviation for scheme MCPF is 1.469410e+03
>>
```

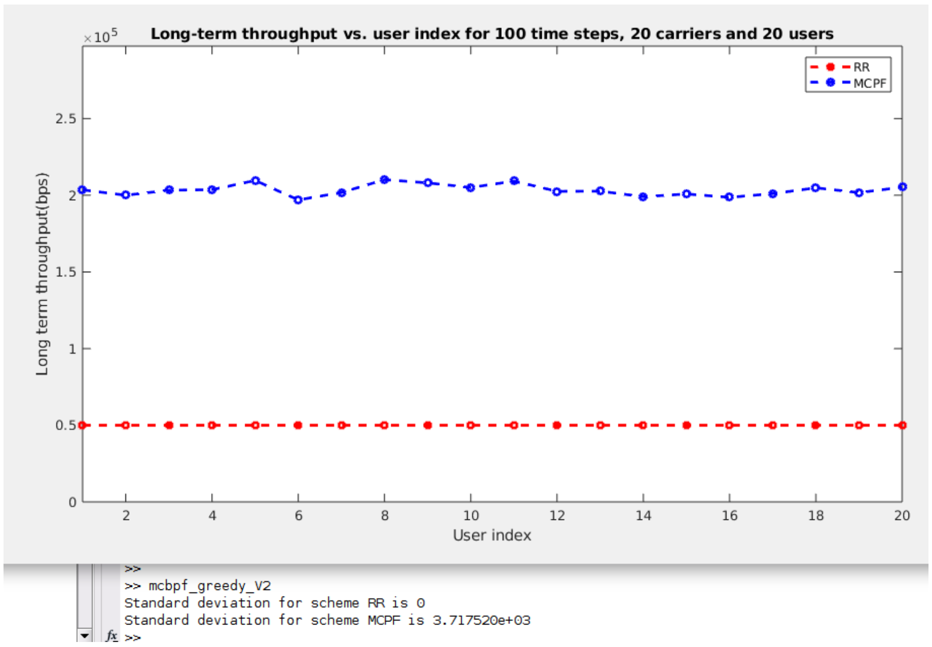
Figure 5.9: Long-term throughput vs. user index for greedy carrier partitioning - block PF scheduling 20 carrier-100 user-5 block system

5.5 The Greedy Carrier Partitioning - Block PF scheduler approach

The other variant of the block proportional fair scheduling schemes incorporates the notion of greediness while partitioning the carrier set. As described in Section 4.2.2, the Random User and Greedy Carrier Partition partitions the carriers in a way that ensures a load balance across all blocks. The carrier is referred to as greedy since that user which has the highest data rate on that carrier is chosen and alloted to that block. The MCPF and RR cases have been explored here as well. Fig. 5.9 represents the long-term throughput for a 20 carrier-100 user system and Fig. 5.10 represents the long-term throughput for a 20 user-20 carrier system. Here, the number of blocks is equal to 5. Fig. 5.11 represents the long-term throughput for a 20 carrier-100 user system that has **10 blocks**. Given a finite number of carriers in the system, a fair experience is guaranteed as long as there are not too many users. For the 20 user-20 carrier case, the RR scheme works perfectly as expected, but as the number of users increases, there is some finite deviation seen in the user rates. The deviation in the MCPF scheme decreases as the number of user increases.

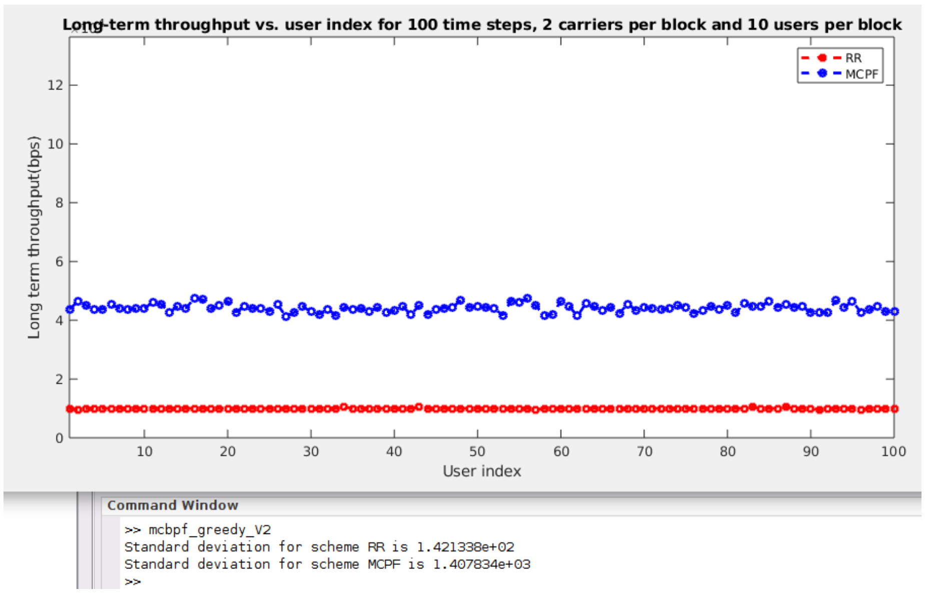
From Fig. 5.6 and Fig. 5.9, we observe that the deviation obtained on greedy partitioning of users is almost the same as that obtained on greedy partitioning of carriers.

Greedy scheduling comes at an advantage here since the users with the best rates are scheduled with priority. Fig. 5.9 and Fig. 5.11 are both 20 carrier-100 user systems but have number of blocks equal to 5 and 10 respectively. The deviation seen in the final average user rates for both the schemes in Fig. 5.9 and Fig. 5.11 falls around the same range. Increasing the number of blocks does not affect the throughput.



```
>>
>> mcbpf_greedy_V2
Standard deviation for scheme RR is 0
Standard deviation for scheme MCPF is 3.717520e+03
fx >>
```

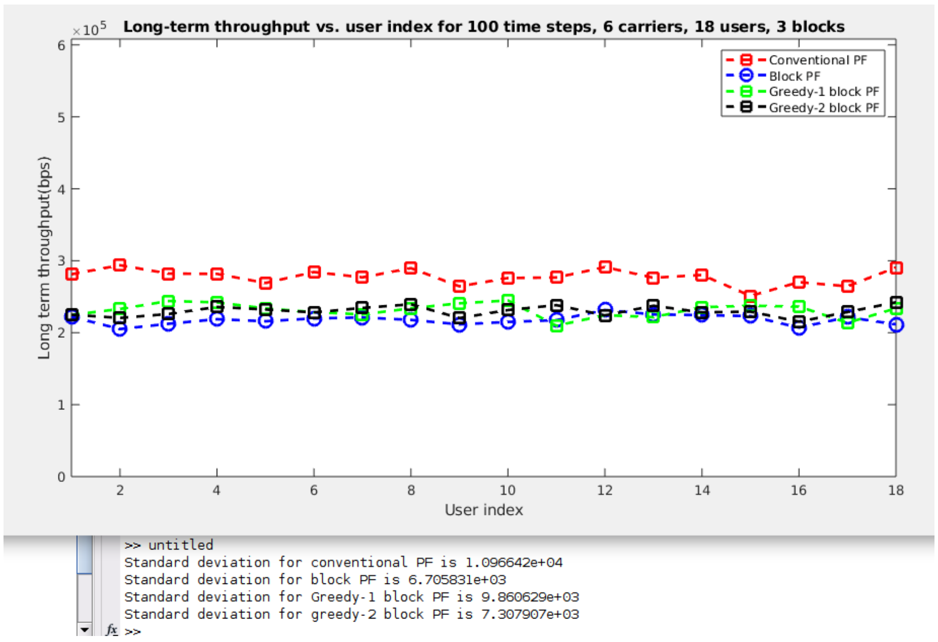
Figure 5.10: Long-term throughput vs. user index for greedy carrier partitioning - block PF scheduling 20 user-20 carrier-5 block system



```
>> mcbpf_greedy_V2
Standard deviation for scheme RR is 1.421338e+02
Standard deviation for scheme MCPF is 1.407834e+03
>>
```

Figure 5.11: Long-term throughput vs. user index for greedy carrier partitioning - block PF scheduling 20 carrier-100 user-10 block system

Fig. 5.12 represents the long-term throughput of a 18 user-6 carrier system. The conventional PF, block PF scheduler, greedy-user and greedy-carrier scheduling schemes are studied for the 18 user-6 carrier system. The number of blocks is equal to 3. As one can observe, the long-term throughput is the highest for the conventional PF scheme. The other schemes offer a reasonably good amount of throughput. The standard deviation in the final user rates of the users is also displayed in Fig. 5.12. All the block PF schemes have a relatively smaller deviation (overall) as compared to the conventional PF scheme. This indicates that the block PF schemes are ensuring a certain amount of fairness while scheduling users onto carriers. In-built MATLAB functionalities were used in order to measure the time taken by each of the routines to schedule the users onto carriers for 100 time steps. Scheduling of users on the 18 user-6 carrier system took around 5 milliseconds for the block PF case and around 11 milliseconds for the greedy scheduling routines. However, the conventional PF scheduling routine took around 3603 seconds (an hour) to schedule on the 18 user-6 carrier system. This is clearly indicative of the power and computational efficiency of the the block PF scheme.



```
>> untitled
Standard deviation for conventional PF is 1.096642e+04
Standard deviation for block PF is 6.705831e+03
Standard deviation for Greedy-1 block PF is 9.860629e+03
Standard deviation for greedy-2 block PF is 7.307907e+03
fx >>
```

Figure 5.12: Long-term throughput vs. user index for various PF scheduling schemes on a 18 user-6 carrier-3 block system

To conclude, the experimental results establish beyond doubt that the proposed technique is scalable with the number of users and carriers. Provisioning of additional computational resources depending on the number of users and carriers is a practically viable solution and is currently implemented in many cloud-based setups. The provisioning of the resources can be done in real time on demand.

CHAPTER 6

CONCLUSION

With the advent of exponential growth in mobile internet usage, there is an imminent need for the network components to handle large amount of data traffic in a scalable fashion. The reputation of the Internet Service Provider (ISP) crucially depends upon the bandwidth offered to its customers in an uninterrupted fashion. This is often termed as QoS. To achieve desired QoS, one of the fundamental need is to have scalable resource schedulers in the data transmission path. This thesis presents one such solution in the context of multi-user multi-carrier wireless communication systems. The scalability of the solution is achieved through decomposition of the problem into multiple sub-problems, each concurrently executing without depending on each other. This enables provision of multi-core environments as a part of the network processing to quickly execute the algorithms, resulting in real-time scheduling. Experimental results are reported that proves the scalability of this methodology and also maximises the throughput. The salient feature of the contribution is the use of an optimal but time consuming algorithm as a sub-routine to solve the larger problem quickly and efficiently. Some of the open questions that could be further explored include,

1. Analytical proof of scalability of the given technique.
2. Better user and carrier partitioning strategies.
3. Parallel implementation on a real system.

APPENDIX A

SOURCE CODE

This function is responsible for generating the set of all possible resource allocation patterns.

```
% Author: Bharati. K
% The following code generates the "next" pattern in the set ...
    of all possible resource
% allocations from which the scheduler picks the best ...
    allocation strategy
% for that time slot.

function A = next_resource_allocate (A,C,U)
    flag1 = 1;
    while (flag1)
        flag2 = 1;
        for i = C: -1 :1
            for j = 1 : U
                if (flag2)
                    if ( (A(i,j) == 1) && (j≠U) )
                        A(i,j) = 0;
                        A(i,j+1) = 1;
                        flag1 = 0;
                        flag2 = 0;
                        break;
                    else
                        if ( (A(i,j) == 1) && (j==U) )
                            A(i,j) = 0;
                            A(i,1) = 1;
                            flag1 = 0;
                        end
                    end
                end
            end
        end
    end
end
```

This function is responsible for carrying out the PF scheduling routine as described in Kim *et al.* (2004).

```
% Author: Bharati. K
% PF scheduling routine as described in paper "A Proportional ...
    Fair Scheduling for
% Multicarrier Transmission Systems" by Hoon Kim et al.

function [R, total_user_alloc] = ...
    PFschedule(r_select,U_block,C_block,T,A,R,dummy,total_user_alloc);
    max_product = 0;
    maxCU = dummy;

    %% For each allocator A over all allocations
    for j = 1: U_block^C_block
        A = next_resource_allocate(A,C_block,U_block);
        product = 1;
        S = r_select.*A;
        agg_user_rate = sum(S,1);

        for k = 1:U_block
            if (T>1)
                product = product * ( 1 + ...
                    agg_user_rate(k)/((T-1)*R(k)) );
            else
                product = product * ( 1 + agg_user_rate(k));
            end
        end

        if (product > max_product)
            maxCU = A;
            max_product = product;
        end
    end

    total_user_alloc = total_user_alloc + sum(r_select.*maxCU,1);
    S1 = r_select.*maxCU;
    max_agg_user_rate = sum(S1,1);

    for k = 1:U_block
        if (T > 1)
            R(k) = ( (T-1)*R(k) + max_agg_user_rate(k) )/T;
        else
            R(k) =  R(k) + max_agg_user_rate(k) ;
        end
    end

end
```

The following code aims to verify the conventional "proportional fair scheduling" as proposed in Kim *et al.* (2004).

```
% Author: Bharati. K
% The following code aims to verify the "proportional fair ...
    scheduling"
% as proposed in the paper "A Proportional Fair Scheduling for
% Multicarrier Transmission Systems" by Hoon Kim et al.
% The simulation parameters are taken as mentioned in the paper.
% The simulations are carried out for three schemes - Single ...
    carrier
% proportional fair scheduling, round robin scheduling and ...
    multi carrier
% proportional fair scheduling.

% SIMULATION PARAMETERS

N = 100; % Number of iterations
U = 40; % Number of users in the User set
BW = 1e+06; %BW of channel (Hz)
SIR_dB = 10; %Average SIR in dB scale
SIR = 10^(SIR_dB/10); %SIR converted from dB scale to normal scale
TS = 0.001; %Duration of one time slot
SD = 0.0001; %Duraton of one symbol
NUM_SYM = TS/SD; %Number of symbols per time slot
```

```
BPS = 2; %Bits per symbol (BPSK - 1, QPSK - 2, 16QAM - 4, ...
    64QAM - 8)
dr= NUM_SYM * BPS/TS; %BW of single carrier

% OTHER USEFUL PARAMETERS
e = 0.01;
R = e.*ones(1,U); % Initialising the average rate for users
agg_user_rate = zeros(1,U); %Aggregate user rates
total_user_alloc = zeros(3,U); %Total rate allocated to the users
Tot = zeros(3,U); %Final average rate of the users
stdev = zeros(1,3); %Standard deviation in the final average ...
    rates of users
B = [1 zeros(1,U-1)];

% MAIN CODE

for ind = 1:3
    for T=1:N
        % Generate the 'r' matrix
        if ind==1 %Single carrier PF
            C = 1;
            x = sqrt(1/2).*(randn(C,U) + 1.0i.*randn(C,U));
            y = abs(x).^2;
            r = (BW/C).*log2(1 + y.*SIR);

        elseif ind==2 %Round robin
            C = 4;
            r = (BW/C).*ones(C,U);

        else %Multi carrier PF
            C = 4;
            x = sqrt(1/2).*(randn(C,U) + 1.0i.*randn(C,U));
            y = abs(x).^2;
            r = (BW/C).*log2(1 + y.*SIR);

        end

        %{
         r is instantaneous transmittable rate of the carrier ...
             when transmitted for
         a user at the current slot and BW is the bandwidth of the
         aggregated carriers.
        %}

        A = zeros(C,U); %Allocation matrix (No. of carriers ...
            versus no. of users)
        dummy = zeros(C,U); %That allocation which gives ...
            maximum data rate

        max_product = 0;
        maxCU = dummy;

        for i = 1:C
            A(i,:) = B;
        end

        %% For each allocator A over all allocations
        for j = 1: U^C
            A = next_resource_allocate(A,C,U);
            product = 1;
            S = r.*A;
            agg_user_rate = sum(S,1);

            for k = 1:U
                if (T>1)
                    product = product * ( 1 + ...
                        agg_user_rate(k)/((T-1)*R(k)) );
                else
                    product = product * ( 1 + agg_user_rate(k));
                end
            end

            if (product > max_product)
                maxCU = A;
                max_product = product;
            end
        end

        total_user_alloc(ind,:) = total_user_alloc(ind,:) + ...
            sum(r.*maxCU,1);
        S1 = r.*maxCU;
        max_agg_user_rate = sum(S1,1);

        for k = 1:U
            if (T > 1)
                R(k) = ( (T-1)*R(k) + max_agg_user_rate(k) )/T;
            else
                R(k) =  R(k) + max_agg_user_rate(k) ;
            end
        end

    end
    Tot(ind,:) = total_user_alloc(ind,:)./N;
    stdev(ind) = std(Tot(ind,:));
end

%------------------------------ PLOTS ...
    -----------------------------------%

k1 = Tot(1,:);
plot(1:U,k1,'--rs','LineWidth',2,'MarkerSize',10,'MarkerEdgeColor','r')
hold on
k2 = Tot(2,:);
plot(1:U,k2,'b--o','LineWidth',2,'MarkerSize',10,'MarkerEdgeColor','b')
hold on
k3 = Tot(3,:);
plot(1:U,k3,'--gs','LineWidth',2,'MarkerSize',10,'MarkerEdgeColor','g')
hold off
title(['Long-term throughput vs. user index for ',num2str(N),' ...
    time steps, ',num2str(C),' carriers and ',num2str(U),' users'])
xlabel('User index')
ylabel('Long term throughput(bps)')
legend('SCPF','RR','MCPF');
axis([1 U 0  max(Tot(:))*10])

%Display outputs
fprintf("Standard deviation for scheme SCPF is %d\n",stdev(1));
fprintf("Standard deviation for scheme RR is %d\n",stdev(2));
fprintf("Standard deviation for scheme MCPF is %d\n",stdev(3));
```

The following code demonstrates the case presented in Section 4.1.

```
% Author: Bharati. K
% The following code aims to perform a "block-proportional ...
    fair scheduling"
% for Multicarrier Transmission Systems". The partitioning of ...
    users and
% carriers is done so as to ensure a primary load balance.
% The simulations are carried out for two schemes - Round ...
    robin scheduling
% and multi carrier proportional fair scheduling.

% BLOCK PARAMETERS
N = 100; % Number of iterations
num_block = 5; %Number of blocks
U_block = 4; %Number of users per block
C_block = 4; %Number of carriers per block
U = num_block*U_block; % Number of users in the User set
C = num_block*C_block; % Total number of carriers
C_select = zeros(num_block,C_block);
U_select = zeros(num_block,U_block);

% SIMULATION PARAMETERS
SIR_dB = 10; %Average SIR in dB scale
SIR = 10^(SIR_dB/10); %SIR converted from dB scale to normal scal
TS = 0.001; %Duration of one time slot
SD = 0.0001; %Duraton of one symbol
NUM_SYM = TS/SD; %Number of symbols per time slot
BPS = 2; %Bits per symbol (BPSK - 1, QPSK - 2, 16QAM - 4, ...
    64QAM - 8)
dr= NUM_SYM * BPS/TS; %BW of single carrier
BW = 1e+06; %BW of channel

% OTHER USEFUL PARAMETERS
e = 0.01; %Initialising the average rate for 'U' users
R = e.*ones(1,U); %Each user is allocated 'e' units
R_select = zeros(1,U_block); %avg user rate for a sub-matrix
A = zeros(C_block,U_block); %Allocation matrix (No. of ...
    carriers versus no. of users)
```

```
B = [1 zeros(1,U_block-1)];
r_select = zeros(C_block,U_block); % Inst data rate of ...
    selected users and carriers of a block
dummy = zeros(C_block,U_block);
total_user_alloc = zeros(2,U);
total_user_alloc_select = zeros(1,U_block); %User alloc for a ...
    sub-matrix
Tot = zeros(2,U); %Final average rate of the users
stdev = zeros(1,2); %Standard deviation in the final average ...
    rates of users

% INITIAL ALLOCATION MATRIX
for i = 1:C_block
    A(i,:) = B;
end

% MAIN CODE

for ind = 1:2
    for T=1:N
        % Generate the 'r' matrix
        if ind==1 %Round robin
            r = (BW/C).*ones(C,U);

        else %Multi carrier PF
            x = sqrt(1/2).*(randn(C,U) + 1.0i.*randn(C,U));
            y = abs(x);
            r = (BW/C).*log2(1 + y.*SIR);
        end

        inv_R = 1./R;

        c_sort = sum(r, 2); %Total carrier rate of 'C' carriers
        [¬, Cp] = sort(transpose(c_sort)); %Cp is the array of ...
            indices of carriers in increasing order of inst rate
        [¬, Us] = sort(inv_R); %Us is the array of indices of ...
            users in increasing order of avg rate

        %% Selecting users and carriers for every block
        %Select carriers for each block from the sorted list ...
```

```
            with load
        %balancing

        L_carr_select = 1;
        R_carr_select = C;
        flag1 = 0;

        for j = 1:num_block
            for l = 1:C_block
                if flag1 == 0
                    C_select(j,l) = Cp(L_carr_select);
                    L_carr_select = L_carr_select + 1;
                    flag1 = 1;
                else
                    C_select(j,l) = Cp(R_carr_select);
                    R_carr_select = R_carr_select - 1;
                    flag1 = 0;
                end
            end
        end

        %Select users for each block from the sorted list with ...
            load
        %balancing

        L_user_select = 1;
        R_user_select = U;
        flag2 = 0;

        for j=1:num_block
            for l = 1:U_block
                if flag2 == 0
                    U_select(j,l) = Us(L_user_select);
                    L_user_select = L_user_select + 1;
                    flag2 = 1;
                else
                    U_select(j,l) = Us(R_user_select);
                    R_user_select = R_user_select - 1;
                    flag2 = 0;
                end
            end
        end

        for j=1:num_block
            %choose carriers and users of 'j'th block
            C_choose = C_select(j,:); %Choose the carriers of ...
               the 'j'th block
            U_choose = U_select(j,:);

            %Choosing the sub-matrix 'r_select' for carriers ...
               and users of 'j'th block
            for k=1:C_block
                for l=1:U_block
                    r_select(k,l) = r(C_choose(k),U_choose(l));
                    R_select(l) = R(U_choose(l));
                    total_user_alloc_select(l) = ...
                       total_user_alloc(ind, U_choose(l));
                end
            end

            %Scheduling
            [R_select, total_user_alloc_select] = ...
               PFschedule(r_select,U_block,C_block,T,A,R_select,dummy,total_use

            for l = 1:U_block
                R(U_choose(l)) = R_select(l);
                total_user_alloc(ind, U_choose(l)) =  ...
                   total_user_alloc_select(l);
            end

        end
    end
    Tot(ind,:) = total_user_alloc(ind,:)./N;
    stdev(ind) = std(Tot(ind,:));
end

% PLOTS
x =1:U;
k1 = Tot(1,:);
plot(x,k1,'--rs','LineWidth',2,'MarkerSize',8,'MarkerEdgeColor','r')
```

```
hold on
k2 = Tot(2,:);
plot(x,k2,'b--o','LineWidth',2,'MarkerSize',8,'MarkerEdgeColor','b')
hold off
title(['Long-term throughput vs. user index for ',num2str(N),' ...
    time steps, ',num2str(C),' carriers and ',num2str(U),' users'])
xlabel('User index')
ylabel('Long term throughput(bps)')
legend('RR','MCPF');
axis([1 U 0  max(Tot(:))*10])

%Display outputs
fprintf("Standard deviation for scheme RR is %d\n",stdev(1));
fprintf("Standard deviation for scheme MCPF is %d\n",stdev(2));
```

The following code demonstrates the case presented in Section 4.2.1. The function routine "user select" is used to patition the user set. Note that the scheduling routine described in Kim *et al.* (2004) is applicable after partitioning the user and carrier sets.

```
% Author: Bharati. K
% The following code aims to perform a "blockproportional fair ...
    scheduling"
% for Multicarrier Transmission Systems". The allocation of ...
    users is done
% in a greedy fashion while that of carriers is balanced.
% The simulations are carried out for two schemes - Round ...
    robin scheduling
% and multi carrier proportional fair scheduling.

% BLOCK PARAMETERS
N = 100; % Number of iterations
num_block = 5; %Number of blocks
U_block = 20; %Number of users per block
C_block = 4; %Number of carriers per block
U = num_block*U_block; % Number of users in the User set
C = num_block*C_block; % Total number of carriers
C_select = zeros(num_block,C_block);

% SIMULATION PARAMETERS
```

```
SIR_dB = 10; %Average SIR in dB scale
SIR = 10^(SIR_dB/10); %SIR converted from dB scale to normal scal
TS = 0.001; %Duration of one time slot
SD = 0.0001; %Duraton of one symbol
NUM_SYM = TS/SD; %Number of symbols per time slot
BPS = 2; %Bits per symbol (BPSK - 1, QPSK - 2, 16QAM - 4, ...
    64QAM - 8)
dr= NUM_SYM * BPS/TS; %BW of single carrier
BW = 1e+06; %BW of channel

% OTHER USEFUL PARAMETERS
e = 0.01; %Initialising the average rate for 'U' users
R = e.*ones(1,U); %Each user is allocated 'e' units
R_select = zeros(1,U_block); %avg user rate for a sub-matrix
A = zeros(C_block,U_block); %Allocation matrix (No. of ...
    carriers versus no. of users)
B = [1 zeros(1,U_block-1)];
r_select = zeros(C_block,U_block); % Inst data rate of ...
    selected users and carriers of a block
dummy = zeros(C_block,U_block);
total_user_alloc = zeros(2,U);
total_user_alloc_select = zeros(1,U_block); %User alloc for a ...
    sub-matrix
Tot = zeros(2,U); %Final average rate of the users
stdev = zeros(1,2); %Standard deviation in the final average ...
    rates of users

% INITIAL ALLOCATION MATRIX
for i = 1:C_block
    A(i,:) = B;
end

% MAIN CODE

for ind = 1:2
    for T=1:N
        % Generate the 'r' matrix
        if ind==1 %Round robin
            r = (BW/C).*ones(C,U);

        else %Multi carrier PF
            x = sqrt(1/2).*(randn(C,U) + 1.0i.*randn(C,U));
            y = abs(x);
            r = (BW/C).*log2(1 + y.*SIR);
        end

        c_sort = sum(r, 2);
        [c_sorted, Cp] = sort(transpose(c_sort));

        %% Selecting users and carriers for every block
        %Select carriers for each block from the sorted list ...
            with load
        %balancing

        L_carr_select = 1;
        R_carr_select = C;
        flag = 0;

        for j = 1:num_block
            %Select carriers for each block from the sorted ...
                list with load
            %balancing
            for l = 1:C_block
                if flag == 0
                    C_select(j,l) = Cp(L_carr_select);
                    L_carr_select = L_carr_select + 1;
                    flag = 1;
                else
                    C_select(j,l) = Cp(R_carr_select);
                    R_carr_select = R_carr_select - 1;
                    flag = 0;
                end
            end
        end

        % Call user set partitioning routine
        U_select = ...
            user_select(r,C_select,num_block,U_block,C_block);

        for j = 1:num_block
```

```
            %choose carriers and users of 'j'th block
            C_choose = C_select(j,:); %Choose the carriers of ...
               the 'j'th block
            U_choose = U_select(j,:);

            %Choosing the sub-matrix 'r_select' for carriers ...
               and users of 'j'th block
            for k=1:C_block
                for l=1:U_block
                    r_select(k,l) = r(C_choose(k),U_choose(l));
                    R_select(l) = R(U_choose(l));
                    total_user_alloc_select(l) = ...
                       total_user_alloc(ind,U_choose(l));
                end
            end

            %Scheduling
            [R_select, total_user_alloc_select] = ...
               PFschedule(r_select,U_block,C_block,T,A,R_select,dummy,total_use

            for l = 1:U_block
                R(U_choose(l)) = R_select(l);
                total_user_alloc(ind,U_choose(l)) =  ...
                   total_user_alloc_select(l);
            end
        end
    end
    Tot(ind,:) = total_user_alloc(ind,:)./N;
    stdev(ind) = std(Tot(ind,:));
end

% PLOTS
x =1:U;
k1 = Tot(1,:);
plot(x,k1,'--rs','LineWidth',2,'MarkerSize',8,'MarkerEdgeColor','r')
hold on
k2 = Tot(2,:);
plot(x,k2,'b--o','LineWidth',2,'MarkerSize',8,'MarkerEdgeColor','b')
hold off
title(['Long-term throughput vs. user index for ',num2str(N),' ...
    time steps, ',num2str(C),' carriers and ',num2str(U),' users'])
xlabel('User index')
ylabel('Long term throughput(bps)')
legend('RR','MCPF');
axis([1 U 0  max(Tot(:))*10])

%Display outputs
fprintf("Standard deviation for scheme RR is %d\n",stdev(1));
fprintf("Standard deviation for scheme MCPF is %d\n",stdev(2));
```

```
% Author: Bharati. K
%The following function partitions the user set in a greedy ...
    function. Refer
%to section 4.2.1 of thesis report for more details.

function U_select = ...
    user_select(r,C_select,num_block,U_block,C_block)
    U_select = zeros(num_block,U_block);
    global_user_list = 1:num_block*U_block; %Global list of users
    r_select = zeros(C_block,num_block*U_block);

    for m = 1:round(U_block/C_block)
        for i = 1: num_block
            k = C_block*(m - 1) + 1; %column index variable
            user_list = [];
            C_choose = C_select(i,:); %Choose carriers in a ...
                given block
            for n=1:C_block
                r_select(n,:) = r(C_choose(n),:);
                [¬,user_index] = max(r_select(n,:));

                if (ismember(user_index, global_user_list))
                    user_list = [user_list user_index];
                    global_user_list = ...
                        global_user_list(global_user_list ≠ ...
                        user_index);
                end
            end

            L = length(user_list);
            if L > C_block
                user_list = user_list(randperm(L,C_block));

            elseif L < C_block
                while (L ≠ C_block)
                    num = randi(num_block*U_block);
                    if (ismember(num, global_user_list))
                        user_list = [user_list num];
                        global_user_list = ...
                            global_user_list(global_user_list ≠ ...
                            num);
                        L = L + 1;
                    end
                end
            end

            for l = 1:length(user_list)
                U_select(i,k) = user_list(l);
                k = k + 1;
                r(:,user_list(l)) = 0;
            end
        end
    end
end
```

The following code demonstrates the case presented in Section 4.2.1. The function routine "user carr select" is used to patition the user set. Note that the scheduling routine described in Kim *et al.* (2004) is applicable after partitioning the user and carrier sets.

```
% Author: Bharati. K
% The following code aims to perform a "blockproportional fair ...
    scheduling"
% for Multicarrier Transmission Systems". The allocation of ...
    carriers is done
% in a greedy fashion while that of users is random.
% The simulations are carried out for two schemes - Round ...
    robin scheduling
% and multi carrier proportional fair scheduling.

% BLOCK PARAMETERS
N = 100; % Number of iterations
num_block = 5; %Number of blocks
U_block = 12; %Number of users per block
C_block = 4; %Number of carriers per block
U = num_block*U_block; % Number of users in the User set
C = num_block*C_block; % Total number of carriers
C_select = zeros(num_block,C_block);

% SIMULATION PARAMETERS
SIR_dB = 10; %Average SIR in dB scale
SIR = 10^(SIR_dB/10); %SIR converted from dB scale to normal scal
TS = 0.001; %Duration of one time slot
SD = 0.0001; %Duraton of one symbol
NUM_SYM = TS/SD; %Number of symbols per time slot
BPS = 2; %Bits per symbol (BPSK - 1, QPSK - 2, 16QAM - 4, ...
    64QAM - 8)
dr= NUM_SYM * BPS/TS; %BW of single carrier
BW = 1e+06; %BW of channel

% OTHER USEFUL PARAMETERS
e = 0.01; %Initialising the average rate for 'U' users
R = e.*ones(1,U); %Each user is allocated 'e' units
R_select = zeros(1,U_block); %avg user rate for a sub-matrix
A = zeros(C_block,U_block); %Allocation matrix (No. of ...
    carriers versus no. of users)
B = [1 zeros(1,U_block-1)];
r_select = zeros(C_block,U_block); % Inst data rate of ...
    selected users and carriers of a block
dummy = zeros(C_block,U_block);
total_user_alloc = zeros(2,U);
total_user_alloc_select = zeros(1,U_block); %User alloc for a ...
    sub-matrix
Tot = zeros(2,U); %Final average rate of the users
stdev = zeros(1,2); %Standard deviation in the final average ...
    rates of users

% INITIAL ALLOCATION MATRIX
for i = 1:C_block
```

```
    A(i,:) = B;
end

% MAIN CODE

for ind = 1:2
    for T=1:N
        % Generate the 'r' matrix
        if ind==1 %Round robin
            r = (BW/C).*ones(C,U);

        else %Multi carrier PF
            x = sqrt(1/2).*(randn(C,U) + 1.0i.*randn(C,U));
            y = abs(x);
            r = (BW/C).*log2(1 + y.*SIR);
        end

        c_sort = sum(r, 2);
        [c_sorted, Cp] = sort(transpose(c_sort));

        %% Selecting users and carriers for every block
        %Select carriers for each block from the sorted list ...
            with load
        %balancing

        L_carr_select = 1;
        R_carr_select = C;
        flag = 0;

        for j = 1:num_block
            %Select carriers for each block from the sorted ...
                list with load
            %balancing
            for l = 1:C_block
                if flag == 0
                    C_select(j,l) = Cp(L_carr_select);
                    L_carr_select = L_carr_select + 1;
                    flag = 1;
                else
                    C_select(j,l) = Cp(R_carr_select);
                    R_carr_select = R_carr_select - 1;
                    flag = 0;
                end
            end
        end

        % Call user set partitioning routine
        U_select = ...
           user_carr_select(r,C_select,num_block,U_block,C_block);

        for j = 1:num_block
            %choose carriers and users of 'j'th block
            C_choose = C_select(j,:); %Choose the carriers of ...
               the 'j'th block
            U_choose = U_select(j,:);

            %Choosing the sub-matrix 'r_select' for carriers ...
               and users of 'j'th block
            for k=1:C_block
                for l=1:U_block
                    r_select(k,l) = r(C_choose(k),U_choose(l));
                    R_select(l) = R(U_choose(l));
                    total_user_alloc_select(l) = ...
                       total_user_alloc(ind,U_choose(l));
                end
            end

           %Scheduling
            [R_select, total_user_alloc_select] = ...
               PFschedule(r_select,U_block,C_block,T,A,R_select,dummy,total_use

            for l = 1:U_block
                R(U_choose(l)) = R_select(l);
                total_user_alloc(ind,U_choose(l)) =  ...
                   total_user_alloc_select(l);
            end
        end
    end
    Tot(ind,:) = total_user_alloc(ind,:)./N;
    stdev(ind) = std(Tot(ind,:));
```

```
end

% PLOTS
x =1:U;
k1 = Tot(1,:);
plot(x,k1,'--rs','LineWidth',2,'MarkerSize',8,'MarkerEdgeColor','r')
hold on
k2 = Tot(2,:);
plot(x,k2,'b--o','LineWidth',2,'MarkerSize',8,'MarkerEdgeColor','b')
hold off
title(['Long-term throughput vs. user index for ',num2str(N),' ...
    time steps, ',num2str(C),' carriers and ',num2str(U),' users'])
xlabel('User index')
ylabel('Long term throughput(bps)')
legend('RR','MCPF');
axis([1 U 0  max(Tot(:))*10])

%Display outputs
fprintf("Standard deviation for scheme RR is %d\n",stdev(1));
fprintf("Standard deviation for scheme MCPF is %d\n",stdev(2));
```

```
%Author: Bharati. K
%The following function partitions the user set in a greedy ...
    function. Refer
%to section 4.2.2 of thesis report for more details.

function U_select = ...
    user_carr_select(r_mat,C_select,num_block,U_block,C_block)

    Us = randperm(num_block*U_block); %Random permutation of users
    user_track = zeros(num_block,U_block); %Matrix tracks ...
        which user goes to which block
    non_full_block = 1:num_block; %List of non-full blocks

    for i = 1:length(Us) %for all the 'U' users

        if (length(non_full_block) ≠ 0)

            block = -1;
            user_index = Us(i); %Pick a user index
            %choose carrier index correspoding to the highest ...
                inst rate
            [¬,carr_index] = max(r_mat(:,user_index));

            %find non-full block in which the carrier with ...
                'carr_index' belongs
            [j,¬] = find(C_select == carr_index);

            if ( ¬ismember(j,non_full_block) )
                index = randperm( length(non_full_block),1) ;
                block = non_full_block(index);
            else
                block = j;
            end

            %fill user_index in the block 'block'
            ind = 1; %column index
            for l = 1:U_block
                if user_track(block,l) == 0
                    ind = l;
                    break;
                end
            end

            %Fill in the user_index in user_track(block,ind)
            if (ind < U_block)
                user_track(block, ind) = user_index;

            else
                user_track(block, ind) = user_index;
                non_full_block = non_full_block(non_full_block ...
                    ≠ block);
            end

        end

    end

    U_select = user_track;
```

```

end
```

REFERENCES

1. **Andrews, M.** and **L. Zhang** (2011). Scheduling algorithms for multicarrier wireless data systems. *IEEE/ACM Trans. Netw.*, **19**(2), 447âĂŞ455. ISSN 1063-6692. URL `https://doi.org/10.1109/TNET.2010.2064175`.

2. **Huang, C.**, **H. Su**, **S. Vitebsky**, and **P.-C. Chen**, Schedulers for 1xev-do: Third generation wireless high-speed data systems. volume 3. 2003. ISBN 0-7803-7757-5.

3. **Kelly, F.**, **A. Maulloo**, and **D. Tan** (1998). Rate control for communication networks:shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, **49**.

4. **Kim, H.**, **K. Kim**, **Y. Han**, and **S. Yun**, A proportional fair scheduling for multicarrier transmission systems. volume 1. 2004. ISBN 0-7803-8521-7.

5. **Liu, X.**, **E. K. P. Chong**, and **N. B. Shroff** (2001). Opportunistic transmission scheduling with resource-sharing constraints in wireless networks. *IEEE Journal on Selected Areas in Communications*, **19**(10), 2053–2064.

6. **Tse, D.** and **P. Viswanath**, *Fundamentals of Wireless Communication.* Cambridge University Press, 2005.

7. **Wang Anchun**, **Xiao Liang**, **Zhou Shidong**, **Xu Xibin**, and **Yao Yan**, Dynamic resource management in the fourth generation wireless systems. *In International Conference on Communication Technology Proceedings, 2003. ICCT 2003.*, volume 2. 2003.