University of Agder

Learning Automata based Shiftable Domestic Load Scheduling in Smart Grid: Accuracy and Fairness

rajan thapa

Master's Thesis, Year 2016, University Of Agder

Master's Thesis

Learning Automata based Shiftable Domestic Load Scheduling in Smart Grid: Accuracy and Fairness

Author:
Rajan Thapa

Supervisor:
Associate Prof. Lei Jiao

January 6, 2017

Preface

This work is a Master thesis, a part of Master in Information and Communication Technology, at University of Agder, Grimstad. My supervisor for this thesis work has been Associate Prof. Dr. Lei Jiao, University of Agder, Grimstad.

I would like to express my heartfelt gratitude to my supervisor who has helped me with his inventive and resourceful guidance, suggestions and feedback throughout the course. I am also grateful to my professors, classmates, the university and my friends and family, who were really supportive and helpful until the successful submission of this thesis.

This thesis has been written using LaTeX and the mathematical calculations and simulation results are obtained from MATLAB.

Abstract

In this thesis, investigation is carried out on scheduling of shiftable loads which involves partly selection of loads within the power budget of operator. Domestic shiftable loads are scheduled along multiple timeslots with the considerations of the **accuracy** of scheduling in terms of optimization of capacity and of the **fairness** between appliances in terms of frequency of usage in smart grids. Since the scheduled load can not be over the capacity, the global optimal point is a combination of loads which are most close or equal to but not over the capacity.

This optimization problem is shown to be NP hard, and has been formulated as a potential game. To solve this problem in a distributed manner, Learning Automata (LA) based methods are proposed. Although the LA based methods do not favour any participants of scheduling which can serve as a fair selection in the long run, the fairness among the loads in finite time is still worth studying. To make the scheduling process fair in short time, virtual coin game is employed into the scheduling.

Simulations have been performed by implementing two LA methods, namely BLA and L_{R-I}, under different number of timeslots, with and without consideration of coin game to evaluate and compare the results. Simulation results show that the accuracy in terms of the closeness of the converged result to the global optimal point achieved by both LA based scheduling methods is high and the fairness of the system is increased by applying the virtual coin game.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

BLA	Bayesian Learning Automata
DM	Demand Management
e.g.	for example
FIFO	First In First Out
i.e.	that is
LA	Learning Automata
L_{R-I}	Linear Reward-Inaction
NE	Nash Equilibrium
SG	Smart Grid
s.t.	subject to
VCG	virtual coin game
VLC	virtual logical controller
w.r.t.	with respect to

Chapter 1

Introduction

1.1 Background

There has always been an imbalance between the supply and demand in the society, and the electrical grid system does not undergo any exception. There has been an ever growing increase in electrical power demand while the capacity of the grid system is not increasing proportionately. On the other hand, the technological and communication advancements have been radical in the 21^{st} century which have been enforced in almost all areas for betterment. Such advancements implied into the grid system to make it smart grid has different advantages, one of which is better load scheduling compared with traditional methods, to balance the capacity and demand in electricity distribution system. This prevents the grid system from crashing and total blackout consequently. Such an issue of demand management by taking advantage of new advanced features of smart grid is a hot topic in the present context.

The traditional grid systems do not exhibit features that the modern techniques can be applied, to bring a balance between the ever increasing demands of electricity and capacity, where the increase in capacity of grids is not in harmony to increasing load of grids. An operator, user and energy friendly grid system is required and this has led to development and existence of more advanced electrical grid systems known as smart grid. The differences between the traditional and modern grids is pointed out in [1] as depicted in Fig. 1.1. The existing grid is the traditional grid while the intelligent grid is smart grid.

Smart grid refers to the electrical grid system with smart communication and computational capacity and memory for automated distribution and

Existing Grid	Intelligent Grid
Electromechanical	Digital
One-Way Communication	Two-Way Communication
Centralized Generation	Distributed Generation
Hierarchical	Network
Few Sensors	Sensors Throughout
Blind	Self-Monitoring
Manual Restoration	Self-Healing
Failures and Blackouts	Adaptive and Islanding
Manual Check/Test	Remote Check/Test
Limited Control	Pervasive Control
Few Customer Choices	Many Customer Choices

Figure 1.1: Features of SG compared to traditional grid [1]

control of electricity supply, with improved efficiency, reliability, and safety of power distribution [2]. The standards of different components of SG (building, substation and powerline) and the communication technology is given in details in [3]. The SG operator and its customers are the two main entities in SG. Operator is the service provider who supplies electricity into the grid system and customers are the ones who utilize the electricity from operator through different appliances. They are connected by the wired and wireless channels through which the supply of electricity and two way communication takes place. The users of SG have smart meters through which the users and their appliances communicate with the operator and other users as per requirement. Smart meter is the most prominent feature of modern SG helping in collection and distribution of information to and from users' appliances and SG operator. The main features of smart meter that differentiate modern SG from traditional grid system can be outlined as follows [4]:

- Powerline communication module - To receive electricity demands from appliances and send electricity control signals.
- Wireless communication module - With same features as powerline

communication module but through wireless connection with appliances.

- Processing unit - To perform computational tasks, e.g. decision-making for optimization purpose.

The main problem with traditional grid system that has led to its improvisation into new SG is the lack of power system stability, a balance in the demand and capacity of grid system with power usage optimization. The imbalance due to higher demands leading to distribution failures may lead to catastrophic problems as bad as blackouts [1]. The capacity of SG is not under much control such that it could be maximized or minimized to meet the frequently changing demands. Capacity at the operator or utility company side can be increased by storage of power at low usage times, lowering the transmission losses, etc. But the lower end of SG's organizational hierarchy [5], i.e., the users or customers are more flexible with their demands. So, the demands generated at the customer side can be adjusted for smooth and uninterrupted functioning of grid system. Many demand or load management schemes have been proposed that can be implemented into advanced SGs. During limited power availability, the demands should be curtailed such that the optimal (maximum available) power is used. While doing so, some of the demands have to be cut-off directly or encouraged to do so indirectly. Different strategies have been developed that target to serve the purpose. This thesis work also intends to mitigate the same problem.

1.2 Demand Management

The demand management schemes will follow certain rules for implementation. The rules, based on how the communication flows and who controls the demands to be selected or rejected for power usage, are categorized into centralized or decentralized methods. The process of selecting the loads or demands within the rules of demand management is referred to as scheduling and the appliances or their loads that are selected are called scheduled appliances/loads. The control and management of loads and the system as a whole is in the hands of single scheduler [6] in centralized scheduling whereas the control of loads and distribution of electricity is distributed among local schedulers or the customers themselves in decentralized method. To avoid the disadvantages like a single point of failure [7], fairness of the selection method, communication overhead at operator, scalability and social and legal barriers [8] etc., distributed scheduling methods have been chosen

and described in this thesis. In the distributed method, the scheduling of demands is done between the users themselves without the external intervention of SG operator through any means.

The aim of the distributed methods is to reduce the communicational and computational overhead at the SG operator side with more scalability and privacy, meanwhile the users of SG can enjoy scheduling within themselves without the intervention of SG operator. Distributed scheduling can be performed in smaller parts by breakdown of whole SG system into regions and areas. Whenever a load is generated, information about the load including its size (units in KWh), time duration it lasts (minutes) and its source (appliance which sent the load through IP address or other IDs) is received by only the ones that the information is intended to be shared with.

Whatever the scheduling method, the scheduling is done at the loads' level and the loads are generated by appliances belonging to same or different users. Three different concepts of SG that can influence load scheduling have been outlined as follows.

i. Load can be served by principle of **FIFO (First In First Out)**, i.e., on the basis of first-come-first-serve [9] [10]. This is a very simple and classical method with centralized approach, which is unbiased from SG's standpoint but distance between appliances might play a vital role in selection process. The advantage of this method is that loads can be added into SG for service until the full capacity has been utilized.

ii. Taking **Importance of appliances** in SG from customers' point of view into account, certain appliances can be more important than others. Scheduling methods that employ this property will categorize the appliances and their loads on the basis of their importance. Loads from same category of appliances from any customer will have equal priority while scheduling. The loads from the important appliances will be scheduled first while the latter ones will have to wait. Since the odds for less important loads getting served is low while important loads continuously snatch the capacity, the tolerance for delay to less important loads is an issue.

iii. **Price of Electricity** - The price of electricity is set high during high demand times so that users are forced to change their electricity usage behaviour. This method is more suited to controlling loads from medium and low income users who consider electricity bills as an issue. But due to dependence on users' decisions solely, the scheduling

may not be as accurate as expected. (Optimization of SG's capacity depends on historical data and accuracy of expected values). This method avoids queueing as all load requests are served.

Among the above mentioned concepts, the most discussed and popular demand management concept is control through price of electricity which involves time-variant electricity prices, referred to as dynamic pricing [11] or real-time pricing or time-of-use pricing [12], which is an incentive based method. The demands are changed (or expected to do so) to make the total demand below the capacity by providing discounted price at low peak hours and charging higher electricity rates at peak hours. Dynamic pricing encourages the customers to utilize the electricity more efficiently and wisely by changing their usage behaviour [13]. This method has been quite popular and different approaches for pricing have been proposed by different authors. The benefit from this method to SG operator is that it escapes the extremely high additional cost that could incur to increase the capacity. Even though the dynamic pricing has been hyped as a characteristic of SG, it requires a practical coordination between the operator and its users to achieve the goal [14]. The second type of scheduling concept has been chosen in this thesis where the classification of loads or appliances[1] is done firstly based on their importance.

The customer side demand management (DM) can be done in several ways but all the DM techniques are based on two aspects, reducing consumption and shifting consumption [15]. Scheduling the loads is also a popular method of DM where some of the loads are selected while others are shifted. Loads enter into selection process, so that a number of those loads will be selected under the restriction of capacity availed by SG. Scheduling methods may consider some or all loads for scheduling (e.g., only less important loads are considered for scheduling in the previously mentioned concept of scheduling by importance of appliances). But the selection can not be random, whether the scheduler is the operator itself or any other entity. They should follow certain rules or algorithms or logics that are implemented on and acceptable to all users. Due to the large and sophisticated computation and communication features available in SG, several simple to complex scheduling methods have been proposed by different authors (Papers [16] and [17] use particle swarm optimization methods for scheduling).

The aim of the thesis is to optimize the power capacity utilization of

[1]Loads and appliances undergo same categorization as loads are generated by appliances. So a load from X Type appliance can be called X type load or a appliance that generated a Y type load can be called Y type appliance.

SG by decentralized demand scheduling method with consideration of fairness among appliances. The scheduling is done by considering the shiftable loads from shiftable appliances to undergo distributed scheduling methods as shown in Fig. 1.2. The rules for selection of users are that the loads will undergo random selection based method of learning automata (LA). Maximum capacity exploitation is expected along with unbiased selection by implementation of two LA based methods, namely Bayesian Learning Automata (BLA) and Linear Reward-Inaction (L_{R-I}) with introduction of coin game[2] together.

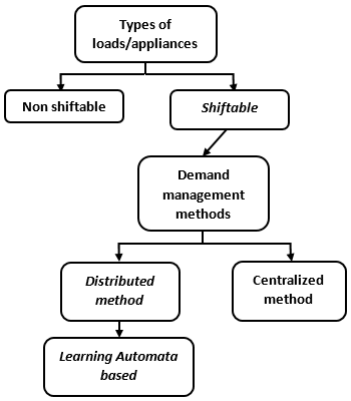


Figure 1.2: Overview of proposed Scheduling Process

1.3 The Scheduling Process

The scheduling is done for demands that are received from appliances in real time rather than the predicted/computed values. It is assumed that there exists no reliable historical information for power forecasting (both demands and capacity) and scheduling accurately. Such predicted calculations will have certain limitations and conditions that need to be avoided. The communication of load values from secondary or shiftable appliances and decision calculations is restricted locally between the customers/appliances and no external influence exists are the conditions predefined for implementing the scheduling methods in this thesis. Due to the distributive property, a new entity or local scheduler called virtual logical controller (VLC) is required. It can effectively communicate with the central SG operator for

[2]Coin game is new proposal in SG field and discussed later in Chapter 4

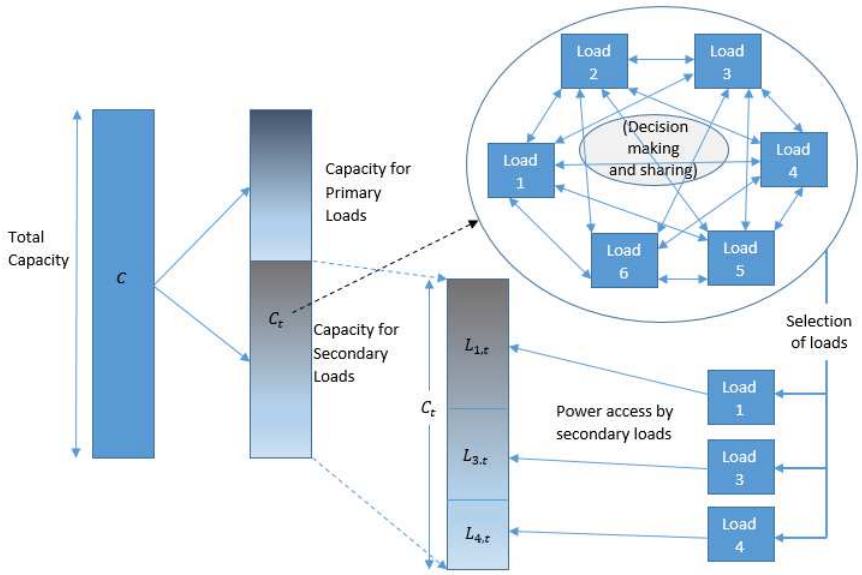


Figure 1.3: An Example of Decentralized Scheduling Process

receiving the capacity and also the user appliances for their load demands in each frame. A VLC will be available on each appliance to run one or multiple LAs. The VLC will take the loads and make decisions for them in a purely distributive manner. As the decision-making is iterative process for each time frame, the decisions are shared among the appliances after each iteration. The VLCs can perform those tasks easily, being self-sufficient device with certain level of computational and memory capacities (for taking load values and applying the scheduling algorithms) and smart communication features (for communicating with other appliances in a fast, secured and reliable way). To summarise, the tasks of receiving the capacity from SG operator and secondary loads from several appliances, performing computational works by executing predefined algorithms, calculating decisions for users and sending total scheduled load to operator etc. are performed by VLC.

The process of scheduling can be explained with the help of Fig. 1.3 in a simple way, where only a few loads from a section of SG are considered. The total capacity is divided into two parts for serving the primary and secondary loads. Capacity that is exactly equal to primary (important) loads is allocated for them. The remaining capacity for secondary (less important) loads is sent to VLCs which look after those particular appliances. The shiftable capacity will be distributed to different loads depending upon their

usage behaviour. Multiple loads could belong to single appliance as well, which is discussed later in section 3.1. When the total load (from shiftable loads Load 1, Load 2, ..., Load 6 considered in figure) from the appliances exceeds the capacity for shiftable loads, VLC implements one of the LA methods to select some loads so that their sum will be within the capacity. As in this example, loads 1, 3 and 4 are selected because the total load is equal (could be less than) to capacity C_t available.

Two targets are set for scheduling in the thesis. The first one, power optimization means all power available for shiftable loads is to be used without letting any wastage, which leads to profit to the operator. It is also advantageous to the customers as more demands can be served when maximum capacity is exploited. Similarly, the next target is to keep customers happy without receiving questions about the scheduling process. If some appliances are continuously rejected and others are selected more often, this may lead to dissatisfaction among the customers. Customers will change the power supplier if they are dissatisfied, as multiple operator distribution system is another feature of SG. Therefore, it is important to keep fairness in SG. Paper [18] takes the average task completion time between different loads that are to be scheduled by game theoritic methods as a measure of fairness. In that work, only the time taken for scheduling by scheduler and processing time by processor are taken into account. But in this thesis, the frequency of selection of appliances is taken as the measure of fairness. Secondary appliances are expected to be selected equal number of times if they send equal number of load requests.

1.4 Contribution of this work

The main contributions of this work are itemized in two aspects:

1. Application of LA based methods for scheduling the domestic loads for accuracy of optimization, and

2. Introduction of coin game along with the LA based methods to increase fairness of selection between appliances.

The main features of the scheduling process presented are application of LA in scheduling the shiftable appliances along multiple timeslots and consideration of fairness of system to bring balance in frequency of selection of loads. Every scheduling is performed along multiple time slots, which is one step forward compared to previous report [19] which considered the application of LA based method in a single time slot only. The LA methods

have been demonstrated to be able to approach the global optimal point with great precision. This means the selection of loads by LA method is good enough to optimize the electricity usage. A new method is introduced to improve fairness between participants of scheduling.

1.5 Organization of the Thesis

The thesis has been presented in seven chapters. Following the introduction in this chapter, a brief discussion on previous studies related to this thesis work is presented in Chapter 2. In Chapter 3, the problem to be solved is formulated. The following Chapter 4 presents the way (in the form of algorithms) of implementing the proposed LA based methods in the SG. Both LA methods, L_{R-I} and BLA have been dealt separately for building their own algorithms and understanding them more properly. The implementation of the algorithms done in MATLAB gave several numerical and graphical results, which are presented and discussed in Chapter 5. Finally, discussion of the methods based upon their results is made in Chapter 6 and the thesis concludes in Chapter 7.

Chapter 2

Related Work

Since this is an era of smart communication and sophisticated technology, every field of human interest is involved in radical changes to be smart and automatic. The electrical grid system is not an exception and many articles have been written and different methods have been proposed to make it smart. A short discussion of articles which are related to importance aspects included in this thesis work has been carried out in this chapter. Different methods of load categorization, distributed and centralized methods, LA based scheduling and fairness of scheduling methods are subjects of significance discussed in the following subsections.

2.1 Categorization of Loads/Appliances for Scheduling

Appliances have been differentiated as power-shiftable and time-shiftable appliances on the basis of their power flexibility and time flexibility in [20] for power optimization by taking advantages of these flexibilities. A much detailed classification of loads has been made in [21] where loads have been categorized on the basis of their physical properties, job types and their sizes. Paper [22] makes a separation between loads as

- Baseline loads - appliances that need to be served immediately upon request, e.g. light bulbs,
- Burst loads - appliances that can withstand delay at start but can not be interrupted until they finish, e.g. washing machine and

- Regular loads - appliances that need to be served continuously but can cope with short interruptions, e.g. refrigerator.

Similarly, three categories have been made by [23] on load types depending upon their power consumption profile namely non-shiftable, power shiftable and time-shiftable loads. The categorization in [22] and [23] is similar to the categorization in this thesis work but only two of those have been considered. Here, the flexibility of some appliances to adjust their power consumption is not considered. The appliances' time flexibility in terms of their tolerance to postponing their loads is considered. The appliances/loads of customers are categorized into two types only, **non-shiftable loads** that resemble the baseline loads whose operation time need to be exactly as specified by customer and **shiftable** resembling the burst loads who can suffer delay without much dismay and these loads are called primary and secondary loads respectively, as they can be compared to the primary and secondary users of cognitive radio networks where primary users are protected while the secondary users compete between each other to exercise the unused spectrum [24]. In similar fashion, in the SG system networks, there exist loads/appliances of greater importance and urgency, that will get service of their demands without any obstructions are the non-shiftable loads or demands or appliances. These loads and appliances are referred to as primary loads or primary appliances. While the other loads/appliances that are offered lower priority, are called shiftable loads or shiftable appliances and referred to as secondary loads or secondary appliances. Having said that, the latter kind of loads are our interest and only they will undergo scheduling process, where the selection is made based on remaining electricity budget after allocation of electricity to the former ones. The terms secondary load and shiftable load can be used interchangeably.

2.2 Learning Automata based Scheduling

As previously discussed in Chapter 1, LA has been implemented into scheduling process against secondary loads from secondary appliances. This is expected to optimize the total scheduled load against shiftable capacity with high accuracy. *"An LA is an adaptive decision-making device that learns the optimal action out of a set of actions through repeated interactions with a random environment"* [25]. The scheduling process which involves either selection or rejection of each and every load to generate a final optimal load that is below the power budget is necessary. The main characterstic feature of LA is that the actions sets have certain probability distribution and the

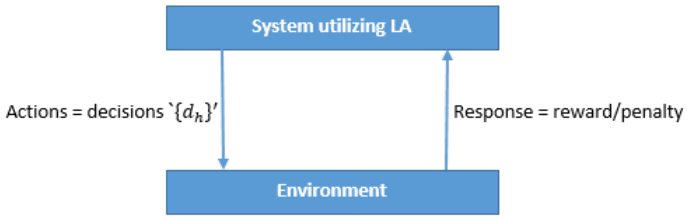


Figure 2.1: The basic schematic of LA

probability distributions are updated based upon feedback from the conditions/environment [25]. The idea implemented by LAs is shown in Fig. 2.1, where the system's job is done by VLC to generate new "improved" actions based on those responses from environment (constraints of optimization). The action set is selection or avoidance of each user, whose probabilities are 0.5 at the beginning, and the environment is the test of decision against the power budget. The environment produces the response through the test (comparison of total load generated against capacity after decision for each user is made), which again is an input to the action set for updating the probabilities. The actions will be updated and sent back to environment. So the decision-making is an iterative process of sending and receiving actions and response respectively to and from environment. As the name LA suggests, the LA methods will learn the best decision for each user that will balance the total load and capacity after some rounds of action-response phenomenon.

LA has a wide range of applications and the fields of its application have been shown in [26]. Even though LA based schemes have not been discussed extensively in SG field, [27] [28] [29] [30] utilize the LA based schemes to manage the demands in SG. The advantage of implementing LA in scheduling for DM is that it does not require a training interval and can be implemented simultaneously when the SG network is under performance but the disadvantage is that they incur higher expenses for computations [31].

Article [27] uses LA based scheduling at all hierarchical levels of SG for power management and also for preventing the unauthorized use of grid system. Four hierarchical levels include the main power stations, transmission units, distribution substations and smart meters at each household. That report does not discuss on the method of implementation of LA at different levels but jumps directly into the results of LA based demand management of SG. Article [28] formulates the decision-making problem to

make decisions for customers based on the load profiles and tariff (electricity) schemes. Instead of manual decision-making, the report proposes an automated decision-making process for each user by implementing two LA based methods, the ϵ-greedy algorithm and the pursuit algorithm. That is a completely different work than this thesis, that being based on the electricity pricing as a factor for time of scheduling. The loads come under different categories based on their time of occurrence, quite unusual and against the fact that most loads occur according to customers' needs and preferences. Article [29] also presents the idea of LA based communication scheme in SG. Routing algorithm is proposed for the efficient communication that contributes to a optimal delivery path. In that article, the decision-making is for determining optimal path. The constraints in the LA based communication model are cost, delay, transmission energy consumption and power usage. As the communication part has not been discussed in this thesis, so despite that report utilizing LA in SG, it's completely different. Article [30] utilizes the Q-learning algorithm to schedule the charging and discharging times of electric vehicles, which are large loads with shiftable features as well. That report also utilizes the hourly pricing as a factor to calculate maximum profit to the owners of vehicles.

BLA [32] [33] and L_{R-I} [34] have been chosen for the scheduling purpose in this thesis, and these methods fall under the dynamic non-cooperative game theories [35]. The dynamic nature of game theories are justified if at least one the participants can choose a strategy [36], and it is true for all participants in this thesis (every load is to be selected or rejected). They fall under the non-cooperative theory as the participants/players of the game (appliances in SG) need to select an action from decision set (between selection and rejection) to improve their own utility function whose value depends on other users' decisions also. In both methods, the final decision is reached after several iterations of independent decision-making between the customers, communicating with each other about one's choice and finally reaching a consensus between themselves (discussed in detail in upcoming chapters about the properties of LA based scheduling and the algorithm for implementation). L_{R-I} stands out from BLA in the sense that it involves a learning parameter whose value can be adjusted to control the decision-making process in terms of speed and accuracy. In contrast, BLA is self-sufficient, which does not involve such user-defined[1] parameters, so the accuracy and the speed of scheduling for decision-making can not be structured.

[1]Here, user is the scheduler.

2.3 Fairness

Fairness from the appliances' point of view is another subject of discussion in this thesis. There has not been much discussion in terms of fairness among users in SG [37]. *"Each user's throughput is at least as large as that of all other users which have the same bottleneck"* is the definition of fair flow control scheme given by Jaffe in [38] which is still valid in our case. The measurement of throughput is not our concern but the number of times any user is selected when they have nearly equal demand requests is the unit for measurement of fairness in this thesis (discussed in next chapter). A fair selection process is exhibited by scheduling process if all participants are happy with the results obtained from the competition. Since the appliances from different customers of SG are competing with each other for power usage, fairness becomes an issue. There can be different aspects in SG that can be considered to measure fairness between the customers, e.g. prioritization of customers, energy consumption, or using multiple objective fairness techniques [39].

Article [37] has studied fairness between users in terms of charging them according to their contribution in total cost of the system, depending on the power demand and the flexibility of those demands. A discussion is made on charging the users in a fair manner depending upon two conditions, first condition where users have equal flexibility but their loads vary and second condition where loads are equal but their flexibility is different. Article [39] applies scheduling in air conditioning devices which contribute to 22.3% [40] of residential power usage, and the proportional fairness among users is measured in terms of contency of users for receiving the power by proportional reduction in their demands by adjusting the thermostats in air conditioning appliances. Three different methods of obtaining fair results while scheduling flexible users (air conditioning devices) is presented in [41], namely Round Robin, Highest Power Next and Reciprocal Fair Management. Round Robin method considers fairness by selecting users in turns after queueing of loads to prevent repeated selection by means of switch off option available in flexible appliances (all users are selected at least once before some start to get selected twice). Highest Power Next is the method where the loads with high power are put first in the queue and the other smaller loads are queued behind (larger loads are switched off more often). The Reciprocal Fair Management is a constraint for scheduling which takes users' comfort and scheduling priorities.

In the following chapter, the optimization problem is modelled and its properties along with the fairness of scheduling methods are discussed.

Chapter 3

System Model and Problem Formulation

Lets take a region of SG system consisting of a VLC as scheduler with N secondary users (appliances which generate shiftable loads [15]) competing for power usage in SG. Appliances and users have been used interchangeably, which are the source of shiftable loads. The aim of this SG model is to select the loads, by a decentralized and fair method, that will match the capacity available. So, the users are required to make decisions for their loads within themselves (without any outside intervention), whether the demand is to be served or the demand will have to wait and compete again in upcoming time slots. Time is segmented into smaller frames or slots for scheduling called *timeslots*. Any load selected in a timeslot could remain passive or enter into scheduling process again as a new load as per the appliance's need in upcoming timeslots. The scheduling process needs to be carried out continuously for infinite time period. But due to computational capacity of SG (to avoid crowding of unserved loads) and ease of calculations, a round of scheduling lasts and repeats after a finite time period, e.g. a single day or a week, comprising of T timeslots. For ease of notation, i and t are the user and timeslot indices respectively, i.e., $i \in [1, N]$ and $t \in [1, T]$. We assume that any user will generate only one new load at a given timeslot. Note that scheduling refers to the load selection process at the beginning of each timeslot and a round of scheduling will end after T timeslots.

The most important and frequently used notations are specified below.

- $d_{i,t}(\tau)$: decision at timeslot τ for a load that emerges at time t from user i, where $\tau \geq t$.

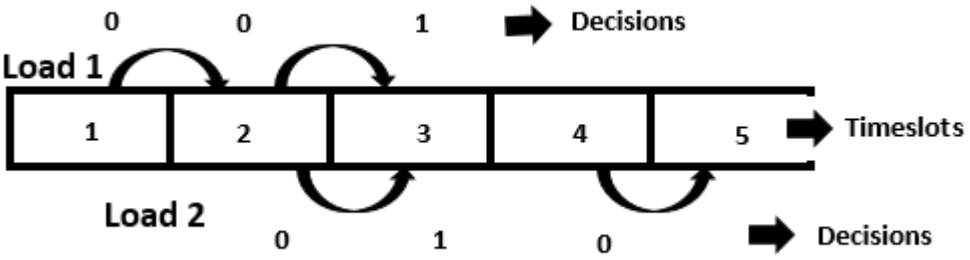


Figure 3.1: Explanation for timeslot notation

Table 3.1: Example for load and decision notations depending upon timeslot

	Timeslot (t) $\rightarrow$	1	2	3	4
Load 1 ($i = 1$)	Load	$L_{1,1}$	$L_{1,1}$	$L_{1,1}$	$L_{1,1}$
	Decision	$d_{1,1}(1) = 0$	$d_{1,1}(2) = 0$	$d_{1,1}(3) = 1$	$d_{1,1}(4) = 0$
Load 2 ($i = 2$)	Load	x	$L_{2,2}$	$L_{2,2}$	$L_{2,4}$
	Decision	x	$d_{2,2}(2) = 0$	$d_{2,2}(3) = 1$	$d_{2,4}(4) = 0$

- $L_{i,t}$: load emerging at time t, in user $i \in [1, N]$. The loads can be served in the same timeslot it appears or the succeeding timeslots. By default, every $L_{i,t}$ has its corresponding $d_{i,t}(\tau)$.
- C_t : capacity of SG for shiftable loads at timeslot t.
- $C = \sum_t C_t$: accumulated capacity.
- t_n : the current timeslot, i.e., timeslot now.

An explanation for better acquaintance to notations of loads and their decisions is shown in Fig. 3.1 and Table 3.1 by employing two loads - Load 1 and Load 2 from two appliances. Each appliance generates one load only at a given timeslot. It is to be noted that there are two notations for timeslot in a decision, e.g. if notation is $d_{i,t'}(t'')$, notation in subscript t' denotes the timeslot in which the corresponding load first appeared for scheduling whereas notation inside small braces t'' is for the indicated timeslot. In reference to Fig. 3.1, where Load 1 appears at timeslot 1 and exists upto timeslot 3 with decisions 0, 0 and 1 respectively ($L_{1,1}$ at timeslot 4 and afterwards will be ignored as it is already served in previous timeslot) while Load 2 appears at timeslot 2, gets served in timeslot 3 and a new load from user 2 appears again at timeslot 4. So the notations for the two loads and their contemporary decisions are as shown in table 3.1. A load will

not exist until it participates in scheduling process, e.g., Load 2, $L_{2,1}$=x at first timeslot and no decision is required to be made for it (same decision rule applies to the loads already served). Due to difference in the nature of appliances, the demand from any appliance may be changing at different timeslots, which calls for the scheduling process to update the load $L_{i,t}$ values against those changes. Only few secondary appliances like mobile phone chargers, TVs etc. are easier to deal with, as their demands remain constant throughout the usage period. So, different cases have been outlined as follows, depending upon the variation of loads from appliances and their effect on scheduling.

3.1 Classification of Appliances According to Selection of Load along Multiple Timeslots

The nature of secondary appliances differs from one another based on demands they generate across many timeslots. Appliances could produce equal or different demands at different timeslots. But due to scheduling, load demands may be postponed and multiple loads from the same appliance may exist in a single timeslot. Based on this situation where multiple loads from same appliance exist at a given timeslot, appliances can be classified into 4 types based on which loads are to be considered in scheduling process. Consider an appliance $i1$ generating two loads at timeslots $t1$ and $t2$ ($t1 < t2$). Assuming load at timeslot $t1$ is rejected and postponed to $t2$, two loads exist at timeslot $t2$. Four categories, on the basis of how the loads are treated by different appliances, have been made as follows:

Type 1 Consider a washing machine with loads $L_{i1,t1}$ and $L_{i1,t2}$ for washing and rinsing purposes respectively. The appliance has different demands at different times but they need to follow a sequence. In this case, $L_{i1,t2}$ needs to be considered only when $L_{i1,t1}$ is completed.

Type 2 E.g., heater is a load whose demand adjusts frequently with change in room temperature and even on mood of user. For these kinds, succeeding load request $L_{i1,t2}$ will be acknowledged while preceding request $L_{i1,t1}$ will be discarded.

Type 3 Certain users may have more than one load existing together, where it does not matter which load, $L_{i1,t1}$ or $L_{i1,t2}$, is scheduled first, but most importantly, both must be served. So either $d_{i1,t1}(t) = 1$ or $d_{i1,t2}(t) = 1$ or both could be served by SG simultaneously.

Type 4 Suppose two demands $L_{i1,t1}$ and $L_{i1,t2}$ exist, where it does not matter who is scheduled first but both loads can not be served simultaneously. In such cases, one of the loads, either $L_{i1,t1}$ or $L_{i1,t2}$ enters into scheduling process. It might be left to user's preference or importance on which one to allow to be involved in scheduling.

Since loads are specially referenced by their user index i and time index t, multiple demands from same appliance can be easily distinguished. So it is easy to address Types 1-3 where the loads are **postponed** (do not appear automatically), **updated** ($L_{i1,t1} = 0$) or in **coexistance** (both $d_{i1,t1}(t)$ and $d_{i1,t2}(t)$ can be considered) for scheduling. Type 4 has not been considered in this work. A further discussion will be carried out in the numerical results section for Type 4.

In the next section, we proceed to the mathematical formulation of scheduling process and its analysis.

3.2 The Optimization Problem

Optimization of selected demands is to be done so that the capacity available is fully utilized. Even when multiple loads from appliances of Type 1, 2 and 3 exist simultaneously, they can be optimized using same equation as there is no issue with which load to be involved in scheduling process. Equation 3.1 followed by three constraints as shown below is formulated, which targets on fulfilling the aim of utilizing the full capacity of SG by properly tuning the decision among the users along the time axis.

$$\max_{d_{i,t}(\tau)} \sum_{\tau=1}^{tn} \sum_{t=1}^{\tau} \sum_{i=1}^{N} L_{i,t} d_{i,t}(\tau), \tag{3.1}$$

$$\text{s.t. } d_{i,t}(\tau) \in \{0,1\}, \text{where } 1 \leq t \leq \tau \tag{3.2}$$

$$d_{i,t}(\tau) d_{i,t}(\tau') = 0, \text{ where } t \leq \tau' \leq tn, \ t \leq \tau \leq tn, \tag{3.3}$$

$$\forall \tau, \ \sum_{t=1}^{\tau} \sum_{i=1}^{N} L_{i,t} d_{i,t}(\tau) \leq C_{\tau}, \text{ where } 1 \leq t \leq \tau \leq tn. \tag{3.4}$$

The first two constraints for the optimization equation 3.1 oblige that load $L_{i,t}$ can be served only once. The appliances that have been served once, if request for power in forthcoming timeslots, should be treated as new loads, as shown in Fig. 3.2. The third constraint guarantees that at any timeslot τ, the old and new loads will not exceed the capacity available

for that timeslot. This aforementioned optimization problem is not practical for solving. The reason is that at the current timeslot, it is impossible to make any changes (tune) in previous decisions indicated by τ in $d_{i,t}(\tau)$, $\forall \tau < tn$.

3.2.1 Optimization in Practical Scenario

$$\max_{d_{i,t}(tn)} \sum_{t=1}^{tn} \sum_{i=1}^{N} L_{i,t} d_{i,t}(tn), \tag{3.5}$$

$$\text{s.t. } d_{i,t}(\tau) \in \{0,1\},\ t \leq \tau \leq tn, \tag{3.6}$$

$$d_{i,t}(\tau)\ d_{i,t}(\tau') = 0,\ t \leq \tau' < \tau \leq tn, \tag{3.7}$$

$$\sum_{t=1}^{tn} \sum_{i=1}^{N} L_{i,t}\ d_{i,t}(tn) \leq C_{tn}. \tag{3.8}$$

The second condition, equation 3.7, means if there is any $d_{i,t}(\tau') = 1$, then $d_{i,t}(\tau) = 0$ for $\tau > \tau'$. Again, third condition (equation 3.8) means that we consider the current timeslot tn, whose capacity is C_{tn}. As opposed to equation 3.1, this practical optimization equation ignores optimization for timeslots preceding current timeslot because we can do nothing for previous timeslots, where decisions have already been made. The optimization by tuning the decisions is only possible for the current timeslot where the loads are known or upcoming timeslots if loads values can be obtained beforehand. To remind, it is assumed that there is not reliable hint from historical information. So no prediction or estimate regarding demands can be provided to system for future calculations. Therefore, we consider the decision at the current timeslot only, i.e., tn.

We know at current timeslot tn, the notations for

- new demand = $\{L_{i,tn}\},\ \forall i$,
- shiftable capacity = C_{tn} and
- unserved loads in previous slots = $\left\{L_{i,t'} | d_{i,t'}(t'') = 0,\ \forall t' < t'' < tn\right\}$.

Expanding the equation 3.5 w.r.t. the new demands and unserved old demands that accumulate for new scheduling,

$$\max_{\{d_{i,tn}(tn), d_{i,t'}(tn)\}} \sum_{i=1}^{N} L_{i,tn}\ d_{i,tn}(tn) + \sum_{i=1}^{N} \sum_{t'=1}^{tn-1} L_{i,t'}\ d_{i,t'}(tn). \tag{3.9}$$

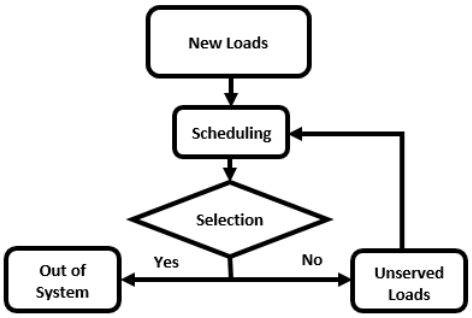


Figure 3.2: Scheduling of new loads and old loads

In equation 3.9, t' denotes the previous timeslot and tn is the current timeslot. New loads will appear in left part of equation and the loads on right hand side are the loads from previous timeslot still unserved, competing for scheduling. Note that for any load $L_{i,t'}$ that has been served any time before tn, $d_{i,t'}(tn)$ is zero due to conditions 3.6 and 3.7 of optimization equation. Thus, only the previous unserved loads are in fact are accumulated and represented by the right side of equation 3.9 in this optimization problem. Fig. 3.2 clearly shows the above explanation where old loads are actually the unserved loads that pass into new timeslot for scheduling with notation t' and loads that have been served will be out of the system. It is to be reminded yet again that the loads from same appliance at different timeslots are considered two different loads.

3.2.2 Comparison to Subset-sum Problem

This problem of optimization can be compared with 0-1 Knapsack problem [42] (maximize $\sum_{i=1}^{N} v_i\ x_i$, s.t. $\sum_{i=1}^{N} w_i\ x_i \leq W$ and $x_i \in \{0,1\}$). With same values for v_i and w_i, it becomes a subset-sum problem [43]. In the subset sum problem, a new subset B (or many subsets if possible) is to be formed from a given integer set A whose elements will produce a given sum X. If $A = \{x_j : 1 < j > z, x_j \in \mathbb{Z}^+\}$, then $B \subseteq A$ such that $\sum_{j=1}^{z} x_j y_j = X,\ \forall i, y_j = \{0,1\}$ where X is the targeted sum that will be defined in the problem already. The load demands $L_{i,t}$ in the SG are comparable to the integer set A from which a demand subset is to be calculated whose sum is equal to the capacity C_t of SG for that timeslot, i.e., $L_{i,t} \rightarrow A$, $d_{i,t} \rightarrow y_i, L_{i,t}\ d_{i,t} \rightarrow B$ and $C_t \rightarrow X$. Satisfying the conditions of a subset sum problem, it becomes a NP hard problem [44]. So, the problem is a challenge

to the SG, to find a solution for the subset sum problem by appropriate selection of loads that provide optimal power usage (as there exist solutions with certain conditions to subset sum problem, e.g. lattice basis reduction [45] [46]). Moreover, the domestic grid system consists of a large number of participants in the scheduling process. The number of users continuously grows higher as the number of timeslots increases with addition of rejected loads from previous timeslots into the current timeslot. Since the problem stated above is NP hard, the application of game theoretic programming algorithm along multiple timeslots will be an issue. Hence the problem will be decomposed by treating a finite number of timeslots T (in a single day), instead of infinite time period. Furthermore, the whole SG system will be split into smaller areas/regions, so that smaller number of appliances N are considered in scheduling each region separately. Reduction in the number of participants is a solution to subset-sum and knapsack problems [47].

3.3 Fairness among Users/Appliances

The fairness of selection between different appliances can be partially addressed by application of game theoretic model as a selection process, as it does not prioritize any user during a timeslot. The issue of fairness can be said to be non-existent in infinite time period as the loads have equal chances of selection in each timeslot (under fair random selection method where no users get any priority). Still some users may be repeatedly selected while others may get rejected concurrently, within a finite time period. The number of times appliances are selected also depends upon their nature, like how often they are used (number of demand requests from each appliance). Such imbalance of repetition and exclusion of users in the scheduling process can be questioned and considered biased from a SG customer's point of view and user dissatisfaction is a huge concern for SG operators. Users do not want to wait longer periods, so the users should be selected rotatively for serving their demands. This issue of fairness has to be addressed and it is represented by symbol $\mathfrak{F}$ in the thesis and obtained from the ρ_i values. $\rho_i(tn)$ is the number of timeslots each appliance i has been selected w.r.t. number of timeslots it sent request until current timeslot tn and expressed in mathematical form as follows.

$$\rho_i(tn) = \frac{P_i(tn)}{Q_i(tn)}, \tag{3.10}$$

where $P_i(tn)$ and $Q_i(tn)$ are number of times (timeslots) any user i is

selected and number of times the appliance sent request for scheduling until current timeslot tn. Equation 3.10 can be enunciated (indicated) as the happiness of any user i upon getting scheduled sooner and/or more often. For any user i at tn, for a given value of Q, $P \in [0, Q]$ means $\rho \in [0, 1]$. E.g., if an appliance i's demand is rejected at first, second and third timeslot and finally selected at forth timeslot, its $\rho_i = \frac{P_i=1}{Q_i=4}$=0.25 until forth timeslot and the values are updated as time proceeds. It is straightforward that $\rho = 1$ means a user (load) will be extremely satisfied for having scheduled without delay (P and Q values match). $P = 0$ leads to $\rho = 0$ meaning frustration and full dissatisfaction due to repeated rejection. In reference to definition of fairness, this thesis considers the power of appliances, in terms of speed or delay, while utilizing the resource of the system. $\rho = 1$ means no delay and the delay increases with its lower values. So, this can also be considered as fairness w.r.t. waiting time (delay).

The scheduling process will be declared fair enough if all users receive equal or considerably equal proportion of time delay against number of demands. Considering the speed of selection of users as a factor for the measurement of fairness, the following equation can be formulated for collective/combined fairness of SG system as a whole, similar to [48].

$$\mathfrak{F} = \frac{|\sum_{i=1}^{N} \rho_{i,t}|^2}{N \sum_{i=1}^{N} \rho_{i,t}^2} \geq \epsilon, \text{where } \epsilon \in [0, 1]. \tag{3.11}$$

In above equation 3.11, the value of $\mathfrak{F}$ determines the fairness of the SG model, in the grade between 0 to 1. It assesses the fairness of the scheduling process of users upon matching or at least being close enough to each other's $\rho_i(tn)$ values. The closer the value of fairness index is to 1, the more fair is the scheduling. Full fairness ($\mathfrak{F} = 1$) is attained when the value of ρ_i, is same forall i. $\mathfrak{F} = 0$ means all appliances have not been selected even once (for some i, $P_i = 0$ exists). But the value of Q in equation 3.10 is different for different users, which mostly leads to different P values as well. As it is difficult to get same ρ_i for all users, there should be a certain limit set to value of ϵ which will be the minimal fairness of SG. And $\mathfrak{F}$ should be equal to or above ϵ value. Because it is uncertain that 100% fairness is possible, this minimal value of $\mathfrak{F} = \epsilon$ needs to be attained by scheduling, for all users to be satisfied.

The problem of utilizing the maximum capacity of SG along the time axis can be modified by adding fairness as a constraint and represented by

$$\max_{d_{i,t}(tn)} \sum_{i=1}^{N} \sum_{t=1}^{tn} L_{i,t}\ d_{i,t}(tn), \tag{3.12}$$

$$\text{s.t.} \sum_{i=1}^{N} \sum_{t=1}^{tn} L_{i,t}\ d_{i,t}(tn) \leq C_{tn}, \tag{3.13}$$

$$d_{i,t}(\tau) \in \{0,1\},\ t \leq \tau \leq tn, \tag{3.14}$$

$$d_{i,t}(\tau)\ d_{i,t}(\tau') = 0,\ t \leq \tau' < \tau \leq tn, \tag{3.15}$$

$$\frac{|\sum_{i=1}^{N} \rho_{i,tn}|^2}{N \sum_{i=1}^{N} \rho_{i,tn}^2} \geq \epsilon. \tag{3.16}$$

The first three constraints are the repetition of optimization equation to maximally exploit the capacity left for secondary users as in equation 3.5. Last constraint adds up the fairness issue into optimization process, allowing all users to be selected proportionately. Due to addition of fairness constraint, it may become infeasible to solve the optimization problem for a given ϵ. A perfectly fair solution that satisfies $\forall i, \sum_{tn} \rho_{i,tn} = \sum_{tn} \rho_{i',tn}, i \neq i'$ may not exist in every timeslot, surely for smaller tn values. In the next chapter, a comprehensive discussion on the solutions to the optimization problem is given and the issue of fairness of SG system has been elaborated.

Chapter 4

Implementation of BLA and L_{R-I} in Demand Scheduling

As the optimization problem is shown as NP hard in previous chapter, we consider not so large number of appliances 'N' participating in scheduling [49]. This can be done by segregating the users into different geographical regions as shown in Fig. 4.1, and considering the customers who have smaller appliances only[1].

Several steps are involved in the demand scheduling processes. Foremost, the length of timeslots is to be predefined, so that users know how long they can get served after they are chosen by scheduling scheme. For

[1]SG users like factories with big machineries can be treated differently, without considering them in the general scheduling process. One of the many solutions could be treating them as primary users/appliances.

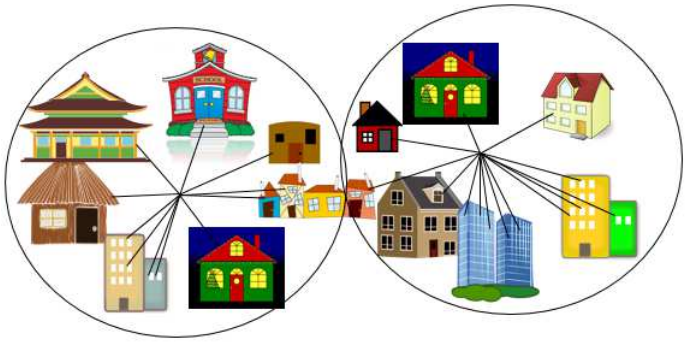


Figure 4.1: Grouping of SG users based on location

every timeslot, the SG should announce the capacity left for the secondary loads, while appliances announce their demands simultaneously[2]. Not all but active appliances [50] (that need to operate) only will need power for a given timeslot tn. Active loads are the set of loads that want to be scheduled in the current timeslot tn, which includes the loads $L_{i,tn}$ that appear in the current timeslot along with the loads $L_{i,t},\ t < tn$ that appeared and remained unserved until preceding timeslot. For ease of notation and understanding, active loads at timeslot tn will be denoted by $l_h(tn)$ and corresponding decision $d_h(tn)$, ignoring the notation for timeslot when it first appeared (VLC should be able to address type 3 and 4 in section 3.1 where two or more loads of same appliance from different timeslots exist simultaneously). Here h is the index of all loads that are to be served at timeslot tn, and H denotes the total number of loads to be served at tn, i.e., $h \in [1, H]$. Before the implementation of any scheduling process, firstly it will be checked at each timeslot if all loads can be accommodated by the capacity ($C_t \geq \sum_{h=1}^{H} l_h(tn)$). If yes, then all demands will be served at once, skipping the implementation of any algorithm for load selection. Another case where each load is larger than capacity ($\forall i, l_h(tn) > C_{tn}$) also permits the scheduling to be skipped for that timeslot as solution does not exist clearly[3]. Otherwise the users need to undergo scheduling by algorithms based on LA, which implement an process iterative (within each timeslot) of decision-making, reward-penalty allocation based on those decisions and again new decisions based on reward/penalty received, as shown already in Fig. 2.1, and this process will repeat in every timeslot.

4.1 Formulating the Decision-making Process as a Game

The distributed decision-making problem can be formulated as a game[4] denoted by $\mathfrak{G} = [\mathbf{G}, \{d_h(tn)\}_{h\in H}, \{U_h(tn)\}_{h\in H}]$, whose utility function $\{U_h(tn)\}_{h\in H}$ for any load h, at a given timeslot tn is formulated as

[2]Those appliances whose demand values are not received are treated as passive appliances in this slot.

[3]SG should be able to predict such situation and increase its capacity large enough to accommodate some of the loads

[4]The task of formulating the distributed decision-making process as a potential game has already been done in previous work and follows the same concept.

$$U_h(tn)(d_h(tn), \mathbf{d}_{-h}(tn))$$
$$= \begin{cases} \frac{1}{l_h(tn)d_h(tn)+\sum_{j\in H\setminus h} l_j(tn)d_j(tn)+C_{tn}}, & C_{tn} < l_h(tn)d_h(tn) + \sum_{j\in H\setminus h} l_j(tn)d_j(tn) \\ \frac{1}{C_{tn}-l_h(tn)d_h(tn)-\sum_{j\in H\setminus h} l_j(tn)d_j(tn)+\varepsilon}, & C_{tn} \geq l_h(tn)d_h(tn) + \sum_{j\in H\setminus h} l_j(tn)d_j(tn), \end{cases} \tag{4.1}$$

where $\mathbf{d}_{-h}(tn)$ denotes the set of decisions taken by loads other than h. Notations in Equation 4.1 are explained as follows:

- $\mathbf{G} \in \{1, 2, 3, ..., H\}$ are the set of shiftable loads from appliances with specific load indexed by h.
- $\{d_h(tn)\}$ is the set of decision actions taken by the loads, i.e., $D_{tn} = \{d_1(tn), d_2(tn), ..., d_H(tn)\}$, where $d_h(tn) \in D_{tn}$ is the decision/action of h at tn.
- $\{U_h(tn)\}_{h\in H}$ is the utility function of h expressed in terms of C_{tn}.

In $\mathfrak{G}$, the players are the set of appliances with active load $\{l_h(tn)\}$ taking part in scheduling and their strategies are their decisions $\{d_h(tn)\}$. The difference of scheduling demands in this thesis compared to previous work [19] is that this work considers the loads from previous timeslots in the current timeslot as well. So, considering each timeslot at once, the properties of game $\mathfrak{G}$ from previous work still come into effect within same explanations. The utility function of any user from the system's perspective is expressed in terms of C_{tn}. The optimization problem becomes a game as the utility function can be expressed as potential function of a potential game. The justification for the potential game is simple : the payoff due to change in strategy of any player should be expressed in a single global function, i.e., potential function. In $\mathfrak{G}$, the payoff is given by the utility function, so utility fucntion is actually the potential function. Game $\mathfrak{G}$ is actually an exact potential game as the change in strategy (decision) of any user leads to exactly equal change in the utility or the potential function. The solution point attained by the decisions $d_1^*(tn), d_2^*(tn), \ldots, d_H^*(tn)$ for H loads is the NE point of $\mathfrak{G}$ if no player can improve its utility function by deviating from it unilaterally. Formally, the NE point satisfies $\big(d_h^*(tn), \mathbf{d}_{-h}^*(tn)\big)$ as $U_h(tn)\big(d_h^*(tn), \mathbf{d}_{-h}^*(tn)\big) \geq U_h(tn)\big(d_h(tn), \mathbf{d}_{-h}^*(tn)\big)$, $\forall h \in \mathbf{G}$ and $\forall d_h(tn) \in D_{tn}/\{d_h^*(tn)\}$[5]. Note that ε in Equation 4.1 is a small positive number to prevent $U_h(d_h, \mathbf{d}_{-h})$ from infinity when $C_{tn} = \sum_h l_h(tn)\ d_h(tn)$ holds.

[5] $d_h^*(tn)$ denotes the change in decision of player h from previous decision d_h ($1 \to 0$ or $0 \to 1$).

It can be easily proved that the global optimal point of $\mathfrak{G}$ is actually the NE point for each timeslot. It is not possible to increase the global optimal point by changing the decision of any user. The term global optimal point means there is not room for new loads to be activated into scheduling. So change in decision of any user from 0 to 1 will cause the total demand of selected users (having decisions 1) to exceed the capacity C_{tn}. Inversely, change in decision of any user from 1 to 0 will lead to decrease in the utility function. Thus, this situation follows that global optimal point is actually the NE point of the game and vice versa.

As already stated, BLA [32] [33] and L_{R-I} [34] methods are chosen for the task of decentralized scheduling. Both methods adhere to cyclic methods of decision-making which leads to reward or penalty which again helps in better decision-making and ultimately a good decision is attained that satisfies the optimization equation and the restrictions (constraints). The number of iterations taken to reach the final set of decisions $(d_1(tn), d_2(tn), \ldots, d_H(tn))$ can not be controlled in BLA but it is possible to do so in L_{R-I} by tuning a parameter λ, known as learning parameter. This is the main distinctness between the selected scheduling methods. Value of λ affects the number of iterations and the accuracy of convergence also. A trade-off exists between smaller number of iterations and better accuracy (convergence closer to C_{tn}) due to converged decisions (D_{tn}) depending on the value of λ, and it is controlled by the computational capacity of VLC (good communication channel with good configuration of processor and memory allow large number of iterations and thus better accuracy, which is achieved by lower values of λ)

Each iteration (round) involves decision-making which will yield either a reward or penalty depending on being able to optimize the power capacity or not. A reward is given to users if, in every round of decision-making, the value of $\sum_{h=1}^{H} d_h(tn)\, l_h(tn)$ is greater than or equal to previous value without exceeding C_{tn}, so that it will finally approach to C_{tn}. Decision yielding a load value lower than the previous one or exceeding C_{tn} will be penalized. The decision-making processes and reward and penalty calculations necessary in implementing both BLA and L_{R-I} are explained in steps by formulation of algorithms, in the following subsections.

4.1.1 Calculation of Reward and Penalty

Since several iterations of decision-making is required to reach a final decision within a given timeslot, let each iteration be indexed by s, so the total of active loads $\sum_{h=1}^{H} l_h(tn)$ due to decisions $d_h(tn)$ at any iteration s is denoted

by $L_F(s)$. Loads and decisions are denoted by $l_h(tn)$ and $d_h(tn, s)$. In order to reach the best possible value of scheduled demand, it is beneficial if the value of $\sum_h l_h(tn) = L_F(s)$ approaches C_{tn} but does not exceed C_{tn} as the iteration continues. The beneficial situation leads to a reward to the players. Otherwise, a penalty is applied, i.e., when $L_F(s)$ is either greater than C_{tn} or less than its previous value, $L_F(s-1)$. The procedure for deciding reward or penalty is outlined as an algorithm in Algorithm 1.

Algorithm 1 Assignment of Reward/Penalty

Input:

- The active loads $\{l_h(tn)\}$, $\{d_h(tn, s)\}$ and capacity C_{tn}.

Output:

- Reward or a penalty for each load.

Begin
  $\forall h$, $d_h(tn, s)$ should be known.
  Calculate $L_F(s) = \sum_{h=1}^{H} l_h(tn)\ d_h(tn, s)$.
  if $L_F(s) \leq C_{tn}$ and $L_F(s) \geq L_F(s-1)$, **then**
    $d_h(tn, s)$ leads to a reward to load h.
  else
    $d_h(tn, s)$ leads to a penalty to load h.
  end if
End

At each timeslot, the reward-penalty calculation continues with increment of s by 1 starting from $s = 1$ until the decision for each load has converged for that timeslot (given as Stopping Criteria in subsection 4.1.2).

4.1.2 Decision-Making

LA based methods employ decision-making with reward penalty schemes which are complementary and act as feedback to each other. New decisions for each user can be taken after knowing how good the previous decisions were, good decision set will yield reward while bad decision set will obtain penalty. This work considers two well-recognized LA based decision-making methods so that comparisons can be made in-between before implementation in SG. The selected popular LA methods are BLA and L_{R-I}, whose decision-making processes are probabilistic where a reward due to old decision increases the probability of selecting the same decision for most loads in the current iteration while penalty means higher chances of changing the

decisions. The description for the LA methods are presented in algorithm forms in following successive sub-subsections.

BLA based Decision-Making Process

Two hyper-parameters $a_{h,j}$ and $b_{h,j}$ are considered to count the number of rewards and penalties respectively, where $j \in 0, 1$ denotes the decisions. So every load has four hyper-parameters $a_{h,0}$, $b_{h,0}$, $a_{h,1}$ and $b_{h,1}$. For each iteration, increment in the value of $a_{h,0}$ or $a_{h,1}$ if the decision (0 or 1 respectively) leads to a reward and $b_{h,0}$ or $b_{h,1}$ if the decision (0 or 1 respectively) leads to a penalty. This can be explained by the Table 4.1.

Table 4.1: Effect of reward and penalty on decision parameters for load h at iteration s.

	$d_h(tn, s)$=1	$d_h(tn, s)$=0
Reward	$a_{h,1} = a_{h,1} + 1$	$a_{h,0} = a_{h,0} + 1$
Penalty	$b_{h,1} = b_{h,1} + 1$	$b_{h,0} = b_{h,0} + 1$

- For decision = 1, $a_{h,1}$ will be increased, if it leads to a reward. Similarly $b_{h,1}$ will be increased if it leads to a penalty.
- For decision = 0, $a_{h,0}$ will be increased if it leads to a reward, and $b_{h,0}$ will be increased if it leads to a penalty.

The algorithm for BLA for decision-making in detailed form is presented in algorithm 2 on page 30. The algorithm is carried out simultaneously for each load.

L_{R-I} based Decision-Making Process

The most peculiar feature of L_{R-I} scheme, in contrast to BLA, is involvement of a learning parameter (denoted by λ) whose value affects the convergence speed and accuracy (proximity to optimal point). Similar to variables $a_{h,j}$ and $b_{h,j}$ in BLA, L_{R-I} consists of two parameters $p_{h,0}(s)$ and $p_{h,1}(s)$ representing the probability of selecting decisions 0 and 1 by h^{th} load in the next $(s + 1)$ round of iteration respectively, where $p_{h,0}(s) + p_{h,1}(s) = 1$. For example, if $p_{1,0}(20) = 0.3$ and $p_{1,1}(20) = 0.7$, then appliance 1 will make a decision $d_1(tn, 21) = 1$ in iteration 21 with 70% probability and decision 0 with 30% probability. Initially, the probability of selecting decision 1 or 0 is set equal, i.e., $p_{h,0} = p_{h,1} = 0.5$ holds and their values will be updated, based on reward or penalty received in every iteration within a timeslot.

Algorithm 2 Calculating Decision by BLA

Input:

- $l_h(tn)$, C_{tn}, T, M and Z.
- Declare variables $x_{h,0}$, $x_{h,1}$, $a_{h,0}$, $b_{h,0}$, $a_{h,1}$, and $b_{h,1}$ for each h.

Output:

- Decision for each i, $d_h(tn)$.

Begin
  Initialize tn. ▷ From $tn = 1 \rightarrow T$
  Initialize s=1 and $a_{h,0}(s) = b_{h,0}(s) = a_{h,1}(s) = b_{h,1}(s) = 1$, where s is the index of iterations.
  for each load h **do**
    Draw two values $x_{h,0}$ and $x_{h,1}$ from the Beta distribution functions given by $\beta(a_{h,0}(s), b_{h,0}(s))$ and $\beta(a_{h,1}(s), b_{h,1}(s))$ respectively.

$$\beta(a_{h,j}(s), b_{h,j}(s)) = \frac{\int_0^{x_{h,j}} v^{(a_{h,j}(s)-1)}(1-v)^{(b_{h,j}(s)-1)}dv}{\int_0^1 u^{(a_{h,j}(s)-1)}(1-u)^{(b_{h,j}(s)-1)}du}. \tag{4.2}$$

    Compare $x_{h,0}$ and $x_{h,1}$.
    if $x_{h,0} < x_{h,1}$ **then**
      Decision $d_h(tn, s)$=1.
    else
      Decision $d_h(tn, s)$=0.
    end if
  end for
  Calculate whether reward/penalty is received as stated in algorithm 1 due to decision set $\{d_h(tn, s)\}$.
  $s = s + 1$.
  for each user i **do**
    if reward in step 13 **then**
      if $d_h(tn, s)$=1 **then**
        $a_{h,1}(s) = a_{h,1}(s) + 1$.
      else ▷ $d_h(tn, s)$=0
        $a_{h,0}(s) = a_{h,0}(s)$+1.
      end if
    else ▷ Penalty in step 5
      if $d_h(tn, s)$=1 in step 5 **then**
        $b_{h,1}(s) = b_{h,1}(s)$+1.

Algorithm 2 Calculating Decision by BLA (continued)

```
            else                                   ▷ d_h(tn, s)=0 in step 5
                b_{h,0}(s) = b_{h,0}(s)+1.
            end if
        end if
    end for
    Check for stopping criteria given in sub-subsection 4.1.2.
    if Stopping criteria satisfied then
        Stop.
    else
        Go to Step 4.
    end if
End
```

The decision-making process for L_{R-I} takes place as outlined in algorithm 3 in page 32.

Stopping Criteria for Algorithms

The LA based scheduling methods run through a number of iterations in each timeslot for calculating the preferred decision, that exploits the optimal capacity. But the iterations can not run all along, consuming a lot of time. So we set a limit M for maximum number of iterations that each round of decision-making can proceed. Also, the preferred decision set may be reached within fewer iterations, where the decisions will keep repeating as no upgrade will be possible. These two are set as criteria for stopping a decision-making algorithm.

- The number of iterations has reached maximum, i.e., $s = M$.
- Decisions leading to rewards remain unchanged for certain iterations Z, i.e., if $d_h(tn, s)$=$d_h(tn, s+1)$=.....=$d_h(tn, s+Z)$ for any value of s.

To summarize the two algorithms in previous sub-subsections, an overall algorithm is illustrated on page 34, by taking BLA as an example for distributed scheduling by LA. It is to keep in mind that, even before first step of the algorithm is started, a pre-check is made at each timeslot, if scheduling can be avoided (enough capacity available or each load is larger than capacity). If so, all or none of the users are allowed to turn on their loads.

Algorithm 3 Decision-making by L_{R-I}

Input:

- $l_h(tn)$, C_{tn} for each timeslot tn, T, M, Z.
- Declare variables $p_{h,0}$, $p_{h,1}$ and Ran_h, where $p_{h,0}$ is action probability of choosing decision 0 by player h, similarly $p_{h,1}$ is probabilility of choosing 1 and Ran_h is a randomly generated numbers betweem 0 and 1.

Output:

- Final decision for each user $d_h(tn)$ for each timeslot $tn = [1, T]$

Begin
  Initialize tn and choose a value for tuning parameter λ.
  Initialize $s = 1$, $p_{h,0} = p_{h,1} = 0.5$.
  for every h **do**
    Generate $Ran_h(s)$.
    if $p_{h,0}(s) \leq Ran_h(s)$ **then**
      $d_h(tn, s) = 0$.
    else
      $d_h(tn, s) = 1$.
    end if
  end for
  Calculate reward or penalty received due to $d_h(tn, s)$ following steps in sub-subsection 4.1.1.
  for each load h **do**
    if reward in step 12 **then** ▷ No update for penalty
      if $d_h(tn, s) = 0$ **then**
        $p_{h,1} = (1 - \lambda)p_{h,1}$.
        $p_{h,0} = 1 - p_{h,1}$.
      else ▷ $d_h(tn, s) = 1$
        $p_{h,0} = (1 - \lambda)p_{h,0}$.
        $p_{h,1} = 1 - p_{h,0}$.
      end if
    end if
  end for

Algorithm 3 Decision-making by L_{R-I} (continued)

Check for stopping criteria given in sub-subsection 4.1.2.
if Stopping criteria satisfied **then**
    Turn on the selected loads.
    Stop the iterations for this tn.
else
    $s = s + 1$.
    Go to Step 4.
end if
End

4.2 Coin Game Implementation for Increasing Fairness of SG

With the help of algorithms in previous section, the decisions for each user can be calculated at every timeslot. Users are selected in some timeslots while they will be passive or rejected in others. Measurement of fairness $\mathfrak{F}$ by comparing the number of times each appliance gets selected (P_i) is to be observed to see the difference in results. So, the need for a solution to address the problem was felt and a solution has been proposed in this section.

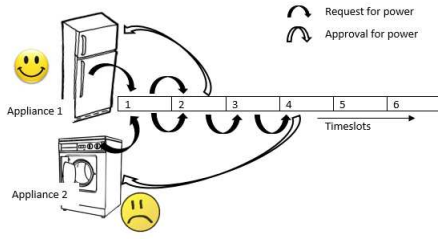


Figure 4.2: Issue of Fairness with random Scheduling

Fairness of SG system is considered in terms of frequency of appliances to access the power, i.e., number of times (timeslots) an appliance sends demands (fresh or rejected or both) versus number of timeslots the demands stay in scheduling process, '$\mathfrak{F}$'. Specifically, it is also a measure of waiting time for appliances, a higher value of ρ_i for appliance i suggests that loads from appliance i did not have to wait for long time in average. A system is declared fair enough if appliances enjoy close values between their individual ratio of number of timeslots to others. In other words, the happiness of a

Algorithm 4 Complete scheduling by LA (example by BLA)

Input:

- The active loads $l_h(tn)$ and capacity C_{tn} for each timeslot tn, M, Z

Output:

- The optimized decision for each user and total load served

Begin
  Initialize s=1, $a_{h,j} = b_{h,j} = 1$, $L_F(s)$=0, $Rep = 0$.
  for each load h **do**
    Draw two random values of $x_{h,j}$, i.e., x_{h0} and $x_{h,1}$ from the Beta distribution function given by $\beta(a_{h,j}, b_{h,j})$ in equation (4.2).
    if $x_{h,0} < x_{h,1}$ **then**
      $d_h(tn, s)$=1.
    else
      $d_h(tn, s)$=0.
    end if
  end for
  Calculate $L_F(s) = \sum_{h=1}^{H} l_h(tn) d_h(tn, s)$.
  if $L_F(s) \leq C_{tn}$ and $L_F(s) \geq L_F(s-1)$ **then**
    a reward.
  else
    a penalty.
  end if
  for each h **do**
    if reward was received **then**
      if $d_h(tn, s)$=1 **then**
        $a_{h,1} = a_{h,1}$+1.
      else
        $b_{h,1} = b_{h,1}$+1.
      end if
    else ▷ penalty was received
      if $d_h(tn, s)$=0 **then**
        $a_{h,0} = a_{h,0}$+1.
      else
        $b_{h,0} = b_{h,0}$+1.
      end if
    end if
  end for

Algorithm 4 Complete scheduling process by LA (example by BLA) (continued...)

32: **if** $\forall h,\ d_h(tn,s)=d_h(tn,s-1)$ **then**
33: $Rep = Rep + 1$. ▷ Rep = counter for successive decision repeats
34: **else**
35: $Rep = 0$.
36: **end if**
37: **if** $s = M$ or $Rep = Z$ **then**
38: Stop for this timeslot, users should acknowledge the results and selected loads are powered on.
39: **else**
40: Go to step 3.
41: **end if**
42: **End**

user depends on values of ρ_i and the fairness of system is measured in terms of $\mathfrak{F}$. As it is easier and reasonable to consider fairness from appliances' point of view, ρ_i is taken instead of ρ_h. The SG operator should try to maintain balanced ρ_i values to keep customers happy. Application of LA itself is anticipated to resolve $\mathfrak{F}$ values in long run of t due to randomness of selection but quicker solution is the target for customer satisfaction.

Adding the fairness constraint directly into LA method as an additional constraint is intricate to implement. So, a separate heuristic based virtual coin game (VCG) scenario has been proposed which can be combined to both LAs, and expected to increase $\mathfrak{F}$ value of SG compared with the scheduling solely by LA games only. As VCG is a supplement to the LA based scheduling process, VCG should act as a catalyst to improve fairness and mitigate the unbalanced situation where some users get scheduled quite often while others need to contend time and again.

The name VCG is chosen as it utlizes the concept of coins in a way similar to virtual games played on mobile phones or computers, where players are awarded coins upon success, e.g. completion of certain tasks, and coins give special benefits to players to boost their game. Opposite to those games where coins are increased for completing certain tasks, the players in VCG will lose coins upon success and will gain coins upon their failure during scheduling process. Similar to VCG in this thesis, [51] presents a game named 'Treasure' which employs coins into the game to measure wireless network strength. In VCG, the players use the electricity at the expense of coins (similar to Super Mario video game where 100 coins are deducted

to give an extra life [52]). The users in VCG will already have coins which are allocated to each appliance before start of simulation at timeslot 1 and act as a ticket to enter into scheduling. The coins that each appliance gets at commencement of scheduling process can be based on equality (same for all) or proportion (according to loads each requires). The stock of coins with a player varies as the game proceeds. A minimum reserve of coins is necessary depending upon the demand value and timeslot (time of day). Failing to maintain such a reserve transfers the active load to next timeslot directly without even entering into scheduling game. Having more coins helps appliances, as appliances can take part in scheduling in more timeslots.

With combination of VCG into LA based scheduling, the users are benefited by double gaming (LA + VCG) with win-win situation, either they win in scheduling process to enjoy the power demand or else they will be rewarded with coins. The coins act like a ticket to enter into the scheduling process. If users are continuously rejected, still collection of more coins will keep them much longer (in terms of timeslots) in scheduling. The repeatedly selected users will keep losing their stock of coins and need to wait until they have collected enough coins to pay for their loads.

The coin game takes place as shown in Fig. 4.3. For every timeslot tn, values for capacity C_{tn}, rate of coins R_{tn} (number of coins per unit load) and all active load values $l_h(tn)$ should be known. A quantity check on the stock of coins $Co_h(tn)$ needed for each load $l_h(tn)$ from coins available with the respective appliance, is made before the scheduling process is started so that only appliances who have enough coins are allowed to take part in scheduling. A scheduling between rich loads (owning enough coins to pay for demands) will be performed by one of the mentioned LA methods. The selected appliances will have their coins reduced according to the rate prescribed for that timeslot while the rejected loads (due to lack of coins and due to scheduling) will receive those coins. Rich users lose coins while poorer and under-previlaged ones gain them (passive appliances which are not part of the game are excluded). So, their chances of getting scheduled in upcoming timeslots will be influenced due to VCG. In addition to Fig. 4.3, coin game is presented in steps by Algorithm 5 in page 37.

To enter into scheduling, a minimum number of coins is to be possessed by each active appliance, the number of coins determined by the quantity of load $l_h(tn)$ and the rate of coins R_{tn} at that timeslot. The rate of coins can be static (same throughout all timeslots) to make it simple or dynamic (changing like in dynamic electricity pricing and controlled by VLC or user themselves on mutual understanding). In this thesis, all loads from an appliance coexisting in a timeslot are rejected if coins that they have are

Algorithm 5 Coin game (VCG) combined with LA

Input:

- The active loads $l_h(tn)$ and capacity C_{tn} for each timeslot tn, $Co_h(tn)$, R_{tn} T, M, Z

Output:

- $\{d_h(tn)\},\ \sum_{h=1}^{H} l_h(tn)\ d_h(tn),\ Co_h(tn)$.
- ρ_i.

1: **Begin**
2: Declare variables Sel, $Co_h(tn)$ and CC_{tn} and initialize $tn = 1$.
▷ CC_{tn}= coins collected from scheduled loads
3: Initialize $CC_{tn} = 0$, $Sel = 0$.
4: **for** each h **do**
5: **if** $Co_h(tn) \geq R_{tn}\ l_h(tn)$ $\left(R_{tn} \sum_h l_h(tn)\right.$ for multiple loads$\left.\right)$ **then**
6: Yes.
7: **else**
8: No.
9: **end if**
10: **end for**
11: **for** all loads with Yes in step 5 **do**
12: Calculate $d_h(tn)$ by BLA (step 3 to 37 in algorithm 4) or L_{R-I} scheduling.
13: **if** $d_h(tn) = 1$ **then**
14: $Co_h(tn) = Co_h(tn) - R_{tn}\ l_h(tn)$.
15: $CC_{tn} = CC_{tn} + R_{tn}\ l_h(tn)$.
16: $Sel = Sel + 1$ ▷ Sel=counter for scheduled
17: **end if**
18: **end for**
19: **for** each user h **do**
20: **if** $Co_h(tn) < R_{tn}\ l_h(tn)$ or $d_h(tn) = 0$ **then**
21: $Co_h(tn) = \frac{CC_{tn}}{H - Sel}$.
22: **end if**
23: **end for**
24: **if** $tn \neq T$ **then**
25: $tn = tn + 1$.
26: Go to step 3.
27: **else**
28: Go to step 30.
29: **end if**
30: **End**

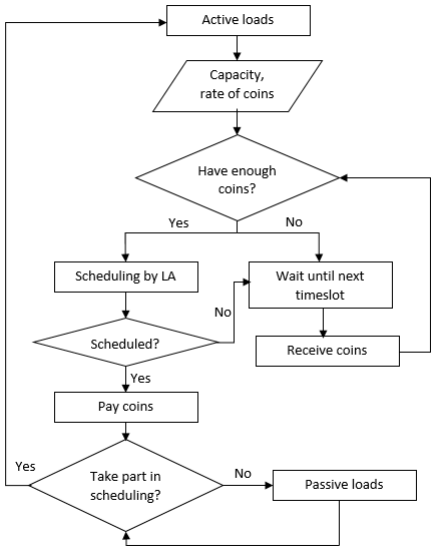


Figure 4.3: Steps showing Coin Game Implementation in Scheduling

insufficient, due to proportional distribution of coins between loads sent by same appliance, rather than prioritizing one or more loads. At least some loads could take part in scheduling if coins were distributed to older or more important ones. But the advantage of proportionate distribution is that the appliances collect coins as no load requests can enter into scheduling. The difference in result due to variation in coin distribution between loads from same appliances can be a subject of further study. Once a load with enough coins gets scheduled, then coins are deducted from the corresponding appliance's stock. Passive appliances that do not have loads will not take part in scheduling and also do not get privilege of adding coins into their stock. Only the appliances with demands (even if they do not have enough coins) will be benefited by coins collected from the scheduled loads. If any appliance i has two or more loads simultaneously in same timeslot (for Type 3 appliances in section 3.1), coins will be reduced from the same stock $Co_h(tn)$ belonging to the appliance that both loads emerged from.

Due to concerns about the existence of parallel load requests from an appliance and availability of coins with that appliance, following tasks are to be performed based on different types of appliances in section 3.1.

- **Case 1** As second load $L_{i1,t2}$ will be shifted by $i1$ if previous load $L_{i1,t1}$ still exists, coins will be checked against $L_{i1,t1}$ at current timeslot.

- **Case 2** Coin check will be made against $L_{i1,t2}$ as it will be the current load. Appliance deactivates $L_{i1,t1}$ as soon as $L_{i1,t2}$ emerges.
- **Case 3** Appliance will divide coins to each load and send both $L_{i1,t1}$ and $L_{i1,t2}$ if coins are enough for serving both. Coins will be distributed proportionately. E.g., $\exists$ ($h1$,$h2$) $\in i1$, then coins distributed to each load is given by $Co_{h1}(tn) = \frac{Co_{i1}(tn)\ l_{h1}(tn)}{l_{h1}(tn)+l_{h2}(tn)}$ and $Co_{h2}(tn) = \frac{Co_{i1}(tn)\ l_{h2}(tn)}{l_{h1}(tn)+l_{h2}(tn)}$ respectively.
- **Case 4** Even though this case is not considered in simulation in this thesis, coin allocation can be discussed. Since only one of the two loads from an appliance exists, coins need not be distributed. So, all coins can be used by the selected load like in case 1 and 2.

4.2.1 VCG Implementation in Real Scenarios

For ease of simulations, different conditions have been considered for implementation of VCG. Some of them have been discussed already in this chapter. But such considerations can be altered for ease of customers and utility company in real SG. These issues are discussed here.

1. The appliances are charged with coins equal to their load demands. In reality, VLC can change the rate of coins depending on the proportion of demand and capacity so that enough loads are available for scheduling. If excess number of loads are rejected, optimal total load can not be achieved due to lack of coins.

2. A preset value of equal coins ($Co_i(tn) = Co_j(tn), tn = 1, i \neq j$), i.e., $\frac{total\ coins}{number\ of\ users}$ is allocated for each user i in the simulations in this thesis. In real life, the preset value can also be set by distributing coins in proportion to the electricity requirement of each user. Appliances whose demands are higher along N timeslots will have bigger stock of coins.

3. The number of coins with each user at the beginning of new round of scheduling can be an issue as well. In this thesis, a reasonable number of coins at $tn = 1$ is given depending upon the appliance types (type 1, 2 or 3) and repeated with same number in next simulation. For real time implementation, two choices can be suggested, either number of coins could be reset to the previous preset value (as in this thesis) or continue with remaining coins after the last T^{th} timeslot of previous round.

4. Fairness is measured from appliance's point of view and coins are allocated to different appliances in this thesis. In reality, when customers are considered, coins can be transferred between appliances of same customer so that important loads can be sent into scheduling. In a broader view, multiple loads from same appliance (type 3) can be viewed as multiple appliances (with single load in each timeslot) from a single user case.

5. The coins received by rejected loads is equal in this work, by equal division of total coins collected from scheduled loads. In practice, the same or division of coins in proportion to their demands can be done, i.e., loads with higher values get more coins as they are the ones who will be paying more coins when scheduled. The same algorithms as suggested in this thesis are sufficient to serve the purpose, where the index i will be customer index and h will become appliance index.

The above mentioned implementation aspects are suggestions and may need simulations to see results before enforcing into action in SG. This is considered further work in extension to this thesis. The implementation of the algorithms, discussion and analysis of the results is presented and discussed in the next chapter.

Chapter 5

Simulations and Numerical Results

After formulation of the problem and outlining the solution methods in the previous chapter, the next step is to see if the results will be as expected. This chapter proceeds into implementation of prescribed algorithms. MATLAB is the platform chosen for simulation of algorithms. Results have been obtained by simulating the coin game into the BLA and L_{R-I} based scheduling methods. Results will be focused on the comparison of fairness between users (ρ) taking part in scheduling and fairness ($\mathfrak{F}$) achieved by system given in section 5.3 and the accuracy of converged decision given in section 5.2. The MATLAB codes are given in Appendix section 7.4 and 7.5.

In the simulations, the number of appliances N was fixed to 15 while the comparison of accuracy and fairness of scheduling is made as a function of the number of timeslots T, LA methods and inclusion/exclusion of VCG in the scheduling processes. The load demands, provided that they get rejected during scheduling, exhibit different behaviour at different timeslots according to nature of appliances, and results are also presented in different subsections under cases 1, 2 and 3 corresponding to appliance types 1, 2 and 3 in 3.1 respectively. Appliances of case 4 are not been considered in scheduling due to selection issues. Case 4 can have a compromised solution (for scheduling) as it can be treated as case 1 by allowing the appliances or customers themselves to select any one of the coexisting loads. Also, bad selection of loads (specially if difference between loads $L_{i1,t1}$ and $L_{i1,t2}$ is high) might lead to a drift of the sum of selected loads away from the capacity C_{tn}. The drift from capacity leads to under utilization of available capacity in case 4.

Even though the values of loads from the appliances are not known in advance in reality, to create uniformity in different scheduling scenarios and better comparison of results, all results are derived by considering loads from the Table 7.1 of matrix size 100×15 given in Appendix 7.1. Each row of the table contains loads for each timeslot in ascending order and the columns are for the 15 appliances considered. E.g., load of 5^{th} appliance at 10^{th} timeslot $L_{5,10}$ is given by element $[10, 5]$ of demand table, which is 53 units. A load demand of 0 units at any timeslot means that the appliance does not generate any load in that timeslot, e.g., load of 2^{nd} appliance at 1^{st} timeslot, $L_{2,1} = 0$ units. Depending upon the appliance type considered (except type 2) and scheduling results from previous timeslots, the actual load demands at any timeslot during simulations alter from loads given in load table. For $T = 25$ and $T = 50$ conditions, the first 25 and 50 rows should be picked.

These loads were simulated in MATLAB under two scenarios of capacity for secondary loads as mentioned in following cases:

- **Insufficient Capacity** In this case, $\sum_t \sum_i L_{i,t} > C$ holds. More specifically, the capacity is fixed to 60% of total demand request in each timeslot. It means the capacity varies proportionately with the total demand. For timeslots with high demands, the capacity is higher and lower for timeslots with smaller demands. The simulations were performed taking T=25 timeslots for comparison of accuracy and T is varied between 25, 50 and 100 for comparison of fairness.

- **Sufficient Capacity** Capacity is made variable at different timeslots such that the condition $\sum_t \sum_i L_{i,t} \leq C$ is fulfilled by capacity. This scenario presumes a fully fledged power usage environment for appliances. Numerical results consider T=25 timeslots only for this assumption. The capacities for each timeslot are taken from Appendix 7.2. The total capacity $C = \sum_{tn=1}^{T} C_{tn}$ is not equally distributed along the timeslots, but rather randomly distributed. So at any tn, C_{tn} can be greater than, equal to or less than $\sum l_h(t_n)$. The capacities have been chosen randomly, and the capacity for each timeslot is sometimes higher and sometimes lower than the total of demands in each timeslot but at the end, the total of loads in successive timeslots (until T=25) is smaller than the sum of capacities during the same timeslots. In those timeslots where capacity is greater than the demands, decision 1 for all users can be done straightforward. Scheduling by LA is not necessary in such situations and measurement of ability of scheduling to match the global optimal point by proper selection of users is not suitable in these cases.

The *sufficient capacity* scenario means that the capacity of SG is large enough to accomodate aggregated loads in a long run, as a result of which loads that are rejected get served within the next few schedulings. This is quite common as the SG capacity is almost constant, with loads accumulating at peak hours and fewer loads appearing at non peak hours. The loads rejected at timeslots with lower capacities are expected to get a count on at higher capacity timeslots. The practice of adjusting the loads for balancing the supply and demand, load shaping as said commonly, is outlined in [14] as peak clipping, load shifting, valley filling, conservation, etc. in accordance to the scenario. The loads rejected at peak times will succeed in the off-peak times if there is enough capacity. However, even if $\sum_t \sum_i L_{i,t} \leq C$ holds, there is no guarantee that all loads will finally get served. Lets take an example with two timeslots with demands {7,8} units and {5,2} units in each timeslot. Let capacities for each slot be 10 units and 12 units respectively, so $\sum_t \sum_i L_{i,t}$ (7+8+5+2=22 units) = C_t (10+12=22 units). Only one of two loads in first timeslot can be scheduled, say 8 units gets scheduled. So, second timeslot will have total demand=14 units which is greater than the capacity=12 units for that timeslot. That is why, even though the assumption holds, it is not always rightful to say that all loads will finally get scheduled, due to granularity of loads. Therefore this scenario deserves a study and will be discussed later in the results section.

Again the results are also presented under the different cases as case 1, case 2 and case 3, depending on the behaviour exhibited by appliances for their load as already discussed in section 3.1.

- **Case 1** The results under this section consider the loads from type 1 appliances. As discussed already, the appliances send rejected loads from previous timeslot in the current timeslot to replace the loads that would possibly emerge in the current timeslot. That shifts the current load (given in Table 7.1) to the next timeslot ($L_{i,tn} \rightarrow L_{i,tn+1}$) and the other consecutive loads further by one timeslot ($L_{i,tn+1} \rightarrow L_{i,tn+2}$ and so on). So, the loads in succeeding timeslots for any appliance will not be exactly as in load table given in Appendix 7.2 once the appliance's load has been rejected. Occurrence of multiple loads from the same appliance at any timeslot does not happen in this case.

- **Case 2** Case 2 considers the type 2 appliances and loads follow the same pattern as discussed already. Only one load per timeslot from one appliance exists under case 2, and it will undergo scheduling. The old rejected loads $L_{i,t}$ from previous timeslots $t < tn$ will be discarded

in the current timeslot if new load is generated at the current timeslot. Only if a load of 0 units exists in current timeslot (in load matrix in Appendix 7.1), the old rejected load is continued in current timeslot assuming that same load needs to be served as the appliance's load has not been updated.

- **Case 3** In this scenario, the appliances' demands that remained unserved in previous timeslots are also considered in the current timeslot regardless of presence or absence of new demand in current timeslot. The loads at current timeslot will also be considered in simulation along with previous unserved loads. So, there is accumulation of loads with progression of timeslots due to lack of coins or rejection by scheduling. The loads will repeat continuously in following timeslots if they are rejected in the current timeslot and they will contend until they are scheduled.

5.1 Considerations for Simulation

Some considerations have been carried out during scheduling for ease of simulations, are given as follows:

1. For every simulation, the load requests for each appliance are predetermined and are taken randomly between 50 units and 100 units as shown in Appendix 7.1. The capacity for each timeslot for sufficient capacity scenario is set to values as given in Appendix 7.2.

2. All loads have completion time of single timeslot only. In reality, if shiftable loads can be stopped even after they have once started, scheduled appliances can undoubtedly take part again in consecutive timeslots. Shiftable loads that can not be stopped once scheduled and last for more than one time slot can be treated as non-shiftable loads after that timeslot and they will therefore be served without interruption regardless of duration of each load.

3. The rate of coins R_{tn} has been set to 1 in the simulations. It can be made changeable for each timeslot in order to obtain a balance between total demand and capacity, so that enough loads are permitted into game.

One of the issues with appliances is that not all appliances will take part equally. Some appliances need to operate more often while others may

come into usage less frequently. So the value of Q_i in equation 3.10 for the first kinds is higher than the latter ones which might significantly impact ρ values. The demands with value 0 units[1] for the appliances in Table 7.1 refer to timeslot where appliances remain passive, as they do not have any demands. The number of timeslots where an appliance remained passive is shown by white bars in figures for comparison of fairness, e.g. Fig. 5.3. But the effect in fairness due to difference in the number of demand requests has not been studied in this thesis.

5.2 Results for Comparison of Accuracy of Scheduling

The purpose of the simulation results in this section is to check if the scheduled load due to converged decisions leads to the maximum objective function. The ratio of total demands selected in each timeslot against the global optimal point within T timeslots is observed and results are compared between different methods with and without application of VCG.

The LA methods' competency to bring scheduled load $\sum l_h(tn)\ d_h(tn)$ equal to or closest to global optimal point for a single timeslot C_{tn} has already been discussed in previous work [19]. The accuracy in results is presented after measuring accuracy along multiple timeslots, in contrast to previous work where a single timeslot was under consideration, and the average is taken for each timeslot from multiple simulations. The number of coins ($Co_i(tn)$) is 400 coins allocated to each user at first timeslot ($tn = 1$) and total number of timeslots for simulations T=25, is the common environment for case 1 and case 2. 500 coins have been allocated for simulations in case 3 unless stated otherwise.

Even though the closeness (accuracy) depends on the number of demands and their values as well [53], a restricted comparison is presented in this section for the same input demand matrix in Appendix 7.2. Tables and figures are used to discuss the results, where the values in table represent the average accuracy in percentage from the ratio of selected demands to global optimal point[2] for a particular timeslot, i.e., $\frac{\sum_{h=1}^{H} l_h(tn)\ d_h(tn)}{global\ optimal\ point} \times 100$ and figures are plots of *total demand* $\sum_{h=1}^{H} l_h(tn)$, capacity C_{tn}, sum of all selected loads $\sum_{h=1}^{H} l_h(tn)\ d_h(tn)$ and *global optimal point* for each timeslot. Due to large number of figures from several scenarios, figures from case 1

[1] The values were allocated randomly to some appliances for each timeslot.

[2] The global optimal point is calculated by exhaustive search.

(appliances of type 1) only has been selected in this section. Rest of the figures have been given in Appendix 7.3. The figures presented for this subsection have four bars for each timeslot, where the first bar in legend denoted by *Total demand* is for $\sum_{h=1}^{H} l_h(tn)$, second bar *Capacity* is for C_{tn}, third bar *Scheduled demand* is for $\sum_{h=1}^{H} l_h(tn)\ d_h(tn)$ and finally, *global optimal* is for the global optimal point that can be achieved for that timeslot. The figures are obtained from a particular simulation and do not exactly exhibit the average results as shown in Tables 5.1 and 5.2.

5.2.1 Insufficient Capacity

This scenario of insufficient capacity considers the presumption where capacity is 60% of total demands. All three cases of appliance types are included together under this subsection. Table 5.1 presents results for all three cases. The results in figure for case 1 are given in Fig. 5.1, while figures for case 2 and case 3 are given in Fig. 7.2 and Fig. 7.3 respectively in Appendix section 7.3. Due to the total load request from all appliances for all timeslots being above the capacity for all timeslots, the global optimal point is mostly the capacity C_{tn} of those timeslots.

The results in Table 5.1 show that the scheduling by the two LA methods are good enough to achieve optimal usage of capacity. It is demonstrated by high accuracy values, that both methods obtain total scheduled load closest to global optimal point which helps appliances to capitalize the most from available capacity. But the slightly better results in terms of better accuracy for scheduling without VCG compared to VCG considered, shows that a trade-off exists between better fairness and accuracy of scheduling (optimization of capacity). The fact that scheduling by LA with VCG limits some privileged users from scheduling, leads to less number of participants in scheduling, so the accuracy is slightly lower than scheduling without VCG. The same reason explains the better results for case 3 than case 1 and 2 which has more loads in scheduling in most timeslots.

In case 3, the scheduling processes took a longer time than other two cases for convergence in each timeslot. This is due to the higher number of participants involved in scheduling. For L_{R-I} methods, the speed of convergence can be increased by increasing value of λ but certainly the accuracy of convergence needs to be sacrificed. Also, BLA shows a slender better edge in terms of accuracy in case 3 with insufficient capacity, which means that BLA has better edge over L_{R-I} when more participants are involved. Figures 5.1, 7.2 and 7.3 can be used to compare the results for differences in scheduled load by simulations and global optimal point for

different cases with insufficient capacity.

5.2.2 Sufficient Capacity

The results for accuracy with sufficient capacity ($\sum_{tn}\sum_{h} l_h(tn) \leq \sum_{tn} C_{tn}$) are presented in Fig. 5.2 (case 1 scenario only) and Table 5.2. All simulations have considered 25 timeslots as in previous simulations of insufficient capacity. Figures for case 2 and case 3 have been shown in Appendix 7.3 in Fig. 7.4 and Fig. 7.5 respectively.

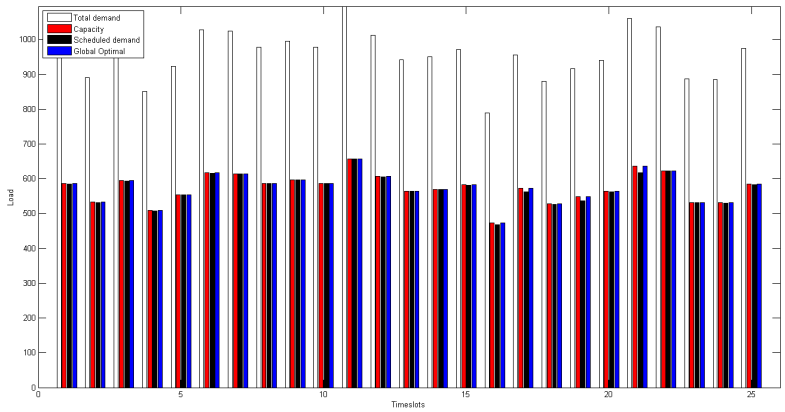
Again, the results show that the LA methods used are good enough to achieve high accuracy in optimizing the capacity of SG when the capacity is sufficient to accommodate all loads within 25 timeslots. Since the accuracy is measured in terms of total scheduled load versus global optimal point in each timeslot, the accuracy in case 1 and 2 is negligibly lower than in case 3 because case 1 and 2 have fewer demands in most timeslots (maximum 1 demand per appliance per timeslot) even though capacity is enough to accommodate all loads and no scheduling is required. In case 3, the number of total demands is higher than or at least equal to the former two cases in most timeslots. Better scheduling precision can be achieved at the expense of longer scheduling time. Also, it can be noticed from the results in figures, that the total demand (first bar) at $tn = 25$ is much higher than the capacity (third bar) at that timeslot. These are much notable in case 3, Fig. 7.5, after timeslot 12 and timeslot 18. Even though the sufficient capacity condition considered is $\sum_{tn}\sum_{h} l_h(tn) \leq \sum_{tn} C_{tn}$, which means all loads are expected to be scheduled by last timeslot, it is not the case. This is due to the granularity of the loads. Another reason is that in the first few slots, there may be less demands even though the capacity is high. So the capacity in those timeslots is wasted. The total demand becomes higher than capacity due to rejection after timeslot 12, but the capacity does not increase accordingly. And the wasted capacity in previous timeslots can not be utilized later, which explains the existance of more rejected loads' even though capacity is sufficient.

Table 5.1: Comparison of Accuracy of Scheduling Methods (Insufficient capacity)

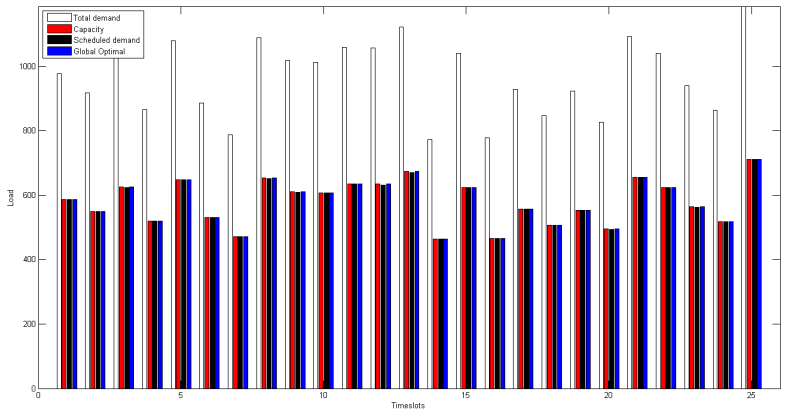
Case	T	Method	VCG	Accuracy(in %)
Case 1	25	BLA	Yes	99.7529
			No	99.8475
		L_{R-I}	Yes	99.8089
			No	99.9010
Case 2	25	BLA	Yes	99.7452
			No	99.8659
		L_{R-I}	Yes	99.8721
			No	99.9211
Case 3	25	BLA	Yes	99.9872
			No	99.9931
		L_{R-I}	Yes	99.9830
			No	99.9989

Table 5.2: Comparison of Accuracy of Scheduling Methods (Sufficient Capacity)

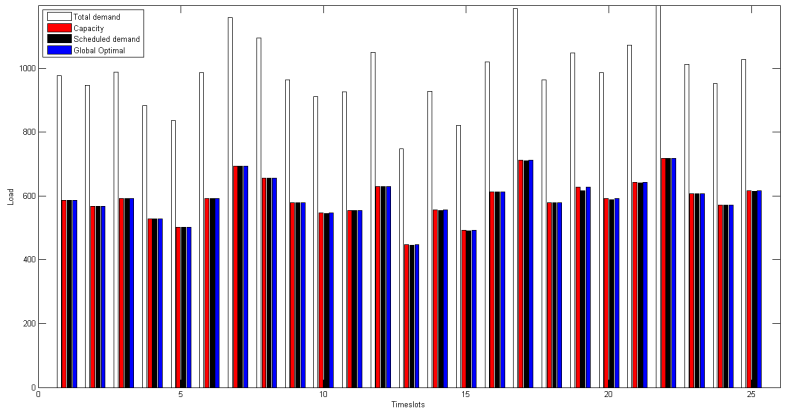
Case	T	Method	VCG	Accuracy(in %)
Case 1	25	BLA	Yes	98.2953
			No	99.8220
		L_{R-I}	Yes	98.7093
			No	99.7399
Case 2	25	BLA	Yes	98.5226
			No	99.8017
		L_{R-I}	Yes	98.5009
			No	99.8486
Case 3	25	BLA	Yes	99.5678
			No	99.8513
		L_{R-I}	Yes	99.8340
			No	99.8266



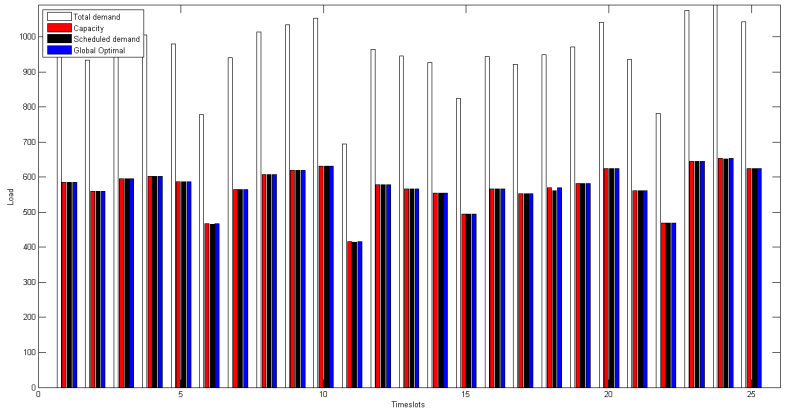
(a) BLA with VCG, Case 1



(b) BLA without VCG, Case 1



(c) $L_{R-I}(\lambda = 0.15)$ with VCG, Case 1

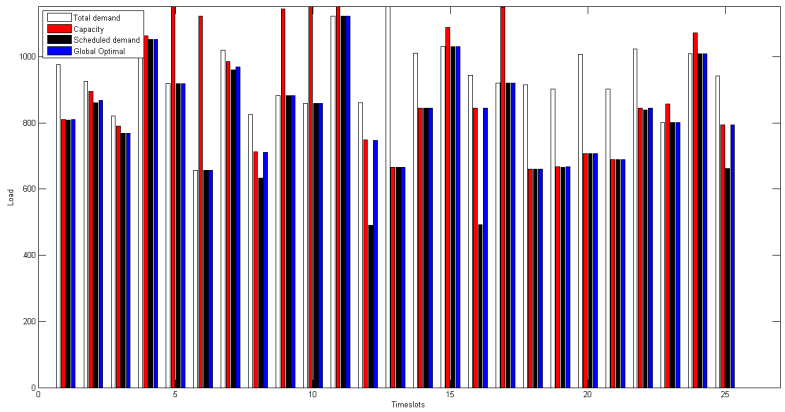


(d) $L_{R-I}(\lambda = 0.15)$ without VCG, Case 1

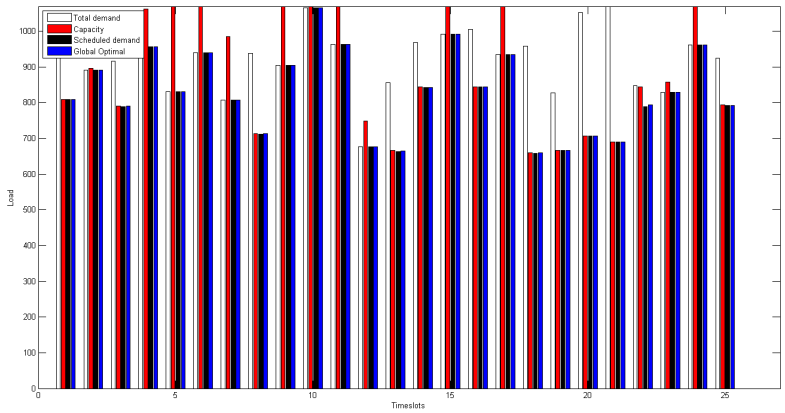
Figure 5.1: Comparison of Accuracy by BLA and L_{R-I} (Case 1, $T = 25$, Insufficient Capacity)

5.3 Results for Comparison of Fairness

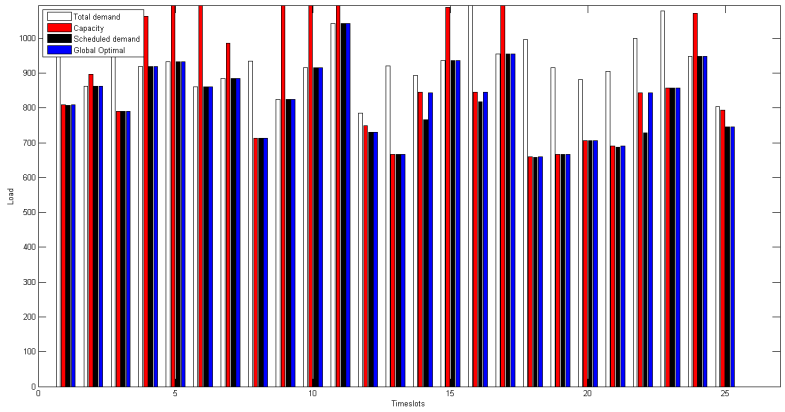
Here, the simulation results have been presented separately under the cases 1, 2 and 3 representing the difference in nature of loads from appliances in section 3.1. Results under case 1 are presented by assuming that all loads are from appliances of type 1. Similarly, case 2 assumes that loads are from type 2 appliances and case 3 for loads from type 3 appliances. Though all kinds of



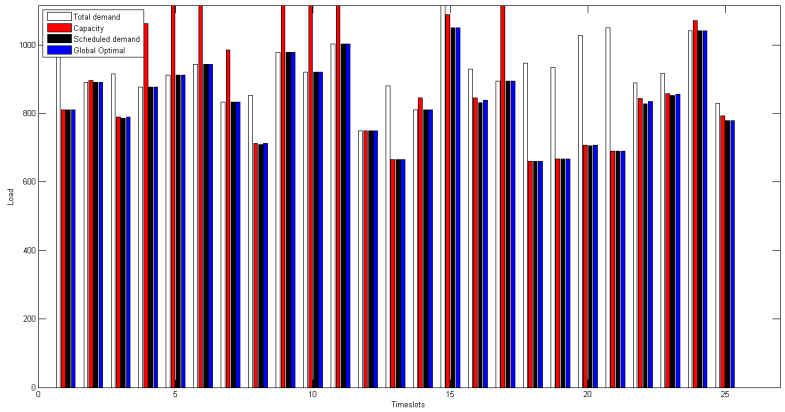
(e) BLA with VCG



(f) BLA without VCG



(g) $L_{R-I}(\lambda = 0.15)$ with VCG



(h) $L_{R-I}(\lambda = 0.15)$ without VCG

Figure 5.2: Comparison of Accuracy by BLA and L_{R-I} (Case 1, $T = 25$, Sufficient Capacity)

appliances exist simultaneously in reality, simulations have considered that all appliances exhibit same behaviour for the loads. However, the nature of the problem does not change even if appliances of all types coexist, so the algorithms given in this thesis are always applicable. Furthermore, the number of loads or participants of scheduling will be even smaller (when appliances of all types coexist) than in case where all loads belong to type 3 only. The figures obtained from each round of scheduling contain four terms in legend (three for scheduling without VCG), namely *scheduled*, *lack of coins*, *rejected* and *passive* for each appliance/user (referred to as *USER* in x-axis) and the *Number of Timeslots* is given in y-axis. Legend *scheduled* refers to the number of timeslots each appliance was selected in the scheduling process. Similarly, *lack of coins*[3], *rejected* and *passive* refer to the number of timeslots that each appliance, i. could not participate in scheduling due to insufficiency of coins, ii. got rejected during scheduling process despite having enough coins and iii. did not have any load request respectively. The figures and tables presented may not be from exactly the same simulation configurations, so differences may exist (in values of P_i) even if the scenario is same.

The results of simulation presented in tabular form consist of five columns as shown in sample Table 5.3. The *Method* column is for LA method used in scheduling which is either BLA or L_{R-I}. Column T is for number of timeslots that each round of simulation lasts for. Simulations were done for 25, 50 and 100 timeslots, so column T takes values *25*, *50* or *100*. Column *VCG* suggests if VCG was applied along with LA scheduling method and expressed by *Yes* or *No* depending on whether VCG was applied or not. Similarly, column ρ_i is for the ρ values of each appliance i and their values are in separate columns under each appliance's index i taking values *1, 2, . . . , 15* for 15 appliances considered in simulations. Finally, column $\mathfrak{F}$ gives the $\mathfrak{F}$ value of the scheduling process calculated from the ρ values for each simulation, as per equation 3.11. An extra row has been added for P_i under every ρ_i value for $T = 100$ only, which is the count of total number of timeslots each user was selected within those 100 timeslots. To be noted, the results for fairness are taken from a single simulation, not the average of several simulations as in previous section for comparison of accuracy.

[3] *Lack of coins* does not exist in figures that do not consider VCG.

Table 5.3: Sample Table for Results of Measuring Fairness

Method	T	VCG	ρ_i															$\mathfrak{F}$
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
BLA/L_{R-I}	25/50/10	Yes/No																

5.3.1 Insufficient Capacity

As already discussed in the previous section, the capacity available for the secondary appliances for any timeslot is considered to be 60% of the total loads to be scheduled at that timeslot. As the loads from the appliances vary at different timeslots, the capacity is also variable but in proportion to the total load demand. So, it is assumed that SG supplies higher capacity during timeslots with higher load demand and lower capacity for lower load demand. The results have been classified under three separate cases to resemble the appliances of types 1, 2 and 3 respectively, excluding appliances of type 4.

Case 1

The results from MATLAB simulations have been presented in tabular form in Table 5.4. Since VCG is the main aspect expected to influence the fairness $\mathfrak{F}$, the results of simulations with and without VCG have been put together for ease of comparison.

As expected, the fairness $\mathfrak{F}$ among appliances is higher with application of VCG than without it for both BLA and L_{R-I}. It is due to the fact that the frequency of selection of appliances given by their ρ_i values are closer to each other while more dispersion is seen in their values in scheduling without VCG. E.g., ρ values are limited between 0.5714 and 0.5977 in BLA with VCG with T=100 while 0.4782 and 0.7303 are the minimum and maximum values for the same scenario without VCG. Also, argument that users with the same P_i can not be guaranteed to have equal ρ values within same T is demonstrated. E.g., both appliances i=4 and i=9 in L_{R-I} ($\lambda = 0.10$) with T=100 and VCG applied, are selected 53 times but yield different ρ values, i.e., ($\rho_4 = 0.6091$) and ($\rho_9 = 0.5955$). This is due to the difference in number of timeslots they were rejected.

Furthermore, comparing the methods of scheduling, the fairness brought by L_{R-I} with all λ values (0.1, 0.15 and 0.3) is almost equal to that of BLA when VCG is applied. Even when VCG is not applied, scheduling results by L_{R-I} are similar to those by using BLA. Notably, scheduling without VCG also delivers better fairness with increase in T (T=100 has $\mathfrak{F}$=0.9908 and T=25 has $\mathfrak{F}$=0.9864 for L_{R-I} with λ=0.15 without VCG), even though

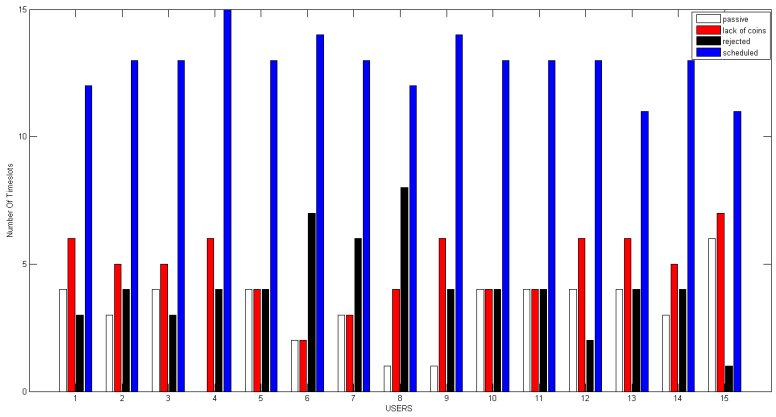
it is slightly less compared to its counterpart which considers VCG.

Figures are also presented to exhibit the results of simulations for case 1, which are presented in Fig. 5.3 and Fig. 5.4. The main focus is to balance the height of last bar[4] (P_i) in the figures for all appliance. Figures explain the results shown in tabular form in more details. Four plots in Fig. 5.3 for T=25 and four others in Fig. 5.4 for T=50 have been chosen to compare results from simulations of BLA and L_{R-I} (with λ=0.15) respectively. Two sub-figures for each LA method are chosen, first one with application of VCG and the latter without VCG. Even though the bar plots from L_{R-I} seem more uniform than plots obtained from BLA, it is hard to notice the difference between them in figures. But the better fairness achieved by application of VCG is clear for both $T = 25$ and $T = 50$ scenarios with both L_{R-I} and BLA scheduling methods. In average, the height of bars representing P_i are shorter when VCG is applied than when VCG is not applied. Better fairness is achieved by cutting off P_i of some users, e.g., only user 4 has been selected 15 times in Fig. 5.3a while four users (user 1, 4, 6 and 13) in Fig. 5.3b have been selected at least 16 times. Fig. 5.5 can be considered for T=100 even though it is obtained from case 2 scenario, as the results in case 1 and case 2 showed much resemblances. The number of times users are scheduled is more uniform in Fig. 5.4 for $T = 50$ than in Fig. 5.3 for $T = 25$. Again, this supports the evidence that fairness increases with increase in number of timeslots. Results from previous work [19] and simulations for L_{R-I} with different values of λ show that L_{R-I} with λ=0.15 is arguably better choice than λ=0.10 and λ=0.30, considering the speed of simulation and accuracy. So, it can be said that the effect of VCG is more when less timeslots are involved, and negligible differences are seen between L_{R-I} and BLA methods in terms of fairness.

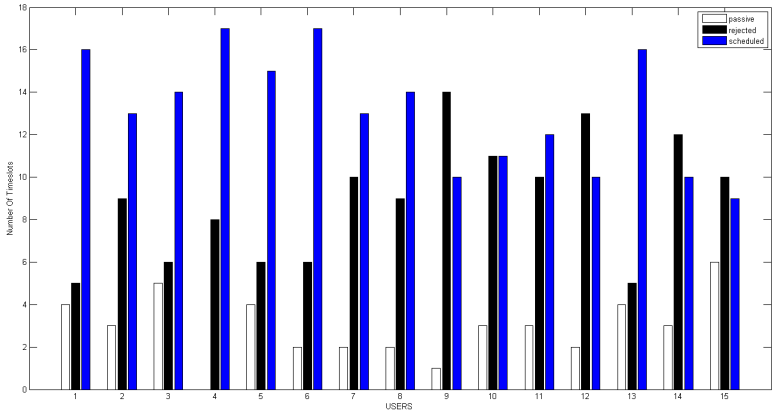
[4]Forth bar in simulations with VCG and third bar for simulations without VCG

Table 5.4: Comparison of Fairness of appliances (Case 1, Insufficient capacity)

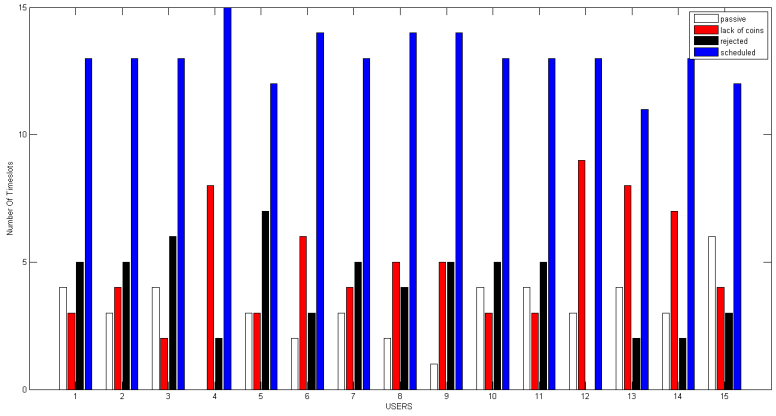
Method		T	VCG	ρ_i 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$\mathfrak{F}$
BLA		100	Yes	0.5730 51	0.5714 48	0.5833 49	0.5977 52	0.5806 54	0.5977 52	0.5556 50	0.5889 53	0.5955 53	0.5778 52	0.5824 53	0.5764 49	0.5909 52	0.5795 51	0.5747 50	0.9996
			No	0.7303 65	0.5058 43	0.6309 53	0.5632 49	0.6304 58	0.5747 50	0.4782 44	0.5889 53	0.5444 49	0.5842 52	0.4946 46	0.6309 53	0.6279 54	0.6395 55	0.6321 55	0.9882
		50	Yes	0.5778	0.5909	0.5813	0.5556	0.6000	0.5909	0.5744	0.5681	0.6000	0.5909	0.5681	0.5952	0.5813	0.5681	0.5853	0.9995
			No	0.6136	0.6136	0.6341	0.5333	0.6222	0.6279	0.5744	0.5111	0.5778	0.6136	0.6818	0.5111	0.5348	0.7045	0.5000	0.9896
		25	Yes	0.5714	0.5909	0.6190	0.6000	0.6190	0.6086	0.5909	0.5000	0.5833	0.6190	0.6190	0.6190	0.5238	0.5909	0.5789	0.9966
			No	0.7619	0.5909	0.7000	0.6800	0.7142	0.7391	0.5652	0.6086	0.4166	0.5000	0.5454	0.4347	0.7619	0.4545	0.4736	0.9615
L_{R-I}	$\lambda = 0.10$	100	Yes	0.5730 51	0.5833 49	0.5952 50	0.6091 53	0.5913 55	0.5977 52	0.6000 54	0.6000 54	0.5955 53	0.5955 53	0.5934 54	0.5882 50	0.5909 52	0.5862 51	0.5862 51	0.9998
			No	0.6067 54	0.6190 52	0.6309 53	0.5402 47	0.6304 58	0.5454 48	0.5164 47	0.6627 57	0.6067 54	0.5556 50	0.6154 56	0.6071 51	0.6136 54	0.5681 50	0.5747 50	0.9956
		50	Yes	0.6000	0.5435	0.6341	0.5333	0.5778	0.6279	0.5745	0.6429	0.5778	0.5455	0.5909	0.6341	0.5581	0.6136	0.5854	0.9966
			No	0.7500	0.6591	0.4651	0.5111	0.4783	0.5455	0.7174	0.5111	0.6136	0.6364	0.5682	0.6341	0.5581	0.6136	0.6585	0.9815
		25	Yes	0.5238	0.6364	0.7000	0.6800	0.6190	0.4583	0.5909	0.4583	0.6667	0.6500	0.6500	0.5455	0.6667	0.5909	0.4737	0.9821
			No	0.7143	0.5909	0.5000	0.4400	0.5909	0.6522	0.7273	0.6522	0.5417	0.5714	0.6190	0.6667	0.4286	0.6818	0.5263	0.9777
	$\lambda = 0.15$	100	Yes	0.5955 53	0.5833 49	0.5952 50	0.6091 53	0.5744 54	0.5862 51	0.5888 53	0.5777 52	0.5842 52	0.5955 53	0.5934 54	0.5882 50	0.5909 52	0.5977 52	0.5862 51	0.9998
			No	0.6067 54	0.6547 55	0.6428 54	0.5287 46	0.5319 50	0.6279 54	0.5384 49	0.5555 50	0.5222 47	0.5108 47	0.6923 63	0.5340 47	0.6206 54	0.6206 54	0.6321 55	0.9908
		50	Yes	0.6000	0.5652	0.6190	0.6000	0.6000	0.5909	0.5957	0.6047	0.5778	0.6136	0.5910	0.6098	0.5814	0.6136	0.6098	0.9994
			No	0.6364	0.5909	0.6585	0.6444	0.5556	0.5682	0.5745	0.4889	0.6000	0.6977	0.6591	0.5227	0.5814	0.6591	0.5000	0.9897
		25	Yes	0.6190	0.5909	0.6190	0.6000	0.5454	0.6087	0.5909	0.6087	0.5833	0.6190	0.6190	0.5909	0.5238	0.5909	0.6316	0.9978
			No	0.6190	0.5000	0.5714	0.6800	0.6667	0.5652	0.5909	0.5416	0.6250	0.7000	0.6190	0.4347	0.5714	0.5454	0.6667	0.9864
	$\lambda = 0.30$	100	Yes	0.5730 51	0.5647 48	0.5595 47	0.5747 50	0.5744 54	0.5747 50	0.5667 51	0.5667 51	0.5618 50	0.5778 52	0.5652 52	0.5517 48	0.5454 48	0.5568 49	0.5747 50	0.9997
			No	0.6966 62	0.7778 63	0.4883 42	0.4886 43	0.5212 49	0.5681 50	0.6292 56	0.5556 50	0.4945 45	0.5000 46	0.5934 54	0.6667 56	0.6705 57	0.5568 49	0.5632 49	0.9797
		50	Yes	0.6136	0.5909	0.5116	0.5333	0.6000	0.5682	0.5957	0.5111	0.6136	0.5909	0.5909	0.6098	0.6047	0.5909	0.6098	0.9966
			No	0.5778	0.5652	0.5349	0.6444	0.5556	0.5682	0.5106	0.5333	0.6591	0.6136	0.6364	0.7250	0.4884	0.6364	0.4524	0.9856
		25	Yes	0.6190	0.5909	0.6190	0.6000	0.4545	0.6522	0.6364	0.5417	0.5000	0.5455	0.5455	0.6190	0.6190	0.6364	0.6316	0.9912
			No	0.5238	0.4091	0.6500	0.6400	0.3913	0.4583	0.7273	0.6522	0.3333	0.7500	0.5455	0.5455	0.6667	0.7273	0.7778	0.9489



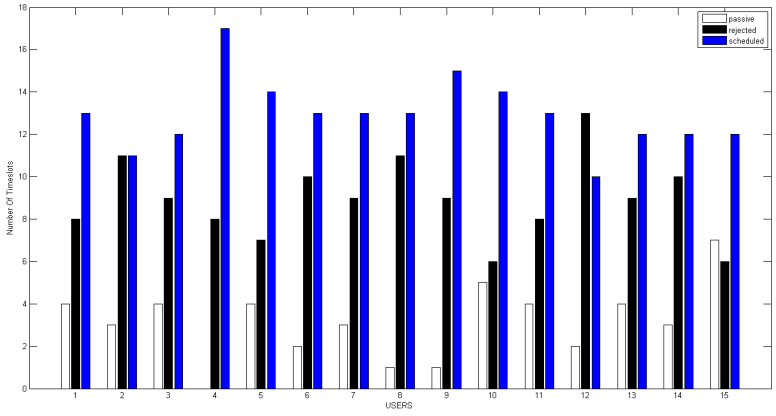
(a) *BLA* with VCG



(b) *BLA* without VCG

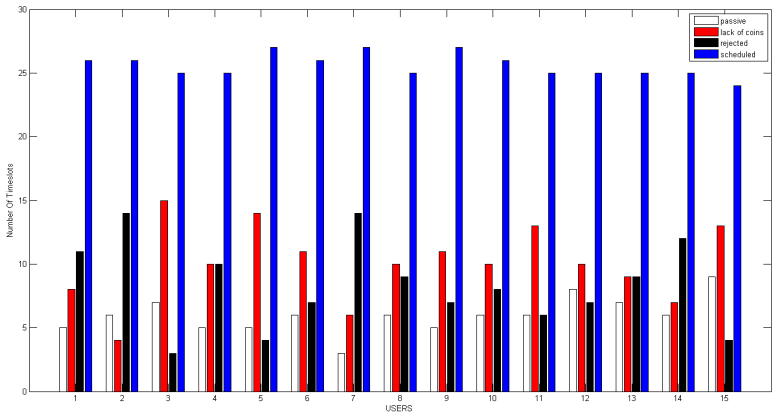


(c) $L_{R-I}(\lambda = 0.15)$ with VCG

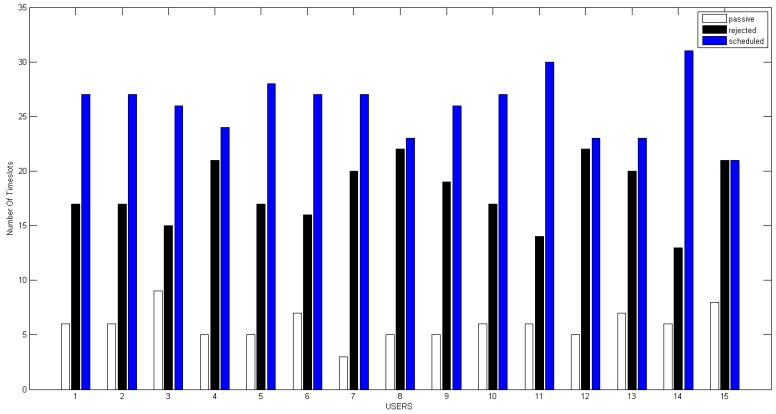


(d) $L_{R-I}(\lambda = 0.15)$ without VCG

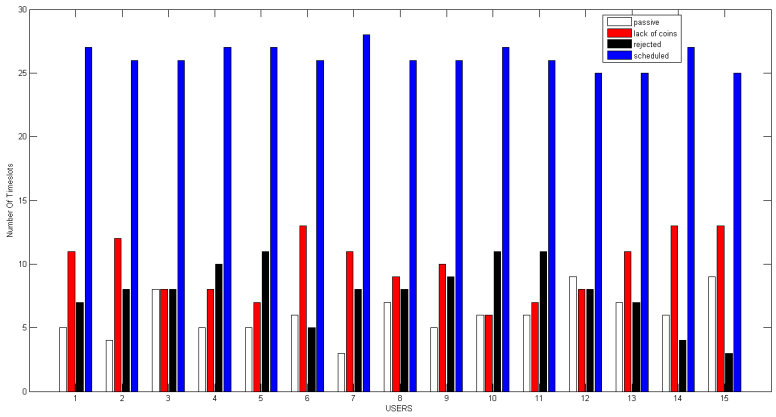
Figure 5.3: Comparison of Fairness between appliances by BLA and L_{R-I} (Case 1, $T = 25$, Insufficient capacity)



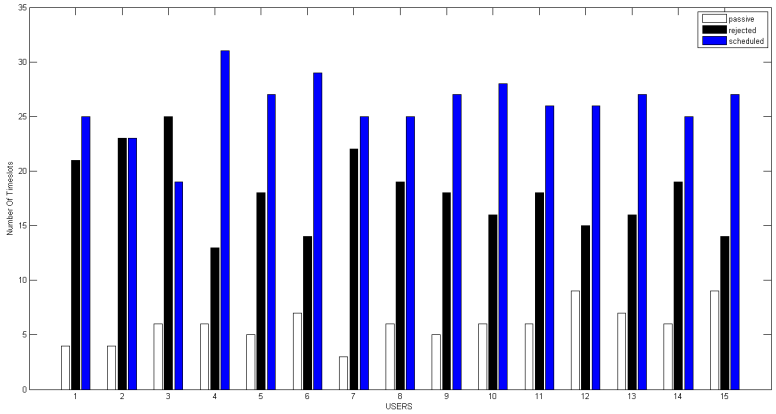
(a) *BLA* with VCG



(b) *BLA* without VCG



(c) $L_{R-I}(\lambda = 0.15)$ with VCG



(d) $L_{R-I}(\lambda = 0.15)$ without VCG

Figure 5.4: Comparison of Fairness between appliances by BLA and L_{R-I} (Case 1, $T = 50$, Insufficient capacity)

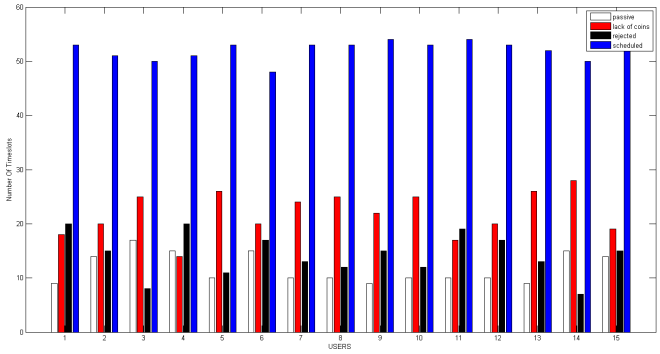
Case 2

The results of simulation in MATLAB for appliances of load types in case 2 are presented in Table 5.5. The result of $T = 100$ for both L_{R-I} ($\lambda = 0.15$) and BLA is given in Fig. 5.5. Figures for $T = 25$ and $T = 50$ have been skipped in this subsection as the results follow the same pattern as for case 1. So the results of case 1 and case 2 can be taken into reference interchangeably. This means the discussion of results for case 1 in previous subsection applies

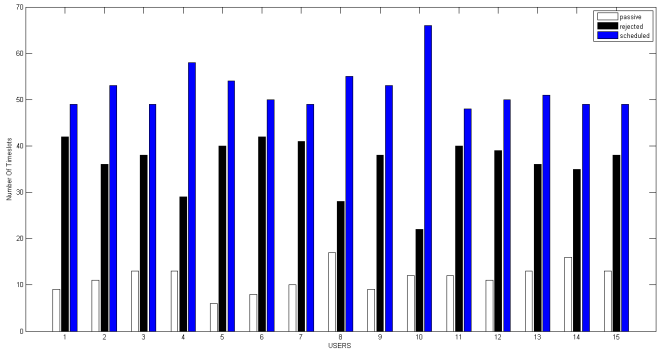
in case 2 as well. From the results in Table 5.5, it can be easily seen that application of VCG increases the fairness and the influence of VCG is more at lower values of T. The value of $\mathfrak{F}$ increases with increase of T. Both L_{R-I} and BLA methods of scheduling are equally efficient with the results from two methods comparable under same scenario of T and VCG. The explanation for similar behaviour despite different appliance types in these two cases 1 and 2 is due to the existance of single load per appliance per timeslot. For comparison of results, it is not important how the loads are treated by appliances before scheduling, unless the number of loads per participant (and their values) are differed [34].

Table 5.5: Comparison of Fairness of appliances (Case 2, Insufficient capacity)

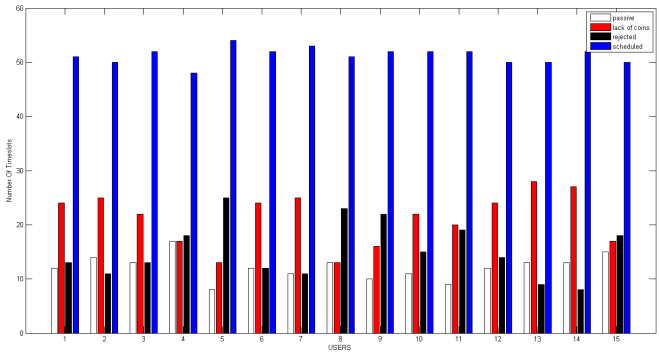
Method		T	VCG	ρ_i 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$\mathfrak{F}$
BLA		100	Yes	0.5824 53	0.5930 51	0.6024 50	0.6000 51	0.5889 53	0.5647 48	0.5889 53	0.5889 53	0.5934 54	0.5889 53	0.6000 54	0.5889 53	0.5714 52	0.5882 50	0.6046 52	0.9996
			No	0.5384 49	0.5955 53	0.5632 49	0.6667 58	0.5744 54	0.5434 50	0.5444 49	0.6626 55	0.5824 53	0.7500 66	0.5454 48	0.5617 50	0.5862 51	0.5833 49	0.5632 49	0.9908
		50	Yes	0.5869	0.5853	0.6153	0.6250	0.5555	0.5531	0.5434	0.5777	0.5777	0.5833	0.5556	0.5777	0.6046	0.5909	0.5853	0.9985
			No	0.5909	0.6511	0.6744	0.5609	0.5957	0.6000	0.7441	0.7619	0.3478	0.5652	0.5333	0.5681	0.5952	0.5348	0.6136	0.9759
		25	Yes	0.5833	0.5909	0.5454	0.5416	0.5909	0.5833	0.5833	0.5909	0.5416	0.6086	0.6086	0.6086	0.6000	0.6000	0.5909	0.9985
			No	0.5217	0.5909	0.5454	0.4583	0.8095	0.5652	0.4545	0.5652	0.6250	0.5909	0.4166	0.6818	0.7368	0.6667	0.7368	0.9675
L_{R-I}	$\lambda = 0.15$	100	Yes	0.5795 51	0.5813 50	0.5977 52	0.5783 48	0.5869 54	0.5909 52	0.5955 53	0.5862 51	0.5778 52	0.5842 52	0.5714 52	0.5681 50	0.5747 50	0.5977 52	0.5882 50	0.9997
			No	0.5604 51	0.6235 53	0.5632 49	0.6455 51	0.6170 58	0.5747 50	0.5227 46	0.5747 50	0.6022 53	0.5889 53	0.6818 60	0.7261 61	0.6292 56	0.4606 41	0.5465 47	0.9890
		50	Yes	0.5319	0.5909	0.5897	0.6190	0.6000	0.5813	0.5778	0.5869	0.5909	0.5744	0.6086	0.5778	0.5778	0.5909	0.6046	0.9989
			No	0.5102	0.5750	0.6341	0.6190	0.6086	0.6000	0.6304	0.5556	0.6279	0.6000	0.5333	0.6590	0.6667	0.6250	0.4250	0.9893
		25	Yes	0.5833	0.6190	0.5909	0.5652	0.6363	0.5652	0.5909	0.6086	0.5833	0.6190	0.6190	0.5909	0.5909	0.6190	0.5714	0.9987
			No	0.6000	0.6363	0.5714	0.6818	0.6086	0.5833	0.6521	0.5217	0.4000	0.7619	0.6190	0.5454	0.5238	0.6818	0.5714	0.9815
	$\lambda = 0.10$	100	Yes	0.5714 52	0.5952 50	0.6071 51	0.5903 49	0.5978 55	0.5889 53	0.5862 51	0.5977 52	0.5760 53	0.6000 54	0.5955 53	0.5977 52	0.5747 50	0.5813 50	0.5714 52	0.9996
			No	0.7386 65	0.5454 48	0.5581 48	0.6219 51	0.6086 56	0.5730 51	0.6179 55	0.5903 49	0.5227 46	0.7209 62	0.6250 55	0.5217 48	0.6279 54	0.6046 52	0.5164 47	0.9888
		50	Yes	0.6087	0.6154	0.6341	0.5435	0.5957	0.5682	0.6222	0.5111	0.6047	0.6087	0.6222	0.6279	0.5455	0.6098	0.5349	0.9959
			No	0.6222	0.7436	0.4762	0.7073	0.5417	0.6512	0.5652	0.6279	0.4318	0.6383	0.5778	0.4894	0.6136	0.6905	0.5682	0.9802
		25	Yes	0.6667	0.5909	0.5714	0.6190	0.6818	0.5455	0.6250	0.5217	0.5000	0.7143	0.4545	0.6957	0.6500	0.5238	0.5556	0.9842
			No	0.6190	0.5000	0.6190	0.5714	0.5909	0.4348	0.6364	0.4400	0.6364	0.7273	0.5238	0.7143	0.7895	0.7000	0.3500	0.9615
	$\lambda = 0.30$	100	Yes	0.5604 51	0.5697 49	0.5604 51	0.5783 48	0.5604 51	0.5681 50	0.5632 49	0.5494 50	0.5730 51	0.5568 49	0.5681 50	0.5568 49	0.5667 51	0.5632 49	0.5384 49	0.9997
			No	0.6483 59	0.6046 52	0.5662 47	0.6951 57	0.6236 58	0.5747 50	0.5681 50	0.5833 49	0.5555 50	0.5889 53	0.5340 47	0.6136 54	0.4945 45	0.5232 45	0.5617 50	0.9930
		50	Yes	0.6122	0.6341	0.5122	0.5000	0.5909	0.5349	0.5870	0.5778	0.6000	0.6087	0.5778	0.5366	0.6279	0.6190	0.5682	0.9952
			No	0.4286	0.5897	0.5854	0.4375	0.5217	0.6190	0.7021	0.6444	0.6279	0.5435	0.6818	0.5349	0.6250	0.6279	0.6429	0.9828
		25	Yes	0.6522	0.5238	0.6190	0.5909	0.6522	0.5417	0.5600	0.6190	0.6087	0.6667	0.5238	0.5652	0.6500	0.4500	0.5714	0.9901
			No	0.5909	0.6087	0.5714	0.5000	0.4545	0.4167	0.6818	0.5714	0.5833	0.4348	0.6842	0.6190	0.8000	0.7619	0.4762	0.9649



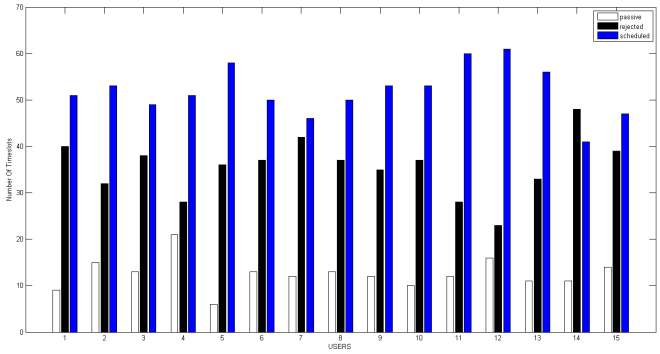
(a) BLA with VCG



(b) BLA without VCG



(c) $L_{R-I}(\lambda = 0.15)$ with VCG



(d) $L_{R-I}(\lambda = 0.15)$ without VCG

Figure 5.5: Comparison of fairness between appliances by BLA and L_{R-I} (Case 2, $T = 100$), Insufficient capacity

Case 3

In this case, the appliances have higher probability of finishing their loads earlier if the coins with the user are abundant. Due to accumulation of loads and the capacity still set to 60% of total demand request, the number of scheduled loads and rejected loads is expected to grow as time (tn) progresses. The value of P_i in equation 3.10 should increase as loads (in load matrix given in Appendix 7.1) participate in their own timeslots along with previously rejected loads.

The simulation results are presented in Table 5.6, where results for L_{R-I} method consists of λ=0.15 only. The values of ρ are expected to increase in case 3 (higher P_i) compared to case 1 and case 2 due to higher number of load demands existing within same number of T values. In compliance, the simulation results in Table 5.6 show that the ρ values for this case are comparatively higher to case 1 and case 2. The reason is that most of the loads get selected until $tn = T$ is reached. Comparison of values of (P_i) for $T = 100$ between the three cases suggests the explanation. The values for P_i in case 1 and case 2 were in the range of 50 timeslots (Table 5.4 and 5.5) whereas the values have increased to minimum of 78 timeslots in case 3 (Table 5.6). Values of P_i for some appliances are higher for scheduling without VCG but the proximity of P_i among appliances is higher while scheduling with VCG. Similarly, the trend of increasing $\mathfrak{F}$ with increase in T values also continues in case 3.

The figures from simulation results, given in Fig. 5.6 for the BLA

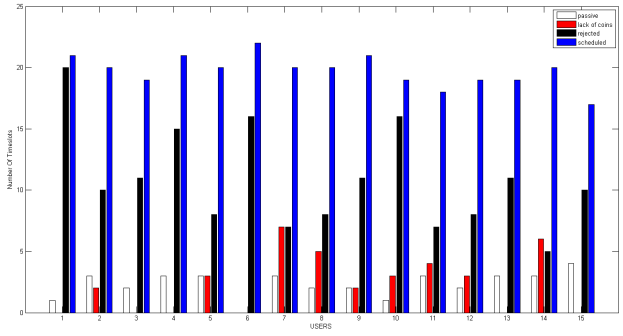
method and in Fig. 5.7 for the L_{R-I} method, are a bit unusual as compared to previous two cases. The sum of number of timeslots for selection, rejection due to coin, rejection from scheduling and passive slots surpasses T instead of being equal to T. This is because the figures are from loads' perspective, where if multiple loads occur, one can get selected and others get rejected in the same timeslot. The values for both *rejected* and *lack of coins* can increase within a single timeslot for multiple loads from same appliance. E.g., in case 1 and 2, an appliance's loads in three timeslots can be rejected three times only but in case 3 the number of rejections can be six within the same period (one load in first, two loads in second and three loads in third if all loads get rejected continuously). The timeslot values of P_i for scheduling with VCG is almost equal to scheduling without VCG. Consequently, this has led to almost equal values of $\mathfrak{F}$ for scheduling with and without VCG, although presence of VCG in scheduling has a slight edge over scheduling without VCG.

While performing the simulations, the effect of initial balance of coins $Co_i(tn)$ was noticeable in this scenario. The previous two cases of simulations consider fewer coins ($Co_i(tn)$ at $tn = 1$) while the same number of coins in this case gave results where loads were rejected more often due to lack of coins. The difference in results due to difference in initial balance of coins with each appliance is shown in Fig. 5.7 which considers L_{R-I} method. Among the first six figures that implement VCG, the first three figures (5.7a, 5.7b and 5.7c) consider fewer coins while the next three figures (Fig. 5.7d, 5.7e and 5.7f) consider comparatively more coins at $T = 1$ (500 coins). Fig. 5.7a and Fig. 5.7c are with 350 coins, while Fig. 5.7b is with 150 coins. It is much evident from Fig. 5.7b that scheduling with less coins leads to more rejection due to lack of coins, as rejection due to coins has subsequently decreased in Fig. 5.7a and Fig. 5.7c which consider higher number of coins and even higher decline in rejection is seen in figures 5.7d, 5.7e and 5.7f which consider highest number of coins. Again, the evidence that loads in case 3 get scheduled sooner than case 1 and case 2 is provided by Fig. 5.8 as well, which clearly reveals higher P_i for *scheduled* loads in case 3 (almost 20 timeslots) than cases 1 and 2 (almost 13 timeslots) within same timeslots ($T = 25$) by BLA method. The bar plot for selected users (P_i) in case 1 and case 2 given in Figures 5.8a and 5.8b have heights below 15 and 14 timeslots respectively, while appliances in case 3 in Figure 5.8c were selected for almost 20 timeslots in average.

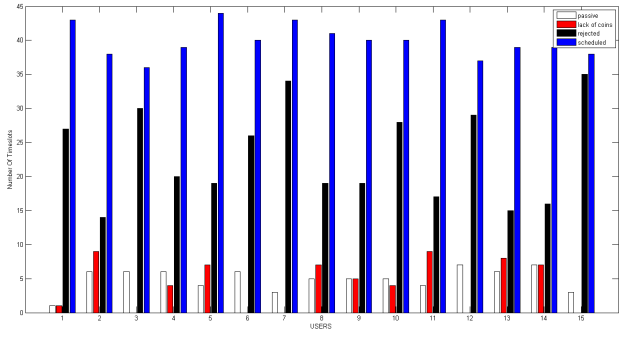
Again, L_{R-I} still holds a slight edge over BLA method. Similarly, the effect of larger T value is also the same as in former two cases.

Table 5.6: Comparison of Fairness of appliances (Case 3, Insufficient capacity)

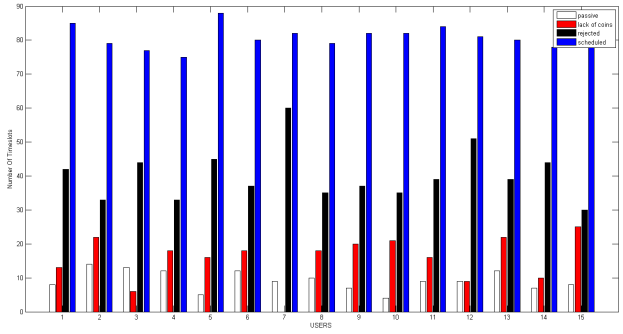
Method	T	VCG	ρ_i 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$\mathfrak{F}$
BLA	100	Yes	0.9239 85	0.9186 79	0.8851 77	0.8523 75	0.9263 88	0.9091 80	0.9011 82	0.8778 79	0.8817 82	0.8542 82	0.9231 84	0.8901 81	0.9091 80	0.8387 78	0.8804 81	0.9991
		No	0.8763 85	0.9176 78	0.8556 77	0.8824 75	0.9462 88	0.9091 80	0.8646 83	0.8764 78	0.8557 83	0.9222 83	0.9333 84	0.9302 80	0.9310 81	0.9286 78	0.9000 81	0.9989
	50	Yes	0.8958	0.9048	0.8372	0.9070	0.9149	0.9091	0.9130	0.8478	0.8696	0.8723	0.8936	0.8667	0.9070	0.8511	0.8837	0.9992
		No	0.9333	0.8837	0.7500	0.8667	0.9556	0.9535	0.9348	0.9111	0.8511	0.9111	0.9149	0.9048	0.9070	0.8636	0.8837	0.9970
	25	Yes	0.5128	0.6452	0.6129	0.6563	0.6563	0.5789	0.6471	0.5714	0.6250	0.6786	0.5000	0.5588	0.6129	0.5882	0.5517	0.9925
		No	0.5833	0.5405	0.5455	0.5676	0.6897	0.4118	0.6286	0.6000	0.6774	0.5135	0.7083	0.6897	0.7917	0.5556	0.6667	0.9779
L_{R-I} $\lambda = 0.15$	100	Yes	0.8660 84	0.8778 79	0.8261 76	0.8605 74	0.8969 87	0.8511 80	0.8542 82	0.8495 79	0.8469 83	0.8723 82	0.8737 83	0.8681 79	0.8617 81	0.8370 77	0.8723 82	0.9996
		No	0.8660 84	0.8478 78	0.8462 77	0.8810 74	0.8866 86	0.8791 80	0.8723 82	0.9080 79	0.8913 82	0.8333 80	0.8557 83	0.8901 81	0.8817 82	0.8556 77	0.8632 82	0.9995
	50	Yes	0.9149	0.8837	0.7609	0.8667	0.8980	0.8333	0.8750	0.8542	0.8333	0.8163	0.8333	0.7660	0.8864	0.8667	0.8667	0.9975
		No	0.8936	0.8810	0.7826	0.9512	0.9149	0.8889	0.9149	0.8125	0.8696	0.9333	0.8936	0.8864	0.8444	0.8511	0.8409	0.9975
	25	Yes	0.5938	0.5714	0.6667	0.5526	0.5000	0.6471	0.5641	0.6563	0.6471	0.4750	0.6667	0.6333	0.5938	0.6061	0.5769	0.9911
		No	0.4545	0.7143	0.4595	0.6176	0.4444	0.5238	0.6176	0.5714	0.5500	0.7143	0.7308	0.6061	0.6552	0.5135	0.6071	0.9762



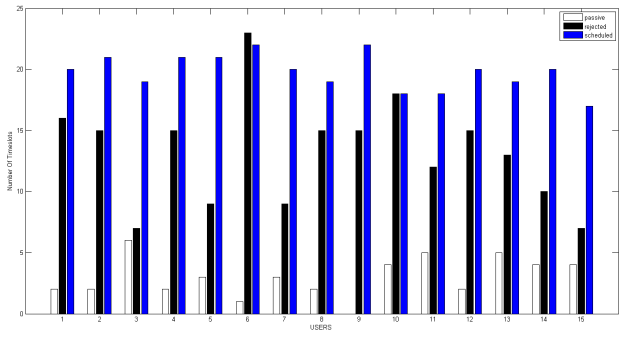
(a) BLA with VCG, T=25



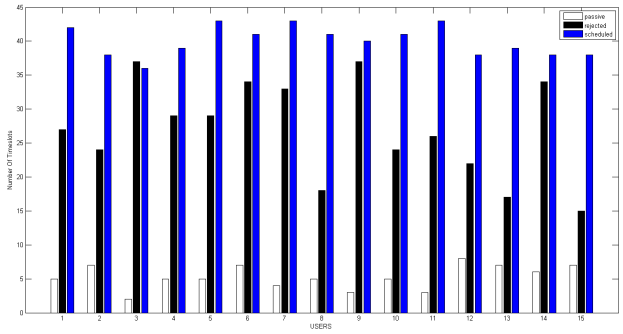
(b) BLA with VCG, T=50



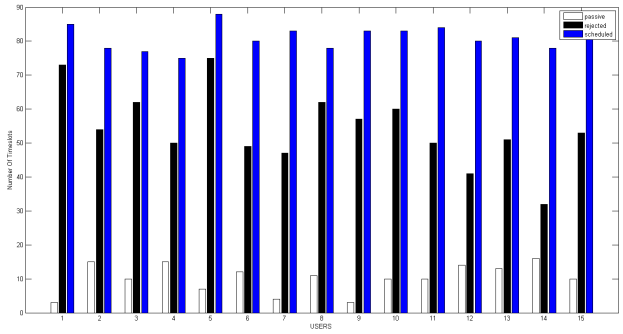
(c) BLA with VCG, T=100



(d) BLA without VCG, T=25

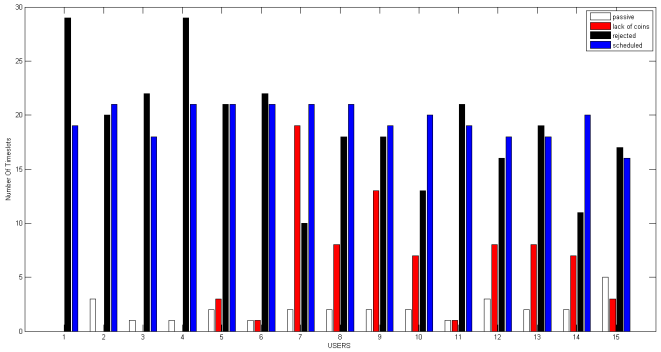


(e) BLA without VCG, T=50

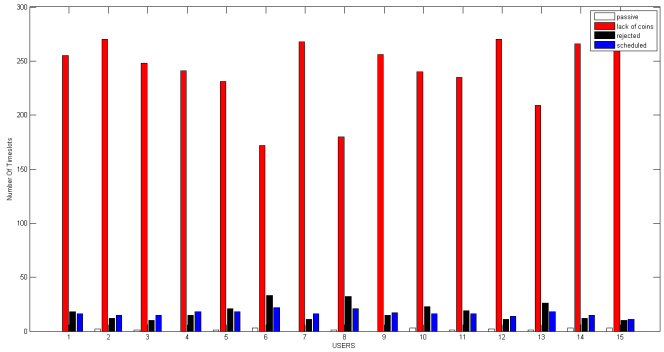


(f) BLA without VCG, T=100

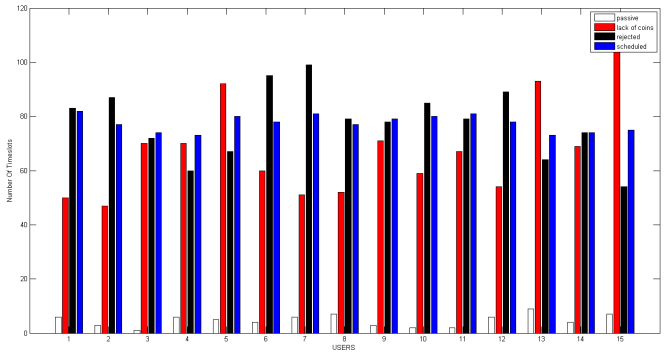
Figure 5.6: Comparison of fairness by BLA - with and without VCG (Case 3)



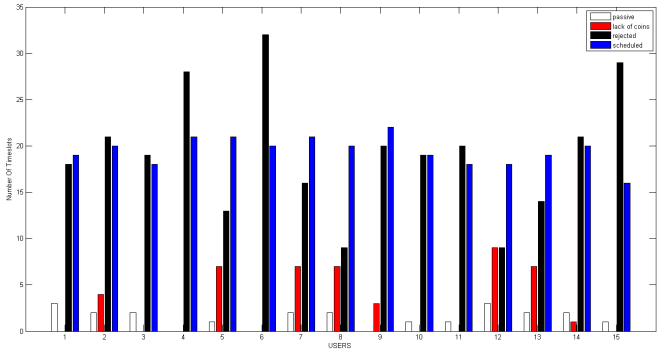
(a) L_{R-I} with VCG (fewer coins), T=25



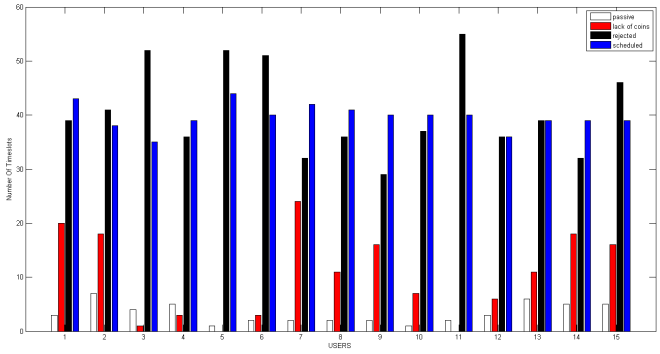
(b) L_{R-I} with VCG (fewer coins), T=50



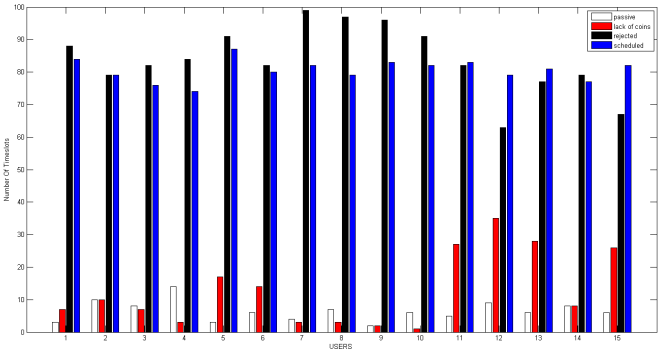
(c) L_{R-I} with VCG (fewer coins), T=100



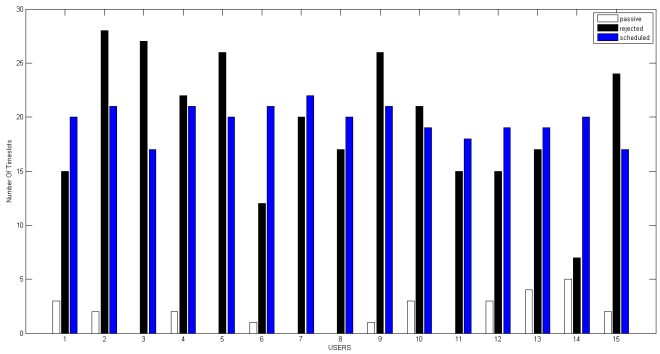
(d) L_{R-I} with VCG (more coins), T=25



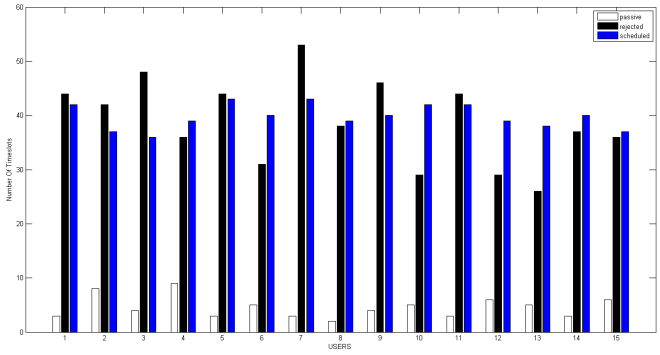
(e) L_{R-I} with VCG (more coins), T=50



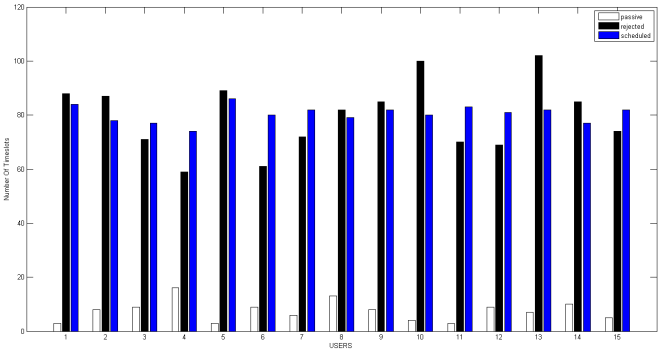
(f) L_{R-I} with VCG (more coins), T=100



(g) L_{R-I} without VCG, T=25

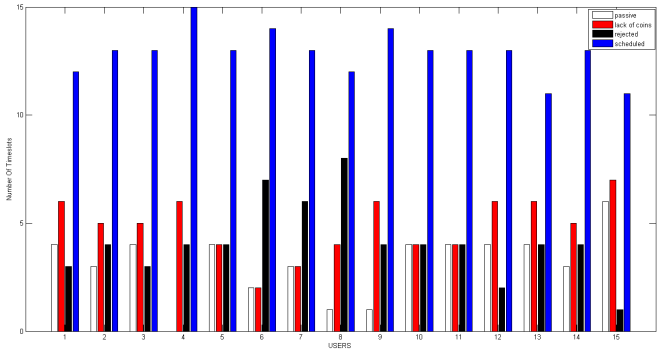


(h) L_{R-I} without VCG, T=50

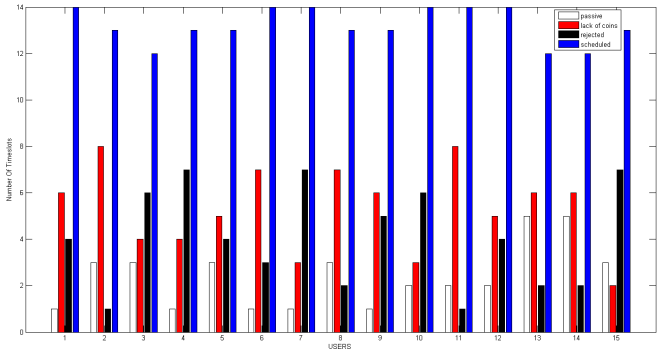


(i) L_{R-I} without VCG, T=100

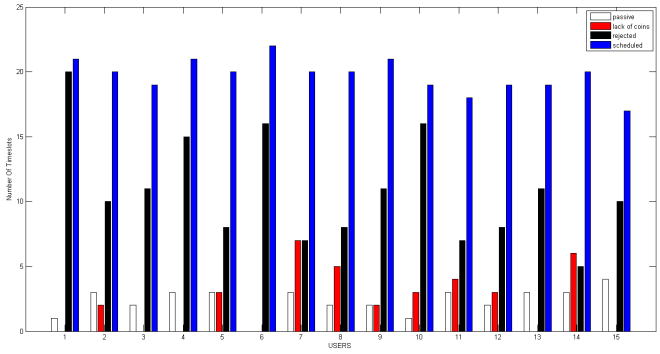
Figure 5.7: Comparison of fairness by L_{R-I} - with and without VCG (Case 3)



(a) BLA with VCG, Case 1



(b) BLA with VCG, Case 2



(c) BLA with VCG, Case 3

Figure 5.8: Comparison of frequency of selection, (by $BLA, T = 25$)

5.3.2 Sufficient Capacity

In this scenario, simulations were conducted to emulate the situation wherein the capacity C_{tn} varies randomly in each timeslot but the condition $\sum_t \sum_i L_{i,t} \leq C$ holds true at the end when $tn = T$. Again, the variable capacities set for each timeslot is given in Appendix 7.2. The simulation results consist of 25 timeslots ($T = 25$) only, with same number of users ($N = 15$) continued as in previous subsections. The simulation results for this scenario are presented in Tables 5.7, 5.8 and 5.9 considering case 1, case 2 and case 3 as in previous sub-section. But $\lambda = 0.15$ only is taken into consideration for the L_{R-I} method. Fig. 5.9 and Fig 7.1 show the simulation results for BLA and L_{R-I} methods respectively.

From the simulation results in Tables 5.7, 5.8 and 5.9, it can be clearly observed that the results obtained from condition $\sum_t \sum_i L_{i,t} \leq C$ are better than the condition of insufficient capacity. The application of VCG has increased the fairness between appliances, which can be noticed from the results in tables and figures as well. Even though the capacity is higher in almost every timeslot compared to insufficient capacity scenario, there is not much change in results for case 1 and case 2. In these cases, the total demand in contention for power is still the same as in previous section of insufficient capacity. The capacity has increased means more loads need to be available for scheduling. Otherwise the capacity goes wasted. More loads contend for power in every timeslot in case 3, so the individual ρ_i and P_i values have increased in case 3. There has been slender increase in values of ρ_i and P_i in case 1 and 2 as well. The increase in individual ρ values than in insufficient capacity scenario is good from users' perspective and high $\mathfrak{F}$ is still obtained, which is good from SG operator's perspective. Within the scenario of sufficient capacity, all three cases produce results where the individual P_i values are closer to each other when VCG is applied. The values are quite large for some users while some others have quite smaller values when VCG is not considered in scheduling, for both L_{R-I} and BLA. The difference in results from the two scheduling methods is still not much and both are equally successful in all three cases of sufficient capacity. Again, it can be noticed from the figures for case 3 that the height of bars for *rejected* timeslots is much higher although the height for *selected* timeslots has not decreased. The explanation is again due to rejection of multiple loads within single timeslot.

Table 5.7: Comparison of Fairness of appliances (Case 1 with $\sum_t \sum_i L_{i,t} \leq C$)

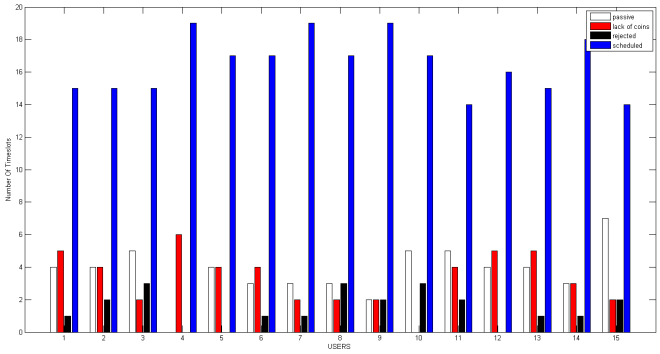
Method	T	VCG	'ρ_i' and 'P_i'															$\mathfrak{F}$
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
BLA	25	Yes	0.7619 16	0.7143 15	0.7000 14	0.7600 19	0.7619 16	0.7727 17	0.7727 17	0.8095 17	0.7500 18	0.7000 14	0.7500 15	0.7619 16	0.8095 17	0.7727 17	0.7222 13	0.9981
		No	0.8571 18	0.9524 20	0.8000 16	0.9130 21	0.8571 18	0.7727 17	1.0000 22	0.8571 18	1.0000 22	0.7500 15	0.9474 18	1.0000 20	0.8095 17	0.8182 18	1.0000 17	0.9906
L_{R-I} ($\lambda = 0.15$)	25	Yes	0.8571 18	0.7143 15	0.7500 15	0.8261 19	0.7619 16	0.7727 17	0.8182 18	0.8095 17	0.7500 18	0.8500 17	0.7000 14	0.7619 16	0.7619 16	0.7273 16	0.7778 14	0.9965
		No	0.9048 19	0.9048 19	0.8947 17	0.8261 19	0.8571 18	0.8182 18	0.9091 20	0.9048 19	0.9091 20	0.9500 19	0.7500 15	0.8500 17	0.8571 18	0.8571 18	1.0000 17	0.9958

Table 5.8: Comparison of Fairness of appliances (Case 2 with $\sum_t \sum_i L_{i,t} \leq C$)

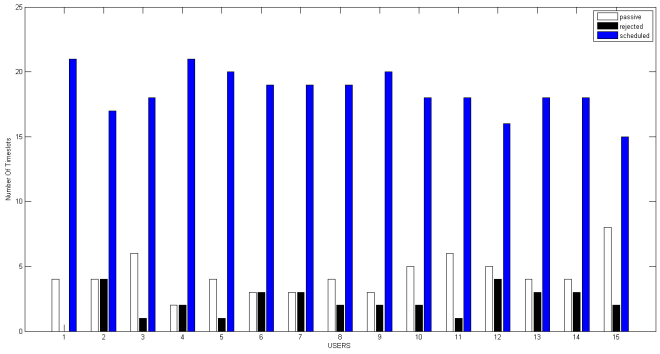
Method	T	VCG	'ρ_i' and 'P_i'															$\mathfrak{F}$
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
BLA	25	Yes	0.7273 16	0.7143 15	0.7368 14	0.7391 17	0.8095 17	0.6667 16	0.7826 18	0.6957 16	0.7727 17	0.7143 15	0.7619 16	0.8182 18	0.7895 15	0.7500 15	0.7647 13	0.9970
		No	0.9048 19	0.9048 19	0.8947 17	0.9524 20	0.7619 16	0.8636 19	0.9545 21	0.9091 20	0.9130 21	1.0000 20	0.9474 18	0.9000 18	1.0000 19	0.7500 15	0.7778 14	0.9929
L_{R-I} ($\lambda = 0.15$)	25	Yes	0.7619 16	0.7619 16	0.7000 14	0.7273 16	0.8571 18	0.7727 17	0.8636 19	0.7619 16	0.7826 18	0.7000 14	0.7143 15	0.7273 16	0.7619 16	0.7143 15	0.7778 14	0.9960
		No	0.9524 20	0.9524 20	0.8500 17	0.8636 19	0.9048 19	0.8261 19	0.8696 20	0.8636 19	0.9545 21	0.8571 18	0.9474 18	1.0000 20	0.7895 15	0.9000 18	0.7222 13	0.9937

Table 5.9: Comparison of Fairness of appliances (Case 3 with $\sum_t \sum_i L_{i,t} \leq C$)

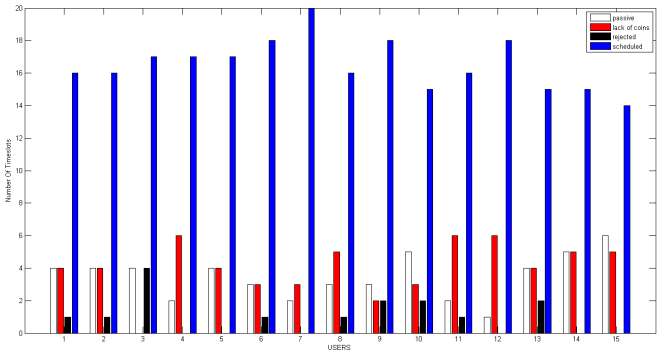
Method	T	VCG	'ρ_i' and 'P_i'															$\mathfrak{F}$
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
BLA	25	Yes	0.6786 19	0.5882 20	0.6786 19	0.7000 21	0.7037 19	0.7692 20	0.7500 21	0.7143 20	0.6452 20	0.7407 20	0.6207 18	0.5667 17	0.5294 18	0.7407 20	0.6800 17	0.9898
		No	0.4048 17	0.8400 21	0.7600 19	0.6250 20	0.6364 21	0.5714 20	0.6250 20	0.7500 21	0.8148 22	0.7500 18	0.7200 18	0.8261 19	0.6207 18	0.6333 19	0.7619 16	0.9741
L_{R-I} ($\lambda = 0.15$)	25	Yes	0.8400 21	0.7037 19	0.6923 18	0.7500 21	0.7143 20	0.7778 21	0.5405 20	0.7241 21	0.7241 21	0.6207 18	0.5667 17	0.7600 19	0.6333 19	0.5000 19	0.7500 15	0.9824
		No	0.7000 21	0.8000 20	0.5152 17	0.6250 20	0.7692 20	0.6774 21	0.7143 20	0.5938 19	0.8400 21	0.5758 19	0.4615 18	0.6552 19	0.5429 19	0.6452 20	0.5357 15	0.9738



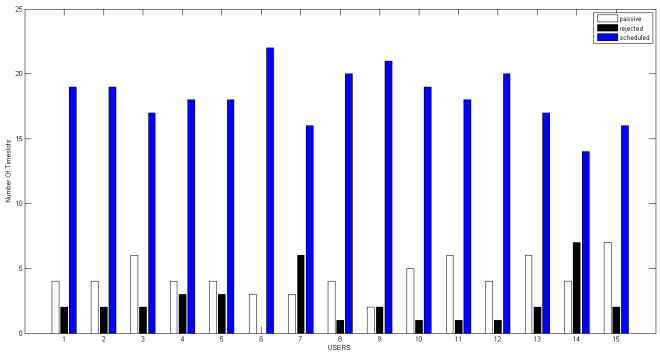
(a) *BLA* with VCG, Case 1



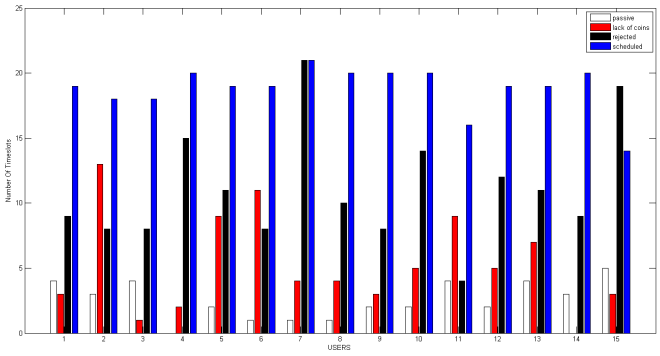
(b) *BLA* without VCG, Case 1



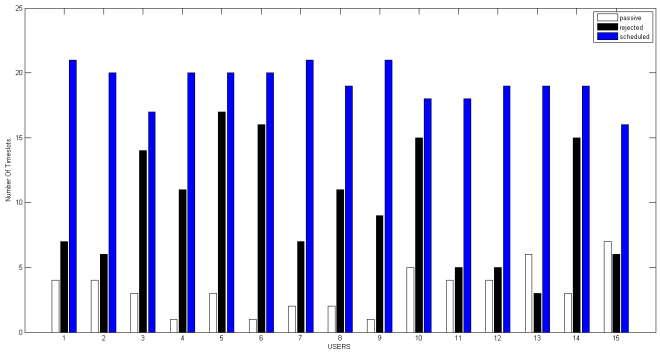
(c) *BLA* with VCG, Case 2



(d) BLA without VCG, Case 2



(e) BLA with VCG, Case 3



(f) BLA without VCG, Case 3

Figure 5.9: Comparison of Fairness by BLA ($T = 25$, Sufficient Capacity)

Chapter 6

Discussions

In this thesis, appliances have been categorized based on whether they can undergo scheduling or not. This protects the important works of customers, while the necessity of DM is fulfilled through scheduling of appliances who have tolerance for delay. The load categorization can be done by considering any other aspect of categorization but the algorithms will still be useful. The efficiency of the implemented algorithms is seen in the results. Both BLA and L_{R-I} have been equally successful in terms of both accuracy and fairness. And the introduction of VCG to be incorporated with LA based scheduling has helped to significantly increase the fairness of SG system. Other algorithms can be also tested for scheduling and be compared with BLA and L_{R-I}'s results obtained in this thesis.

While the two LA methods succeed in terms of accuracy of optimization, the main issue is the trade-off between fairness and accuracy after implementation of VCG. This can be addressed by allowing a variable rate of coins R_{tn} for each timeslot and/or proper allocation of initial stock of coins $Co_i(tn)$ at $tn = 1$ for each user. These constraints play a tricky role in fairness and proper handling of their values can help reduce and hopefully eliminate the trade-off. Also, in those timeslots where capacity is larger than the total of demands or each demand is greater than the capacity, the scheduling should be avoided to save time of computation, as done in this thesis work.

The existence of many rejected users, shown by very high demand compared to capacity at the end of T timeslots ($tn = 25$) in scenarios of sufficient capacity (evident in figures for measurement of accuracy in case 3), means that having sufficient capacity does not necessarily lead to selection of all users. The granularity of loads can be improved, so the results are expected

to be much better than shown in this thesis. Moreover, case 3 which deals with multiple loads from the same appliance within the same time frame can be visualized as scenario of multiple appliances from the same customer (where appliances have single load only). So the simulation results for case 3 can be used to compare fairness between customers of SG as well.

The effect in fairness due to difference in the number of demands between different appliances is not studied in this thesis. Some appliances which send request once in a while compared with those which send request frequently can have different impact on maintaining fairness level. Simulations can be done by creating such scenario to observe the results and suggestions can be made if fairness between those appliances turns out to be an issue. In addition, when there is excess capacity available, customers should be encouraged to send loads at those timeslots to make full utilization of the capacity. This requires the R_{tn} value to be dropped. As R_{tn} is fixed in the simulations in this thesis, the alternative was to allocate more coins to users at tn=1 when more loads from each appliance are expected, as previously discussed. But the stock of coins must not be so high that the effect of VCG is neutralized. High stock of coins with users means all users have enough coins to enter into scheduling in every timeslot.

In this thesis work, all loads from an appliance are rejected together due to proportional distribution of coins among the loads. Scheduling can be done by sending only those loads as the stock of coins permit to avoid complete rejection of any appliance. E.g., when $R_{tn} = 1$ and two loads of 20 units each from an appliance with 30 coins exist, one of these two loads can be sent into scheduling process instead of rejecting both loads.

Chapter 7

Conclusions

The main conclusion of this thesis work is that the optimization by both BLA and L_{R-I} methods can be performed with high accuracy and the introduction of VCG for improvement of fairness succeeds in its purpose without much adverse effect in accuracy. VCG can be overlooked when T is too large, since both kinds of scheduling, LA with and without VCG produce similar $\mathfrak{F}$. The difference in results in terms of accuracy of optimization under different implementation scenarios (cases 1, 2 and 3) and VCG (with or without) is very marginal, indicating BLA and L_{R-I}'s worth in optimization of electricity usage. To avoid the accumulation of demands, proper selection of intial stock of coins with each user and/or application of variable rate of coins R_{tn} in scheduling with VCG should be carried out.

Bibliography

[1] Farhangi H. (2010). The path of the smart grid. *IEEE power and energy magazine, 8*(1), 18-28.

[2] Gungor V. C., Lu B. and Hancke G. P. (2010). Opportunities and challenges of wireless sensor networks in smart grid. *IEEE transactions on industrial electronics, 57*(10), 3557-3564.

[3] Gungor V. C., Sahin D., Kocak T., Ergut S., Buccella C., Cecati C. and Hancke G. P. (2011). Smart grid technologies: communication technologies and standards. *IEEE transactions on Industrial informatics, 7*(4), 529-539.

[4] Chen K. C., Yeh P. C., Hsieh H. Y. and Chang S. C. (2010, March). Communication infrastructure of smart grid. In *Communications, Control and Signal Processing (ISCCSP), 2010 4th International Symposium on* (pp. 1-5). IEEE.

[5] Moslehi K. and Kumar R. (2010, January). Smart grid-a reliability perspective. In *Innovative Smart Grid Technologies (ISGT), 2010* (pp. 1-8). IEEE.

[6] Molderink A., Bakker V., Bosman M. G., Hurink J. L. and Smit G. J. (2010). Management and control of domestic smart grid technology. *IEEE transactions on Smart Grid, 1*(2), 109-119.

[7] Xhafa F. and Abraham, A. (2010). Computational models and heuristic methods for Grid scheduling problems. *Future generation computer systems, 26*(4), 608-621.

[8] Ibars C., Navarro M. and Giupponi L. (2010, October). Distributed demand management in smart grid with a congestion game. In *Smart grid communications (SmartGridComm), 2010 first IEEE international conference on* (pp. 495-500). IEEE.

[9] Mets K., Verschueren T., Haerick W., Develder C. and De Turck F. (2010, April). Optimizing smart energy control strategies for plug-in hybrid electric vehicle charging. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP* (pp. 293-299). Ieee.

[10] Yu R., Zhang Y., Gjessing S., Yuen C., Xie S. and Guizani M. (2011). Cognitive radio based hierarchical communications infrastructure for smart grid. *IEEE network, 25*(5), 6-14.

[11] Liang X., Li X., Lu R., Lin X. and Shen X. (2013). UDP: Usage-based dynamic pricing with privacy preservation for smart grid. *IEEE Transactions on Smart Grid, 4*(1), 141-150.

[12] Erol-Kantarci M. and Hussein T. M. (2010, October). Prediction-based charging of PHEVs from the smart grid with dynamic pricing. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on* (pp. 1032-1039). IEEE.

[13] Samadi P., Mohsenian-Rad A. H., Schober R., Wong V. W. and Jatskevich J. (2010, October). Optimal real-time pricing algorithm based on utility maximization for smart grid. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on* (pp. 415-420). IEEE.

[14] Logenthiran T., Srinivasan D. and Shun T. Z. (2012). Demand side management in smart grid using heuristic optimization. *IEEE Transactions on Smart Grid, 3*(3), 1244-1252.

[15] Mohsenian-Rad A. H., Wong V. W., Jatskevich J., Schober R. and Leon-Garcia A. (2010). Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *IEEE transactions on Smart Grid, 1*(3), 320-331.

[16] Pedrasa M. A. A., Spooner T. D. and MacGill I. F. (2010). Coordinated scheduling of residential distributed energy resources to optimize smart home energy services. *IEEE Transactions on Smart Grid, 1*(2), 134-143.

[17] Saber A. Y. and Venayagamoorthy G. K. (2012). Resource scheduling under uncertainty in a smart grid with renewables and plug-in vehicles. *IEEE Systems Journal, 6*(1), 103-109.

[18] Subrata R. and Zomaya A. Y. (2008). Game-theoretic approach for load balancing in computational grids. *IEEE Transactions on Parallel and Distributed Systems, 19*(1), 66-76.

[19] Thapa R. (2015). Shiftable Power Load Scheduling: A Learning Automata Based Scheme Using Bayesian Learning. Submitted to *Department of ICT, University of Agder, Grimstad, Norway.*

[20] Zhu Z., Tang J., Lambotharan S., Chin W. H. and Fan Z. (2012, January). An integer linear programming based optimization for home demand-side management in smart grid. In *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)* (pp. 1-5). IEEE.

[21] Bayram I. S. and Ustun T. S. (2016). A survey on behind the meter energy management systems in smart grid. *Renewable and Sustainable Energy Reviews.*

[22] Costanzo G. T., Zhu G., Anjos M. F. and Savard G. (2012). A system architecture for autonomous demand side load management in smart buildings. *IEEE Transactions on Smart Grid, 3*(4), 2157-2165.

[23] Zhu Z., Lambotharan S., Chin W. H. and Fan Z. (2012). Overview of demand management in smart grid and enabling wireless communication technologies. *IEEE Wireless Communications, 19*(3), 48-56.

[24] Hoven N., Tandra R. and Sahai A. (2005). Some fundamental limits on cognitive radio. *Wireless Foundations EECS, Univ. of California, Berkeley.*

[25] Thathachar M. A. and Sastry P. S. (2002). Varieties of learning automata: an overview. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 32*(6), 711-722.

[26] Najim K. and Poznyak A. S. (2014). *Learning automata: theory and applications.* Elsevier.

[27] Misra S., Krishna P. V., Saritha V. and Obaidat M. S. (2013). Learning automata as a utility for power management in smart grids. *IEEE Communications Magazine, 51*(1), 98-104.

[28] Ali S. Q., T. Parambath I. A. and Malik N. H. (2013). Learning automata algorithms for load scheduling. *Electric Power Components and Systems, 41*(3), 286-303.

[29] Misra S., Krishna P. V., Saritha V., Agarwal H. and Ahuja A. (2014). Learning automata-based multi-constrained fault-tolerance approach for effective energy management in smart grid communication network. *Journal of Network and Computer Applications, 44*, 212-219.

[30] Shi W. and Wong V. W. (2011, October). Real-time vehicle-to-grid control algorithm under price uncertainty. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on* (pp. 261-266). IEEE.

[31] Williams R. J. and Zipser D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation, 1*(2), 270-280.

[32] Mitchell T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill, 45*, 37.

[33] Granmo O.C. and Glimsdal S. (2013). Accelerated Bayesian learning for decentralized two-armed bandit based decision making with applications to the Goore game. *Applied Intelligence, 38*(4):479-488.

[34] Narendra K. S. and Thathachar M. A. L. (2012). *Learning automata: an introduction*. Courier Corporation.

[35] Saad W., Han Z., Poor H. V. and Basar T. (2012). Game-theoretic methods for the smart grid: An overview of microgrid systems, demand-side management, and smart grid communications. *IEEE Signal Processing Magazine, 29*(5), 86-105.

[36] Basar T. and Olsder G. J. (1999). *Dynamic noncooperative game theory* (Vol. 23). Siam.

[37] Baharlouei Z., Hashemi M., Narimani H. and Mohsenian-Rad H. (2013). Achieving optimality and fairness in autonomous demand response: Benchmarks and billing mechanisms. *IEEE Transactions on Smart Grid, 4*(2), 968-975.

[38] Thomas J. Watson IBM Research Center. Research Division and Jaffe J. M. (1980). *A Decentralized, "optimal", Multiple-user Flow Control Algorithm*.

[39] Chen Y. W., Chen X. and Maxemchuk N. (2012). The fair allocation of power to air conditioners on a smart grid. *IEEE Transactions on Smart Grid, 3*(4), 2188-2195.

[40] Zhang Y., Zeng P. and Zang C. (2015, June). Optimization algorithm for home energy management system based on artificial bee colony in smart grid. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on* (pp. 734-740). IEEE.

[41] Koutitas G. (2012). Control of flexible smart devices in the smart grid. *IEEE Transactions on Smart Grid, 3*(3), 1333-1343.

[42] Pisinger D. (1995). A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research, 83*(2), 394-410.

[43] Murty K. G. and Kabadi S. N. (1987). Some NP-complete problems in quadratic and nonlinear programming. *Mathematical programming, 39*(2), 117-129.

[44] Lagarias J. C. and Odlyzko A. M. (1985). Solving low-density subset sum problems. *Journal of the ACM (JACM), 32*(1), 229-246.

[45] Coster M. J., LaMacchia B. A., Odlyzko A. M. and Schnorr C. P. (1991, April). An improved low-density subset sum algorithm. In *Workshop on the Theory and Application of of Cryptographic Techniques* (pp. 54-67). Springer Berlin Heidelberg.

[46] Radziszowski S. and Kreher D. (1988). Solving subset sum problems with the Lˆ3 algorithm. *The Charles Babbage Research Centre: The Journal of Combinatorial Mathematics and Combinatorial Computing, 3*.

[47] Sahni S. (1975). Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM (JACM), 22*(1), 115-124.

[48] Jain R., Chiu D. M. and Hawe W. R. (1984). *A quantitative measure of fairness and discrimination for resource allocation in shared computer system* (Vol. 38). Hudson, MA: Eastern Research Laboratory, Digital Equipment Corporation.

[49] Freiman G. A. (1993). New analytical results in subset-sum problem. *Discrete mathematics, 114*(1), 205-217.

[50] Xiong G., Chen C., Kishore S. and Yener A. (2011, January). Smart (in-home) power scheduling for demand response on the smart grid. In

Innovative smart grid technologies (ISGT), 2011 IEEE PES (pp. 1-7). IEEE.

[51] Chalmers M., Bell M., Brown B., Hall M., Sherwood S. and Tennent P. (2005, June). Gaming on the edge: using seams in ubicomp games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology* (pp. 306-309). ACM.

[52] Togelius J., Karakovskiy S., Koutnk J. and Schmidhuber J. (2009, September). Super mario evolution. In *2009 IEEE Symposium on Computational Intelligence and Games* (pp. 156-161). IEEE.

[53] Narendra K. S. and Thathachar M. A. (1974). Learning automata-a survey. *IEEE Transactions on systems, man, and cybernetics*, (4), 323-334.

Appendix

7.1 The Load demand matrix

The loads for each user against each timeslot is given by the following matrix where 15 columns represent the 15 users and the 100 rows are for the 100 timeslots. For T=25 and T=50, first 25 and 50 rows of the matrix were selected.

7.2 The Variable capacity vector

Capacity Vector=[810, 896, 790, 1063, 1237, 1124, 986, 713, 1144, 1216, 1306, 749, 666, 845, 1089, 845, 1150, 660, 667, 707, 690, 844, 858, 1072, 794].

7.3 Additional Figures for Comparison of Fairness and Accuracy

The figures for comparison of fairness between different cases while scheduling by L_{R-I} with sufficient capacity are given in Fig. 7.1. Figures for scheduling by BLA are already presented in subsection 5.3.2, so comparison can also be made between scheduling by these two methods.

The figures while measuring the accuracy by BLA and L_{R-I} for $T = 25$ for case 2 and case 3 with insufficient capacity are given in Fig. 7.2 and 7.3 respectively. The figures for case 1 are already presented in subsection 5.2.1.

In continuation to subsection 5.2.2, the figures for measurement of accuracy while scheduling by BLA and L_{R-I} methods for case 2 and case 3 with sufficient capacities are given in Fig. 7.4 and Fig. 7.5 respectively.

Table 7.1: The demands for each appliances at different timeslots

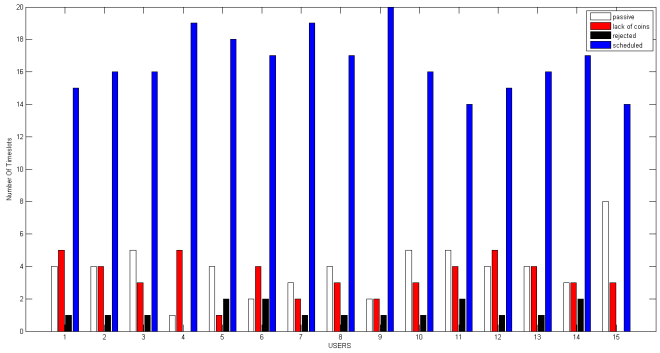
Appliances (i)														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
85	0	83	77	77	70	72	89	97	78	88	70	0	0	91
71	0	66	58	71	59	90	65	88	0	76	79	94	0	55
74	68	53	85	0	55	53	95	0	69	72	61	81	58	0
84	73	0	69	63	89	88	69	62	99	71	94	54	95	0
52	96	86	52	58	81	81	0	51	90	62	81	0	76	0
75	70	0	99	74	53	0	96	61	56	71	67	78	0	59
77	93	89	68	82	60	50	99	91	0	67	0	91	69	0
0	82	87	59	70	50	52	83	69	68	70	64	0	74	0
67	65	74	77	53	62	87	55	53	54	0	0	94	66	62
74	66	0	89	90	0	99	82	85	0	87	96	87	63	70
0	56	62	88	0	61	89	67	55	54	0	93	99	63	89
0	0	60	85	62	96	92	59	89	65	68	65	0	56	87
92	52	86	68	0	91	55	51	87	95	0	95	65	90	0
0	94	50	97	55	0	0	63	91	57	85	78	93	65	54
99	90	0	81	87	64	82	89	56	75	71	0	61	80	83
75	63	97	52	65	74	0	0	64	0	77	79	100	97	92
69	99	73	54	0	69	92	58	88	71	0	0	55	76	78
61	96	0	68	90	89	58	66	98	0	0	62	56	54	91
98	0	52	80	66	91	95	0	59	61	61	89	55	86	0
74	94	73	0	65	0	91	60	0	61	87	64	68	88	95
75	97	78	0	54	58	68	0	68	84	84	0	89	98	82
62	53	73	68	72	86	91	63	58	66	0	73	100	0	0
76	66	0	54	78	100	99	78	0	78	55	95	100	63	66
71	64	92	0	52	78	65	51	60	60	84	52	0	93	85
89	57	50	0	77	99	83	76	62	55	73	59	0	0	50
63	92	0	0	78	0	75	86	68	95	79	77	51	50	54
97	97	72	53	95	0	71	0	95	0	76	96	84	95	66
80	71	63	57	76	96	77	87	0	87	96	65	0	96	55
99	54	88	93	0	71	98	62	66	77	86	0	62	0	98
67	80	72	53	89	0	70	54	0	90	85	0	90	59	87
0	0	69	50	60	95	56	0	73	62	82	77	57	67	64
56	0	100	70	95	52	74	84	93	76	52	0	94	57	0
0	78	51	72	82	74	96	0	0	91	87	96	91	87	97
78	57	0	91	74	0	61	99	53	72	53	0	78	85	84
66	77	0	55	0	57	63	0	93	64	63	68	87	72	82
77	99	72	0	90	72	87	82	82	0	72	86	61	96	72

Table: The demands for each appliances at different timeslots (continued)

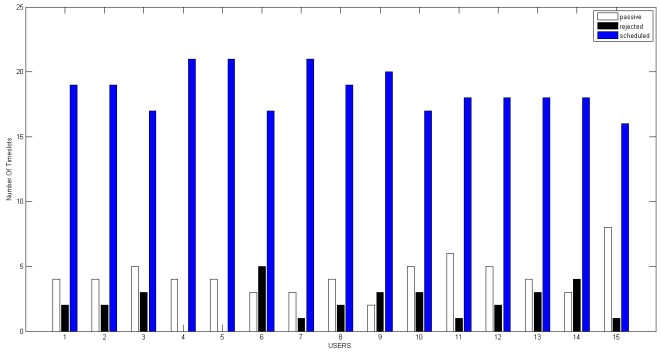
73	97	95	63	85	82	0	71	67	51	0	0	92	75	76
63	0	99	0	74	61	99	84	80	70	73	86	75	0	57
89	91	0	57	92	92	85	63	0	91	74	60	0	79	69
50	0	59	64	53	60	0	86	56	77	94	60	77	67	0
0	0	54	89	54	84	90	91	82	85	78	90	66	63	0
50	60	76	99	92	55	100	91	51	58	88	0	0	0	99
59	0	64	0	81	0	87	77	80	82	79	84	67	84	66
58	78	61	61	81	82	0	73	81	65	80	83	0	0	83
66	71	0	0	98	83	57	85	89	79	78	93	69	77	52
53	63	0	74	97	55	79	66	0	80	50	57	64	0	61
100	52	0	53	81	92	86	60	0	71	92	100	0	74	73
53	55	76	56	93	0	75	0	64	0	59	52	51	100	72
88	0	84	0	77	65	0	70	67	56	57	99	77	78	93
79	0	0	0	51	80	71	50	53	76	54	58	93	68	66
0	95	54	56	85	79	0	100	53	63	81	83	58	0	92
93	61	75	73	84	60	90	0	63	69	88	89	95	62	0
75	80	90	90	65	0	57	76	93	89	79	65	92	0	58
88	0	90	80	81	0	58	54	52	99	68	0	70	61	78
99	0	82	0	91	57	0	98	98	69	81	76	100	72	84
90	0	83	56	50	61	50	78	94	85	58	77	62	94	0
0	88	87	0	71	56	83	59	0	98	61	91	57	91	64
0	68	73	84	91	0	59	87	67	87	81	60	0	78	75
0	93	0	96	59	70	50	77	72	0	98	98	86	97	65
84	78	100	58	68	51	0	98	100	0	0	55	95	91	92
96	88	52	85	0	82	55	80	0	87	84	0	54	62	75
90	71	0	95	69	83	70	68	57	93	0	98	91	52	98
56	79	54	77	64	0	96	84	53	0	93	0	92	83	88
92	0	85	97	56	62	79	71	62	76	74	0	70	0	50
66	81	93	68	83	82	0	0	99	100	82	57	67	64	77
58	62	84	100	0	79	78	0	98	81	75	61	83	55	58
65	96	95	64	68	100	54	89	66	0	82	0	0	76	55
89	71	74	50	83	0	81	0	90	94	78	61	0	66	60
50	56	58	81	92	0	62	58	65	50	82	73	69	0	0
94	97	97	56	88	95	64	80	0	0	67	53	90	91	83
61	69	65	0	0	55	77	0	88	73	58	73	55	78	97
100	56	0	0	72	52	81	85	97	93	63	83	0	88	63
60	66	0	0	63	50	64	77	92	59	50	53	91	63	64
64	97	90	97	0	90	0	62	70	85	51	94	90	0	100
62	52	0	62	77	87	0	80	53	91	55	57	95	0	76

Table: The demands for each appliances at different timeslots (continued)

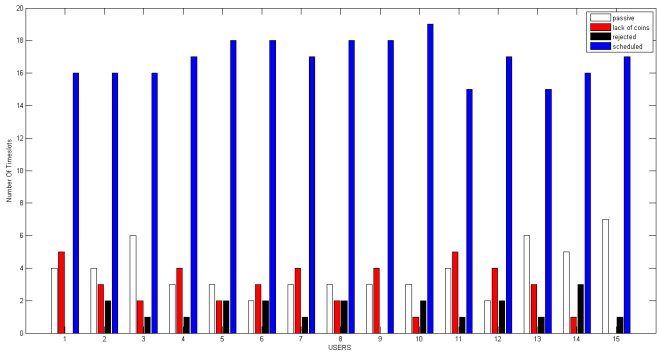
75	0	65	99	57	82	0	73	79	68	66	85	89	86	0
80	89	75	0	69	79	0	81	0	65	79	99	59	71	62
53	87	53	0	64	88	60	52	83	0	0	92	71	98	70
96	0	80	0	0	64	91	85	54	82	96	70	56	85	68
82	87	75	59	0	86	0	0	85	76	93	98	80	97	63
72	0	61	0	88	91	89	69	79	84	91	50	0	63	83
0	73	67	69	79	98	62	87	77	54	0	60	83	0	63
97	70	52	0	58	58	70	0	0	63	66	74	78	74	90
83	61	93	94	100	81	0	90	0	58	88	63	94	67	0
50	83	82	0	87	55	98	0	75	78	54	59	56	78	60
88	75	85	81	74	0	79	84	74	0	0	62	57	79	89
0	81	90	92	87	0	54	69	95	98	76	53	69	80	0
52	70	61	95	64	79	70	58	68	73	50	0	80	0	93
53	0	52	0	59	0	66	87	53	60	72	68	94	97	100
67	81	0	54	63	64	61	60	57	77	0	97	50	71	51
58	0	50	94	73	84	85	0	86	0	58	78	64	56	79
96	88	80	62	79	58	87	80	99	81	0	0	68	0	84
86	61	65	77	88	0	66	77	100	50	0	80	95	0	83
0	53	0	85	90	73	90	84	66	69	64	0	67	77	85
50	67	100	66	85	82	62	0	76	83	72	58	78	0	57
78	58	60	72	92	65	52	92	79	0	99	58	0	62	71
53	81	0	62	74	54	80	89	0	90	74	89	0	95	63
0	63	79	0	81	100	57	90	80	94	56	85	65	0	57
82	83	0	55	86	51	61	0	58	66	57	91	70	56	0
64	91	72	65	86	89	77	0	57	65	0	88	54	62	73



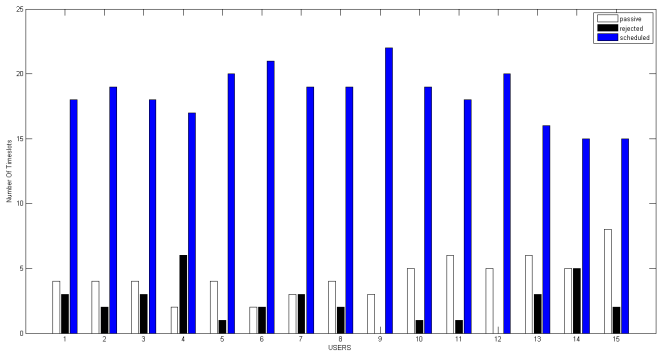
(a) L_{R-I} with VCG, case 1



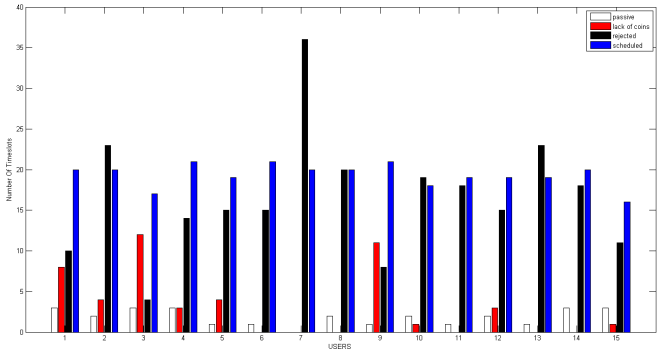
(b) L_{R-I} without VCG, case 1



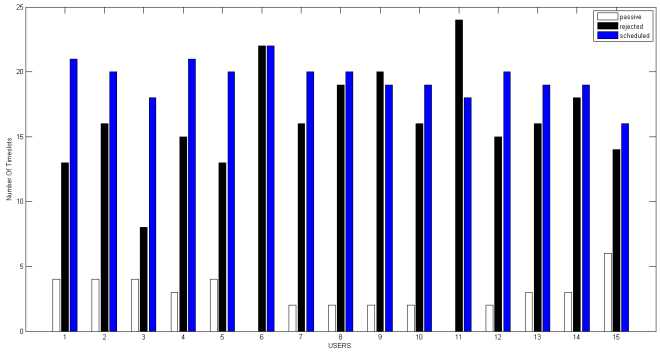
(c) L_{R-I} with VCG, case 2



(d) L_{R-I} without VCG, case 2

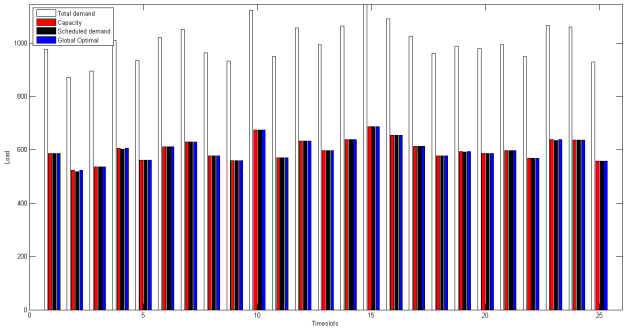


(e) L_{R-I} with VCG, case 3

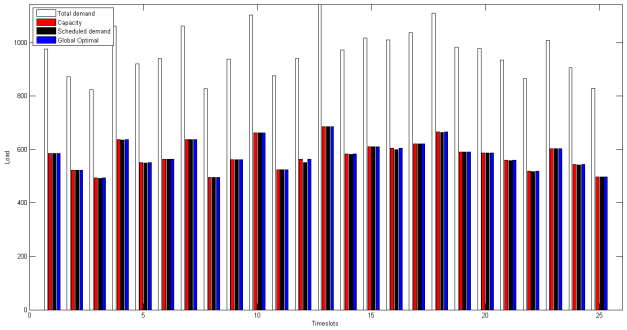


(f) L_{R-I} without VCG, case 3

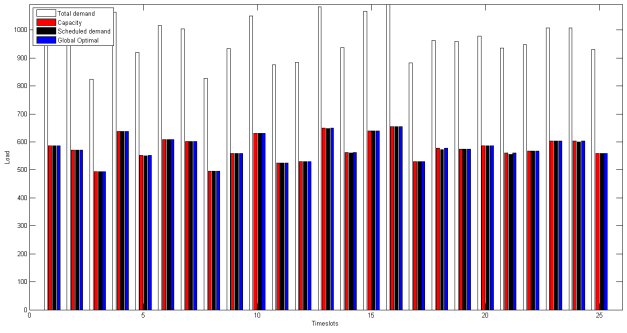
Figure 7.1: Comparison of Fairness by L_{R-I} ($T = 25$, Sufficient Capacity)



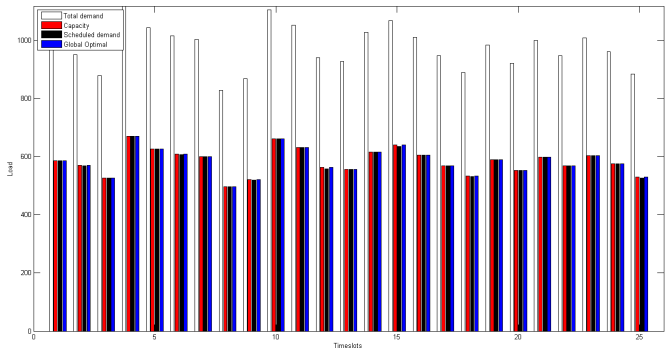
(a) BLA with VCG, case 2



(b) BLA without VCG, case 2

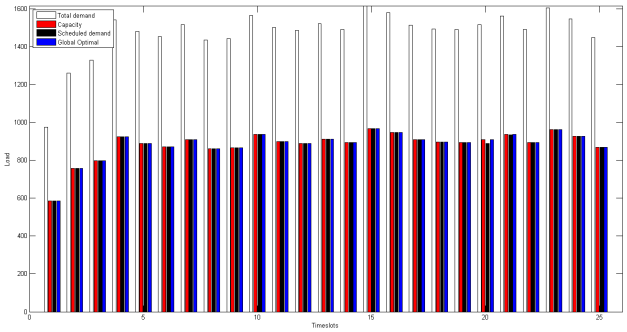


(c) $L_{R-I}(\lambda = 0.15)$ with VCG, case 2

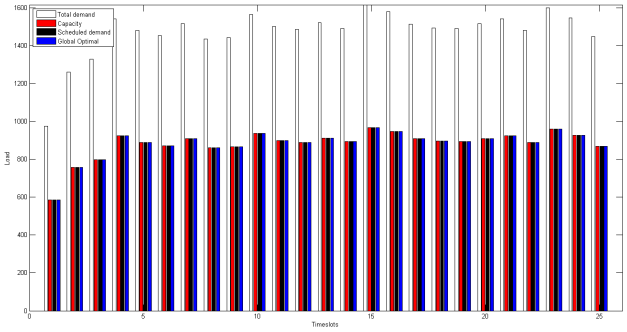


(d) $L_{R-I}(\lambda = 0.15)$ without VCG, case 2

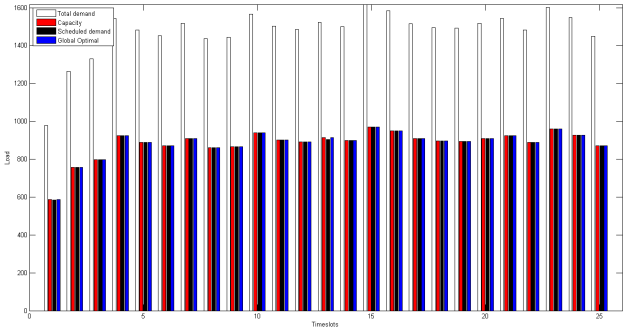
Figure 7.2: Comparison of Accuracy by BLA and L_{R-I} with Insufficient Capacity (Case 2, $T = 25$)



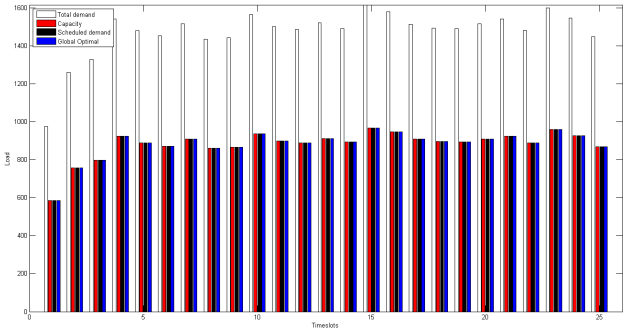
(a) BLA with VCG, case 3



(b) BLA without VCG, case 3

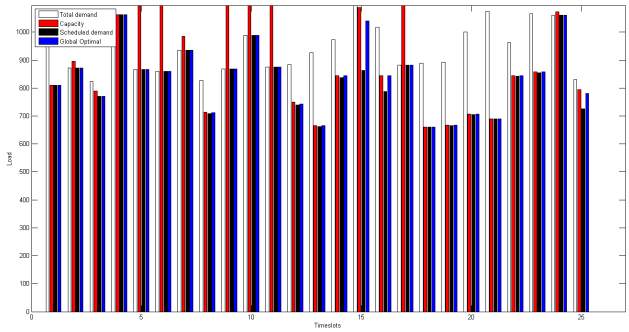


(c) $L_{R-I}(\lambda = 0.15)$ with VCG, case 3

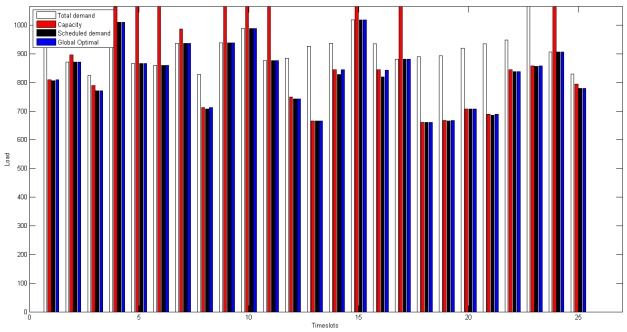


(d) $L_{R-I}(\lambda = 0.15)$ without VCG, case 3

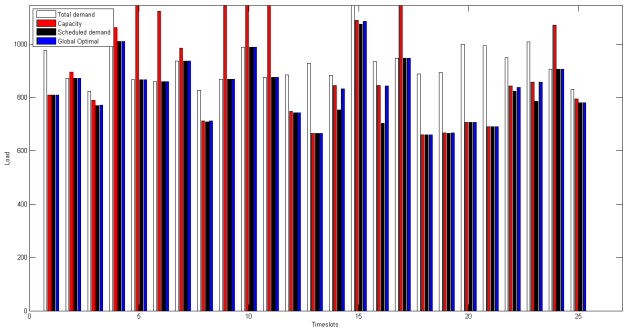
Figure 7.3: Comparison of Accuracy by BLA and L_{R-I} with Insufficient Capacity (Case 3, $T = 25$)



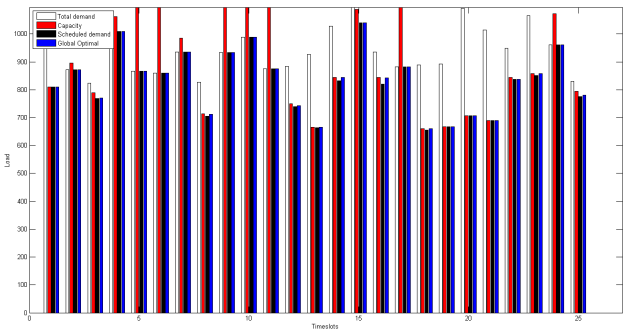
(a) BLA with VCG (case 2)



(b) BLA without VCG (case 2)

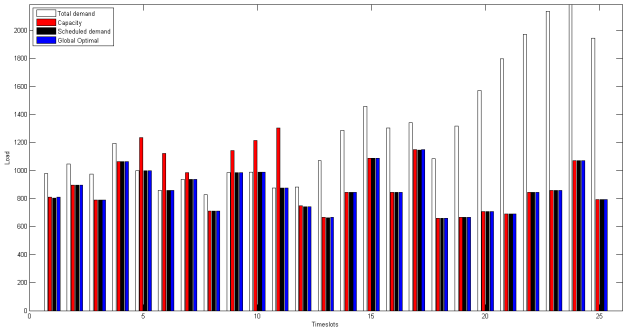


(c) $L_{R-I}(\lambda = 0.15)$ with VCG (case 2)

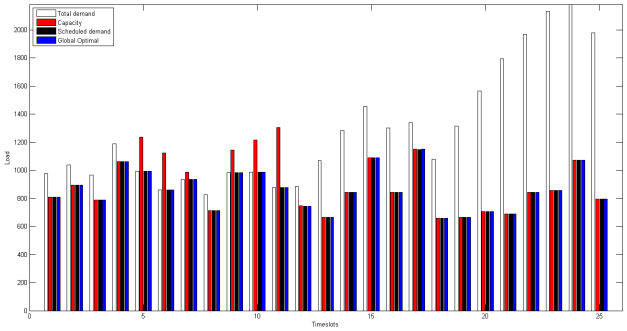


(d) $L_{R-I}(\lambda = 0.15)$ without VCG (case 2)

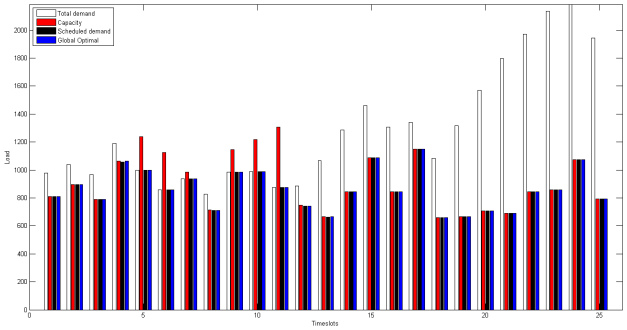
Figure 7.4: Comparison of Accuracy by BLA and L_{R-I} with Sufficient Capacity (Case 2, $T = 25$)



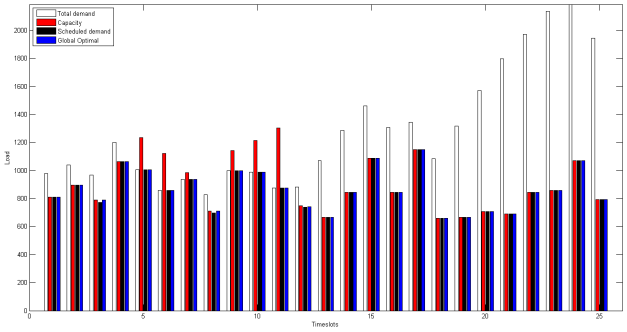
(a) BLA with VCG (case 3)



(b) BLA without VCG (case 3)



(c) $L_{R-I}(\lambda = 0.15)$ with VCG (case 3)



(d) $L_{R-I}(\lambda = 0.15)$ without VCG (case 3)

Figure 7.5: Comparison of Accuracy by BLA and L_{R-I} with Sufficient Capacity (Case 3, $T = 25$)

7.4 Matlab Code for case 1 and case 2 by BLA method

This section contains the MATLAB codes for scheduling by BLA with VCG, for two cases 1 and 2 with sufficient and insufficient capacity scenarios together. Only one of them for each scenario is to be chosen as per requirement of simulation. In the following codes, case 1 with sufficient capacity scenario is chosen. For implementation without VCG, the VCG part can simply be removed. The L_{R-I} method is skipped in this section but the codes for implementation of L_{R-I} can be seen in next section and case 1 and case 2 by L_{R-I} be simulated easily.

```
clc;
close all;
close all hidden
ts=25; % number of TimeSlots- max timeslots=100
users=15;
X0=zeros(1,users);
X1=zeros(1,users);
resultMatrix=zeros(ts,4);  % holds total demand,
   capacity and scheduled demand
coins=250*ones(1,users);
convergence=zeros(1,ts);
LeaDemandX=GenLod(ts,users);
LeaDemand111=zeros(ts,users); % for  decisions
gop=zeros(1,ts);  %for storing global optimal point
```

```
14
15 for z=(1:ts)   % loop to continue for ts timeslots
16
17 loop=150000; %total iterations
18 count=0;
19 LeaDemand= LeaDemandX(z,:);
20 totalDemand=sum(LeaDemand);
21    %SUFFICIENT CAPACITY
22    %{
23 capacityMatrix
      =[810,896,790,1063,1237,1124,986,713,1144,1216,1306,749,666,845,1089,84

24 capacity=capacityMatrix(z);
25    %}
26    %INSUFFICIENT CAPACITY
27 capacity=fix(totalDemand*0.6); % 60% of total demand
      as capacity, fix=integer value
28 rate=1;
29
30    %GlobalOptimalPoint(gop)
31  oldgop=gop(1,z);
32   for i=(1:users)  %assuming at least users are needed
         to reach gop
33    combos=nchoosek(LeaDemand,i);  %gives subsets of
         LeaDemand set with i elements each
34    sumcombos=sum(combos,2);
35    for j=(1:size(sumcombos));
36     if (sumcombos(j,1)<=capacity)&&(sumcombos(j,1)<=
          totalDemand)&&(sumcombos(j,1)>oldgop)
37      oldgop=sumcombos(j,1);
38     end
39    end
40    gop(1,z)=oldgop;
41    if ((gop(1,z)==capacity)||(gop(1,z)==totalDemand))
42     break  % for loop broken
43    end
44   end
45
46 passive=sum(LeaDemand==0);
47  %checking if coins are enough
```

```
48    for x=(1:users)
49     if LeaDemand(1,x)~=0
50     if (coins(1,x)<rate*LeaDemand(1,x))
51       LeaDemand(1,x)=1; %%demand is changed to 1 for
             all loads with insufficient coins
52     end
53     end
54    end
55
56    oldDecision = zeros(1,users);
57    decision = ones(1,users);
58    oldLeaDemandReqd=0;
59
60    maxLeaDemand=0;
61
62    iteration=1;
63    a0=ones(1,users); b0=ones(1,users); a1=ones(1,users)
         ; b1=ones(1,users);
64  while (iteration<=loop)
65   if (capacity>totalDemand)
66     oldLeaDemandReqd=totalDemand;
67     break
68   end
69   for i=(1:users);
70     if (LeaDemand(1,i)==0 || LeaDemand(1,i)==1)
71        decision(1,i)=0;
72     else
73       X0(1,i)=betarnd(a0(1,i),b0(1,i)); %obtaining
             value from BETA distribution
74       X1(1,i)=betarnd(a1(1,i),b1(1,i));
75       if(X0(1,i)>X1(1,i))
76        decision(1,i)=0;
77       else
78        decision(1,i)=1;
79       end
80     end
81   end
82   newLeaDemandReqd=sum(decision.*LeaDemand);
83    for i=(1:users)
84     if (LeaDemand(1,i)==0||LeaDemand(1,i)==1)
```

```
85      else
86       if(X0(1,i)>X1(1,i))
87        if((newLeaDemandReqd<=capacity)&&(
              newLeaDemandReqd>=oldLeaDemandReqd))
88          a0(1,i)=a0(1,i)+1;
89        else
90          b0(1,i)=b0(1,i)+1;
91        end
92      end
93      if(X0(1,i)<=X1(1,i))
94        if((newLeaDemandReqd<=capacity)&&(
              newLeaDemandReqd>=oldLeaDemandReqd))
95        a1(1,i)=a1(1,i)+1;
96        else
97        b1(1,i)=b1(1,i)+1;
98        end
99      end
100    end
101    end
102
103    %check if decision is being repeated
104   if(decision==oldDecision)
105    count=count+1;
106   else
107    count=0;
108   end
109   oldDecision=decision; % present decision is old
         decision for next iteration
110   if (count>50) % convergence if same decision occurs
         50 times
111    fprintf('%d communication to be stopped due to
         repeated answer from users \n',z)
112    break
113   end
114   if ((oldLeaDemandReqd<newLeaDemandReqd)&&(
         newLeaDemandReqd<=capacity))
115    oldLeaDemandReqd=newLeaDemandReqd;
116   end
117   iteration=iteration+1;
118
```

```
119 end  %end of while loop
120
121 resultMatrix(z,1)=totalDemand;
122 resultMatrix(z,2)=capacity; %store capacity
123 resultMatrix(z,3)=oldLeaDemandReqd;     %store total
      scheduled LoaDemand
124 resultMatrix(z,4)=oldgop;
125
126 convergence(z)=(oldLeaDemandReqd/gop(1,z))*100;
127
128 if (capacity<totalDemand)
129    % adding and reducing the coins
130  coinsD=0; selected=0;
131  for y=(1:users)
132  if (decision(1,y)==1 && (LeaDemand(1,y)~=0 ||
       LeaDemand(1,y)~=1)) % decision 1 & non-zero
       demands
133   coins(1,y)=coins(1,y)-rate*LeaDemand(1,y); % coins
        left with users
134   coinsD=coinsD+rate*LeaDemand(1,y);    % coins from
        selected users distributed to non selected
135   selected=selected+1;
136  end
137  end
138
139  coinDist=coinsD/(users-selected-passive);
140  coinDist=fix(coinDist);
141  for y=(1:users)
142   if (decision(1,y)==0 && LeaDemand(1,y)~=0) %
        decision 0 with non-zero demands
143    coins(1,y)=fix( coins(1,y)+coinDist   );
144   end
145  end
146 end
147
148    %SHIFTING NON ZERO DEMAND to next timeslot IF
         decision is zero
149 for y=(1:users)
150  if LeaDemand(1,y)==1 || (decision(1,y)==0 &&
       LeaDemand(1,y)>1)
```

```
151     %--CASE 1--
152     for y1=ts:-1:z+1  %%shifting loads for each user
          from last timeslot until next timeslot
153     LeaDemandX(y1,y)=LeaDemandX(y1-1,y); %%% same demand
          in next timeslot if current decision is zero
154     end
155     %--CASE 2--
156     %{
157   if (z<=(ts-1))
158   if LeaDemandX(z+1,y)==0
159     LeaDemandX(z+1,y)=LeaDemandX(z,y);
160   end
161   end
162     %}
163   end
164 end
165
166   for y=(1:users)
167     if decision(1,y)==1;
168      LeaDemand111(z,y)=1;
169     elseif (LeaDemand(1,y)==0)
170      LeaDemand111(z,y)=7;
171     elseif (LeaDemand(1,y)==1 )
172      LeaDemand111(z,y)=3;
173     elseif (LeaDemand(1,y)>1 && decision(1,y)==0)
174      LeaDemand111(z,y)=0;
175     end
176   end
177
178 end
179
180 passiveC=zeros(1,users);
181 insufficientcoin=zeros(1,users);
182 rejected=zeros(1,users);
183 scheduled=zeros(1,users);
184 rho=zeros(1,users);
185
186 for yy=(1:users)
187     passiveC(1,yy)=sum(LeaDemand111(:,yy)==7);
188     insufficientcoin(1,yy)=sum(LeaDemand111(:,yy)==3);
```

```
189     rejected(1,yy)=sum(LeaDemand111(:,yy)==0);
190     scheduled(1,yy)=sum(LeaDemand111(:,yy)==1);
191     rho(1,yy)= scheduled(1,yy) /(ts-passiveC(1,yy));
192 end
193
194 epsilonN=sum(rho);
195 epsilonD=sum(rho.*rho);
196 epsilon=(epsilonN*epsilonN)/(users*epsilonD);
197
198 format short  % round off the numbers to 4 digits
199 rho
200 format short
201 epsilon
202
203 for i=(1:users)
204 user(i)=(i);
205 end
206
207 figure(1)
208 bargraph=[passiveC(:),insufficientcoin(:),rejected(:),
      scheduled(:)];
209 hb=bar(user,bargraph,'grouped');
210 set(hb(1), 'FaceColor','w')
211 set(hb(2), 'FaceColor','r')
212 set(hb(3), 'FaceColor','k')
213 set(hb(4), 'FaceColor','b')
214 xlabel('USERS');
215 ylabel('Number Of Timeslots')
216 legend 'passive' 'lack of coins' 'rejected' 'scheduled
      ';
217
218 maximumatrix=max(resultMatrix);
219
220 figure(2)
221 bg=bar(resultMatrix); %creating bar graphs for
      differnet timeslots
222 set(bg(1), 'FaceColor','w')
223 set(bg(2), 'FaceColor','r')
224 set(bg(3), 'FaceColor','k')
225 set(bg(4), 'FaceColor','b')
```

```
226 axis([0 ts+1 0 maximumatrix(1) ]) ;
227 xlabel('Timeslots');
228 ylabel('Load')
229 legend 'Total demand' 'Capacity' 'Scheduled demand' '
      Global Optimal';
230 legend('Location','northwest');
```

7.5 Matlab Code for case 3 by L_{R-I} method

The MATLAB codes given here includes codes for both the insufficient and sufficient capacity scenario of case 3 by L_{R-I} method with VCG, so only one of them is to be chosen in each simulation. In the given codes, the insufficient capacity with VCG scenario is under implementation. The VCG part can be removed to simulate the scenario without VCG. The BLA method can be implemented by replacing the part of codes for L_{R-I} method by implementing codes from previous section for BLA.

```
1  clc;
2  close all;
3  close all hidden
4  lambda=0.15;
5  ts=25;
6     % number of TimeSlots- max timeslots=100, 25 for
         sufficient capacity
7  users=15;
8  resultMatrix=zeros(ts,3);
9  coins=500*ones(1,users);
10 LeaDemandX=2*ones(ts,users,5);
11 LeaDemandX(:,:,1)=GenLod(ts,users); %calling function
      GenLod that stores the loads
12 LeaDemand111=5*ones(ts,users,5); %%% this matrix holds
       the decisions
13 gop=zeros(1,ts);  %for storing global optimal point
14
15 for z=(1:ts)
16 fprintf('This is timeslot %d \n',z)
17 loop=150000; %total iterations
18 count=0;
19 LeaDemand= LeaDemandX(z,:,:);
20 totalDemand=0;
```

```
21
22 for y=(1:users)
23   for y1=(1:5)
24       if LeaDemand(1,y,y1)>20
25       totalDemand=totalDemand+LeaDemand(1,y,y1);
26       end
27   end
28 end
29    %SUFFICIENT CAPACITY
30    %{
31 capacityMatrix
      =[810,896,790,1063,1237,1124,986,713,1144,1216,1306,749,666,845,1089,84

32 capacity=capacityMatrix(z);
33    %}
34    %INSUFFICIENT CAPACITY
35 capacity=fix(totalDemand*0.6); % 60% of total demand
      as capacity
36    %GlobalOptimalPoint(gop)
37 LeaDemandxx=zeros(1,75);  %assuming max 5 loads per
      user per ts
38 mm=0;
39  for j=1:5
40  for i=1:users
41    if LeaDemand(1,i,j)>10
42      mm=mm+1;
43       LeaDemandxx(mm)=LeaDemand(1,i,j);
44    end
45  end
46  end
47
48  if(capacity<totalDemand)
49   oldgop=gop(1,z);
50   for i=(10:24)  %assuming  10-24 loads are enough to
        reach gop
51     combos=nchoosek(LeaDemandxx(1,1:24),i);  %gives
          subsets of LeaDemand set with i elements each
52     sumcombos=sum(combos,2);
53    for j=(1:size(sumcombos));
```

```
54      if ((sumcombos(j,1)<=capacity)&&(sumcombos(j,1)>
            oldgop))
55        oldgop=sumcombos(j,1);
56      end
57      if ((oldgop==capacity))
58      break  % for loop broken
59      end
60     end
61    gop(1,z)=oldgop;
62
63     if ((gop(1,z)==capacity))
64     break  % for loop broken
65     end
66    end
67   end
68  rate=1;
69  passive=sum(sum(LeaDemand == 0));
70  receivers=0;
71    %checking if coins are enough
72  appload=zeros(1,users);
73     for x=(1:users)
74     for y=(1:5)
75       if LeaDemand(1,x,y)==2
76       else
77         appload(1,x)=appload(1,x)+LeaDemand(1,x,y);
78       end
79     end
80     end
81     for x=(1:users)
82     if coins(1,x)<rate*appload(1,x)
83       for x1=(1:5)
84       if (LeaDemand(1,x,x1)~=0 && LeaDemand(1,x,x1)~=2
             )
85          LeaDemand(1,x,x1)=1; %%demand is changed to 1
               for all loads with insufficient coins
86       end
87       end
88     end
89     end
90    oldDecision=zeros(1,users,5);
```

```
91    decision=ones(1,users,5);
92    oldLeaDemandReqd=0;
93
94   iteration=1;
95   PMat=ones(2,users,5)*0.5; %initial probability of
         decision of each user is 0.5
96
97   while (iteration<=loop)
98     %newLeaDemandReqd=0;
99   if (capacity>=totalDemand)
100    oldLeaDemandReqd=totalDemand;
101    gop(1,z)=totalDemand;
102    for x=(1:users)
103    for y=(1:5)
104      if LeaDemand(1,x,y)>10
105          decision(1,x,y)=1;
106      end
107    end
108    end
109    break
110  end
111
112
113   for i=(1:users)
114   for i1=(1:5)
115     if (LeaDemand(1,i,i1)==0 || LeaDemand(1,i,i1)==1
            || LeaDemand(1,i,i1)==2 )
116          decision(1,i,i1)=0;
117     else
118       Random=rand(1,1);
119       if (PMat(1,i,i1)>=Random)
120       decision(1,i,i1)=1;
121       else
122       decision(1,i,i1)=0;
123       end
124     end
125   end
126   end
127   newLeaDemandReqd=sum(sum(sum(decision.*LeaDemand)));
128   for i=(1:users)
```

```
129     for i1=(1:5)
130       if (LeaDemand(1,i,i1)==0 || LeaDemand(1,i,i1)==1
              || LeaDemand(1,i,i1)==2)
131       else
132       if(newLeaDemandReqd<=capacity && newLeaDemandReqd
             >=oldLeaDemandReqd) %%REWARD
133         if(decision(1,i,i1)==1)
134           PMat(2,i,i1) = (1-lambda)*PMat(2,i,i1);
135           PMat(1,i,i1) = 1-PMat(2,i,i1);
136         elseif (decision(1,i,i1)==0)
137           PMat(1,i,i1) = (1-lambda)*PMat(1,i,i1);  %
                change in NO
138           PMat(2,i,i1) = 1-PMat(1,i,i1);
139         end
140       end
141       end
142     end
143     end
144
145      %check if decision is repeated
146    if(decision==oldDecision)
147     count=count+1;
148    else
149     count=0;
150    end
151    oldDecision=decision; % present decision is old
          decision for next iteration
152    if (count>50) % convergence if same decision occurs
          50 times
153    fprintf('%d communication to be stopped due to
          repeated answer from users \n',z)
154    break
155    end
156    if ((oldLeaDemandReqd<newLeaDemandReqd)&&(
          newLeaDemandReqd<=capacity))
157     oldLeaDemandReqd=newLeaDemandReqd;
158    end
159    iteration=iteration+1;
160  end
161
```

```
162 resultMatrix(z,1)=totalDemand;
163 resultMatrix(z,2)=capacity; %store capacity
164 resultMatrix(z,3)=oldLeaDemandReqd;     %store total
      scheduled LoaDemand
165 resultMatrix(z,4)=gop(1,z);
166
167 convergence(z)=(oldLeaDemandReqd/gop(1,z))*100;
168
169  % adding and reducing coins from users
170 if (capacity<totalDemand)
171  coinsD=0; selected=0;
172 for y=(1:users)
173  for y1=(1:5)
174   if (decision(1,y,y1)==1 ) % decision 1 with non-zero
       demands
175   coins(1,y)=coins(1,y)-rate*LeaDemand(1,y,y1); %
      coins left with users
176   coinsD=coinsD+rate*LeaDemand(1,y,y1);   % coins from
       selected users distributed to non selected
177   selected=selected+1;
178   end
179   if (decision(1,y,y1)==0 && LeaDemand(1,y,y1)~=0 &&
      LeaDemand(1,y,y1)~=2  )
180   receivers=receivers+1;
181   end
182  end
183 end
184 coinDist=fix( coinsD/receivers);
185
186 for y=(1:users)
187 for y1=(1:5)
188  if (decision(1,y,y1)==0 && LeaDemand(1,y,y1)~=0  &&
      LeaDemand(1,y,y1)~=2 )
189  coins(1,y)= coins(1,y)+coinDist   ;
190  end
191 end
192 end
193     %shifting of non-zero DEMAND to NEXT ITERATION if
       rejected
194 for y=(1:users)
```

```
for y1=(1:5)
 if LeaDemand(1,y,y1)==1 || (decision(1,y,y1)==0 &&
    LeaDemand(1,y,y1)>5)
 if (z<=(ts-1))
  for y2=(1:5)    %assign current load to same
     appliance's first non-0 non-2 load in next
     timeslot
  if ( LeaDemandX(z+1,y,y2)==0 || LeaDemandX(z+1,y,y2)
     ==2 )
    LeaDemandX(z+1,y,y2)=LeaDemandX(z,y,y1);
    break
  end
  end
 end
 end
end
end
end

for y=(1:users)
for y1=(1:5)
 if (capacity<totalDemand)
  if  decision(1,y,y1)==1;
    LeaDemand111(z,y,y1)=1;
  elseif (LeaDemand(1,y,y1)==0)
    LeaDemand111(z,y,y1)=7;
  elseif (LeaDemand(1,y,y1)==1 )
    LeaDemand111(z,y,y1)=3;
  elseif (LeaDemand(1,y,y1)>5 && decision(1,y,y1)==0)
    LeaDemand111(z,y,y1)=0;
  end
 else
  if (LeaDemand(1,y,y1)==0)
   LeaDemand111(z,y,y1)=7;
  elseif (LeaDemand(1,y,y1)>10)
   LeaDemand111(z,y,y1)=1;
  end
 end
end
end
```

```
231
232 end % end of for loop
233
234 passiveC=zeros(1,users);
235 insufficientcoin=zeros(1,users);
236 rejected=zeros(1,users);
237 scheduled=zeros(1,users);
238 rho=zeros(1,users);
239
240 for yy=(1:users)
241 for yy1=(1:ts)
242   for yy2=(1:5)
243     if LeaDemand111(yy1,yy,yy2)==7
244       passiveC(1,yy)=passiveC(1,yy)+1;
245     elseif LeaDemand111(yy1,yy,yy2)==3
246       insufficientcoin(1,yy)=insufficientcoin(1,yy)+1;
247     elseif LeaDemand111(yy1,yy,yy2)==0
248       rejected(1,yy)=rejected(1,yy)+1;
249     elseif LeaDemand111(yy1,yy,yy2)==1
250       scheduled(1,yy)=scheduled(1,yy)+1;
251     end
252    end
253 end
254 rho(1,yy)=scheduled(1,yy)/(rejected(1,yy)+
    insufficientcoin(1,yy)+scheduled(1,yy));
255 end
256 epsilonN=sum(rho);
257 epsilonD=sum(rho.*rho);
258 epsilon=(epsilonN*epsilonN)/(users*epsilonD);
259
260 format short  % round off the numbers to 4 digits
261 rho
262 format short
263 epsilon
264
265 for i=(1:users)
266 USER(i)=(i);
267 end
268
269 figure(1)
```

```
bargraph=[passiveC(:),insufficientcoin(:),rejected(:),
    scheduled(:)];
hb=bar(USER,bargraph,'grouped');
set(hb(1), 'FaceColor','w')
set(hb(2), 'FaceColor','r')
set(hb(3), 'FaceColor','k')
set(hb(4), 'FaceColor','b')
xlabel('USERS');
ylabel('Number Of Timeslots')
legend 'passive''lack of coins''rejected''scheduled';

maximumatrix=max(resultMatrix);
figure(2)
bg=bar(resultMatrix);
set(bg(1), 'FaceColor','w')
set(bg(2), 'FaceColor','r')
set(bg(3), 'FaceColor','k')
set(bg(4), 'FaceColor','b')
axis([0 ts+1 0 maximumatrix(1)]);
xlabel('Timeslots')
ylabel('Load')
legend 'Total demand''Capacity''Scheduled demand''
    Global Optimal';
legend('Location','northwest');
```