Received November 7, 2017, accepted December 19, 2017, date of publication December 29, 2017,
date of current version February 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2788051

A Learning Automaton-Based Scheme for Scheduling Domestic Shiftable Loads in Smart Grids

RAJAN THAPA[1], LEI JIAO[1], B. JOHN OOMMEN[1,2], (Fellow, IEEE), AND ANIS YAZIDI[3]
[1]Department of Information and Communication Technology, University of Agder, 4879 Grimstad, Norway
[2]School of Computer Science, Carleton University, Ottawa, ON K1S 5B6, Canada
[3]Department of Computer Science, Oslo and Akershus University College, 0167 Oslo, Norway

Corresponding author: Lei Jiao (lei.jiao@uia.no)

ABSTRACT In this paper, we consider the problem of scheduling shiftable loads, over multiple users, in smart electrical grids. We approach the problem, which is becoming increasingly pertinent in our present energy-thirsty society, using a novel *distributed* game-theoretic framework. In our specific instantiation, we consider the scenario when the power system has a local-area Smart Grid subnet comprising of a single power source and multiple customers. The objective of the exercise is to tacitly control the total power consumption of the customers' shiftable loads, so to approach the rigid power budget determined by the power source, but to simultaneously not exceed this threshold. As opposed to the ''traditional'' paradigm that utilizes a central controller to achieve the load scheduling, we seek to achieve this by pursuing a distributed approach that allows the users[1] to make individual decisions by invoking negotiations with other customers. The decisions are essentially of the sort, where the individual users can choose whether they want to be supplied or not. From a modeling perspective, the *distributed* scheduling problem is formulated as a game, and in particular, a so-called ''Potential'' game. This game has at least one pure strategy Nash equilibrium (NE), and we demonstrate that the NE point is a global optimal point. The solution that we propose, which utilizes the theory of learning automata (LA), permits the total supplied loads to approach the power budget of the subnet once the algorithm has converged to the NE point. The scheduling is achieved by attaching a LA to each customer. The paper discusses the applicability of three different LA schemes, and in particular, the recently-introduced Bayesian learning automata. Numerical results, obtained from testing the schemes on numerous simulated data sets, demonstrate the speed and the accuracy of proposed algorithms in terms of their convergence to the game's NE point.

INDEX TERMS Smart grids, scheduling, potential game, machine learning, learning automata.

I. INTRODUCTION

As society becomes increasingly energy-thirsty, the problems associated with collectively controlling the use of energy resources so that the electrical grids are not overloaded, are becoming more dominant.[2] Power utility companies often warn customers to limit their power consumption especially in the warmer summer months. Although this is deemed to be voluntary, these utility companies attempt to enforce it by charging higher rates for the power that is consumed during ''peak hours''.

Utility companies attempt to monitor and control the use of energy by resorting to so-called ''Smart Grids'' (SGs). In a SG, loads can be categorized as being either ''shiftable'' or ''non-shiftable''. Non-shiftable loads comprise of devices such as bulbs, where there is no room for scheduling, since the power required by the device must be supplied as soon as the device is turned on. Shiftable loads, on the other hand, such as water and floor heaters,[3] can

[1]In this paper, we use the terms ''customers'' and ''users'' interchangeably.

[2]*Wikipedia* records an interesting report about the so-called *Northeast Blackout* of 2003. This caused a widespread power outage that occurred throughout parts of the Northeastern and Midwestern United States and the Canadian province of Ontario on Thursday, August 14, 2003, during a ''heat wave'', just after 4:10 PM (EDT). Apparently, the outage affected an estimated 10 million people in Ontario and 45 million people in eight U.S. states. Although some power was restored by 11:00 PM (EDT) on the same day, many others did not get their power back until about two days later. Indeed, in more remote areas, it took nearly a week to restore power. At that time, it was the world's second-most widespread blackout in history. *We conjecture that the universal use of SGs could have mitigated this blackout*!

[3]These are every-day, commonplace appliances in countries with colder climates.

tolerate a certain amount of delay, permitting the users the possibility to schedule them when they are turned on. Since these shiftable loads can be adaptively scheduled, the system is capable of smoothing the domestic power consumption curve.

SGs have gained interest and popularity in the academia [2]–[4], and also in the industry [5]. Indeed, they have already become a reality in many developed countries; many of these countries have already deployed SG architectures in their power networks.

Power scheduling approaches that work with the SG paradigm fall under two main classes, namely, those that use centralized [6]–[9] or distributed approaches[4] [10]–[12]. Centralized scheduling schemes involve a Central Controller (CC) that decides the responses that are made when it concerns the users' demands. Within these schemes, customers are required to send their demands to the supplier's CC so that the scheduling can be achieved. These methods are frowned upon – because the deployment of an external controller to regulate the power supplied to the entire customer base could violate the privacy that the customers expect. This could also lead to a biased "differentiated" treatment based on the identity of the customer, where certain "preferred" customers (for example, those who are economically sound), are provided with a superior class of service. As opposed to this, a method that invokes distributed power scheduling, effectively reduces the significance of the power controller in the process of load scheduling, because it permits the peer users to negotiate a scheduling protocol between themselves. Consequently, distributed power scheduling methods are of great interest due to these potential benefits. Besides, over and above these, one reaps the well-acclaimed advantages associated with the distributed model of computation.

There is a vast body of literature associated with achieving distributed scheduling in SGs, all of which focus on the various facets of the problems encountered in this area [12]–[14]. The main focus of the existing studies that use distributed algorithms is to distribute the computational load to multiple controllers/agents in order to reduce the overall communication and computational complexity, and consequently to "spread them out" to be handled by the individual users. In these cases, the appliances of the end-users (the actual customers) may still be controlled by a local controller/agent.

As opposed to this, in this current study, the various customers are allowed to decide by themselves whether they want to turn a load on or not. Giving the end-user the ability to make these decisions eliminates the role of the suppliers/agents to achieve the scheduling of the load in the local-area subnets. To achieve this goal, we advocate a distributed Learning Automata (LA)-based approach, where each customer is equipped with a LA to learn from the environment in order to decide whether to turn on its appliances or not.

[4]A more detailed explanation of these two modes of operation is found in Section III-C.

A LA is an adaptive decision-making unit, which learns the optimal action out of a set of actions provided by the environment it operates in [15] and [16]. The beauty of a LA is that it *learns* from its environment *during the course of interacting* with it. In other words, the tasks of learning and taking actions happen simultaneously. In classic LA, such as in the Linear Reward-Inaction (LRI) scheme and the family of Pursuit algorithms, the learning speed needs to be pre-determined (by determining the learning parameter which controls the speed of learning) in order to achieve the optimal trade-off between the machine's speed and accuracy. Indeed, this becomes a limitation in the the load balancing scenarios where the environment can be quite different from one system to another, and can even change over time. Because it is not easy to find a universal learning parameter which can provide a high performance for the LA in all different environments, a superior type of LA, whose learning speed can be adaptive to different types of environments, is desirable.

The application of LA in SGs has been studied a little, including using them in the underlying communication network in SGs [17], [18], and in the power scheduling approaches [19]. The solution model we propose in this paper is distinct. Firstly, we model the system as a specific type of game and proceed to study its properties. Based on the these properties, we design a distributed LA-based algorithm to solve the game. With regard to solution strategies, we propose the deployment of three LA schemes, namely the LRI [16], the Coordination-game Learning Automata (CLA) [20], and the more-recently introduced Bayesian Learning Automata (BLA) [21]. We emphasize that in the case of all these LA-based schemes, the consumers do not need to share information to the provider. Rather, they can negotiate the power utilization and make a decision between themselves, implying that the power supplier has to merely perform the task of being a power budget *provider*, rather than also a *scheduler*.

A. CONTRIBUTIONS OF THE PAPER

The novel contributions of this paper are listed below:

- We present a comprehensive solution to using LA to control SGs in a distributed manner.
- We demonstrate that the control of the SG reduces to a multi-player game, and in particular, to a so-called Potential Game possessing at least one pure strategy Nash equilibrium.
- We have shown how we can control the SG by assigning a LA to each user.
- We have also recorded the distinct advantages of using three types of LA to achieve this control, and in particular, the recently-introduced Bayesian Learning Automata (BLA).
- Finally, we have included experimental results that demonstrate the power of each of these LA in achieving the control of the SG.

B. ORGANIZATION OF THE PAPER

The rest of this paper has been organized as follows. Section III presents the preliminaries that the paper is built on and a fairly brief survey of the field is included so that this document is a stand-alone publication. Section IV contains the model of the system and formulates the problem to be solved. In Section V, we study the analysis of the game and its properties. In Section VI, we explain the details of the implementation of the distributed decision-making algorithm (i.e., the protocol for the selection or rejection of the load) for the LA-based scheduling. The simulations and numerical results of the algorithms have been presented in Section VII. We conclude this paper in Section VIII, where we also briefly visit the avenues for future work.

II. WHAT ARE SMART GRIDS

Before we proceed, it is wise to briefly explain what SGs are and how they operate.

A lot of research has been carried out in the field of SGs with the attempt to replace/upgrade the traditional electric grid to yield advanced "intelligent" or Smart Grid systems. Many power grid operators in the USA, Canada, China, South Korea etc. have already implemented different features of SGs to overcome the shortcomings of traditional electrical grid system [3]. SG systems are expected to implement these aspects, possibly with some slight modifications, based on their specific requirements. Various studies have been performed on all of these aspects, and improvements have been suggested for each of them. These improvements have led to the development of the overall SG network, and have been incorporated into the various sub-components such as electrical substations, transmission lines, and the corresponding electrical and electronic devices that communicate with each other in a fast, secured and reliable manner.

The primary significant differences between the traditional grid systems, that still dominate most parts of the world, and SGs are explained in [22]. These are as summarized in Table 1.

TABLE 1. Differences between the traditional grid and modern SG [22].

Traditional grid	Smart Grid
Electromechanical network	Digital network
One-way communication	Two-way communication
Centralized generation	Distributed generation
Few sensors	Sensors throughout
Manual monitoring	Self-monitoring
Manual restoration	Self-healing
Failures and blackouts	Adaptive and islanding
Limited control	Pervasive control
Few customer choices	Many customer choices

Gungor *et al.* [3] have also explained how traditional grid systems are being upgraded and replaced by SGs. They have also given due considerations for the issues/hurdles encountered, and also for the potentials and opportunities for even smarter grid systems. Different wired and wireless means of communication have been used in SGs, as per the respective requirements and feasibility. This has led to the real-time monitoring and access control of customers' appliances, so as to achieve real-time electricity-price signaling and efficient fault diagnosis. The communication methods between the SG entities for transferring data and control commands are discussed in [23]. More recent advancements in the SG technology have resulted in the establishment of bi-directional connectivity of IP addressable devices which, in turn, provide services to both the set of consumers and the operators [24]. In this manner, all the details of the devices used can be known by other remote devices or controllers by the sharing of information through sensors attached to the devices.

III. FUNDAMENTAL ISSUES IN SGs

A. DEMAND RESPONSE

The phenomenon of Demand Response (DR) is a prime focus area in field of SGs, and it can involve several possible methods. The DR is defined as follows: "Changes in electric usage by end-use customers from their normal consumption patterns in response to changes in the price of electricity over time, or to incentive payments designed to induce lower electricity use at times of high wholesale market prices or when system reliability is jeopardized" [25]. Put informally, anything that is done on the consumers' side (loads) to change the total demand is referred to as the DR [26]. In other words, it involves the response made by the loads (demands) due to changes in certain constraints in SG system. In particular, "Demand Scheduling" is also a component of DR, where the consumers have to change their usage behavior, by the process of selection or rejection, to either utilize or refrain from utilizing the capacity offered by the SG.

Demand Scheduling can be achieved by applying decisions that are based on mathematical logical operations, directly on the demands or through constraints set by the SG's CC. The task of Demand Scheduling is a DR process which can be done either distributively or centrally, and several algorithms and ideas that have been implemented in SGs have been reported. These are explained in more detail in Section III-C. The effect of these operations on the system's demands varies depending upon the methods applied to achieve the scheduling. In particular, techniques such as load shifting, peak clipping, conservation, load building, valley filling and assigning flexible loads, are used for DR [9]. The methods devised in this paper fall within the family of load shifting techniques, in which the loads encountered at high demand times are transferred so that they are done at times when the demand is lower.

Classical Direct Load Control (DLC) mentioned in [27] is a classical and yet the simplest method in which the CC is assumed to have the complete control on the consumers' loads. From the users' perspective, such a scheme is sub-optimal when it concerns fairness of selection [28], privacy, and the security associated with the users' information [29].

Loads can be scheduled on the basis of the priority of their time constraints, i.e., on a first-come-first-serve basis. Such a scheme has been described in [30] for electric vehicles, where the demand requests arriving earliest are provided with the highest priority, while those arriving later are assigned a lower priority. In this method, the loads with a higher priority get served early and demands with lower priority will be placed in a queue if the SG is not able to serve them all simultaneously. This can be seen to be a centralized mechanism, because the controller should be able to know their demand values along with their time of arrival. The SG's controller uses the time of arrival as a decision parameter to accomplish the scheduling, i.e., to decide whether they have to place the demands in a queue and/or to serve them. Thereafter, the demands in the queue will be activated as per their positions, and whenever one/some of the demands get served it will provide sufficient space for other demands to be placed in the queue.

B. DYNAMIC PRICING

Dynamic Pricing is one of the most popular approaches by which DR is achieved. This motivates the users of the SG to change the behavior of their usage according to the varying prices of electricity. This, in turn, is enforced with the aid of smart meters that obtain information about the end users' load devices [31]. This method favors both the users and the SG provider, because by virtue of their choice, the users can take advantage of these dynamic electricity prices and thus lower their electricity bills. Simultaneously, operators can indirectly control (at least, in a weak manner) the unusual peaks and troughs of the electricity demands on the SG.

Time-based and incentive-based DR programs, that should be compared against the "flat" electricity pricing model, have been proposed and studied in [26]. These models have already been implemented in many SGs around the world. The distributed dynamic pricing system proposed in [29] is a method by which the users are charged as per dynamic prices, that are set in place to encourage users to bring changes to their demands. A high price and a low price at peak demand times and at low demand times respectively, have been proposed, and these have been seen to be helpful in bringing about a change in the behavior of the users' power usage. Of course, however, this method is dependent on the users' financial condition and their behavioral patterns.

Another method which achieves Demand Scheduling employs the strategy of postponing the load demand up to its deadline, when the power utilization in the SG has reached it's peak. This method is called Threshold Postponement, and was devised in [32]. The method utilizes the advantage of time-flexibility, so that shiftable loads can wait to get served. When the SG's power consumption is low, the loads can be added into the grid for consumption. Otherwise, they will be placed in a queue, waiting to be served. Koutsopoulos and Tassiulas [32] also discussed the economical advantages of scheduling the demands over the cost of supplying the extra demands whenever the capacity of the SG is lower than its total demands.

C. DISTRIBUTED VS CENTRALIZED SCHEDULING ALGORITHMS

Generally speaking, there are two reported families of methods available for load selection:

- By the CC itself taking care of the task, and this is referred to as "centralized" selection, and
- The decision being made by the users, and this is referred to as "distributed" selection.

The main issues that have to be taken into consideration when one considers these families of methods are fairness of selection [27], privacy and secrecy of user information, and consumer's usage behavior [33]. These aspects are addressed in different ways by the above-mentioned families of methods depending upon the algorithm utilized. Most of the research done in SGs is based on the centralized selection model. This is the simpler model and it is easier to design and implement, because it allows the proprietor of the SG to implement the decision-making algorithm through its CC. Methods within this family are, generally, considered to be to the advantage of the SG's operator. The operators of the SG attempt to satisfy the user's concerns, so as to avoid their dissatisfaction and the possibility of them changing their providers. Distributed methods utilize the decisions of users themselves for the usage of electricity, with the user possibly accessing the SG fully, partially or by withdrawing from it all together. The family of distributed methods are expected to carry out a fair selection algorithm so as to avoid the disapproval of rejected users, and they do this by transferring the control to the users themselves – unlike the fully owner-controlled centralized methods.

The real-time pricing system, where the controller of the SG changes the electricity prices according to fluctuations in demands and capacity, have been devised in [7] and [34], and by many other researchers. This is a distributed method of power scheduling which forces the consumers to change their usage behavior to "synchronize" with the changing rates. The authors generally claim that such a model motivates users to change their electricity usage behavior. This is because the consumers' demand is expressed as a function of numerous factors such as electricity price, time of day, region, weather etc. Likewise, the classical DLC method discussed in [27] is a centralized method of load selection where the CC has full control over the devices, whence it can supply power as per its own will. Wang *et al.* [35] proposed scheduling algorithms were distributed approaches in nature, while a central unit was adopted for the purposes of information sharing.

The most prominent disadvantage of using centralized selection methods is the fact that they are (or can be perceived to be) biased. They can favor certain preferred users, e.g., those who are more financially secure, and who are thus less influenced by dynamic pricing. Further, the SG system's CC can directly and/or indirectly control user demands by invoking dynamic electricity prices. The distributed methods were proposed to address these issues, i.e., to permit users themselves to decide whether to get involved in power scheduling or not. But these distributed methods have not been able to fully remove the CC's control over the users' demands. Rather, the SG's CC can indirectly control the demands through different constraints in these distributed methods.

The aim of this paper is to propose "pure" distributive method where the users' decision is not influenced by the CC in any way, and where the CC's role, on the other hand, is restricted only to supplying the power needed on the SG after the scheduling has been done as shown in Figure 1.

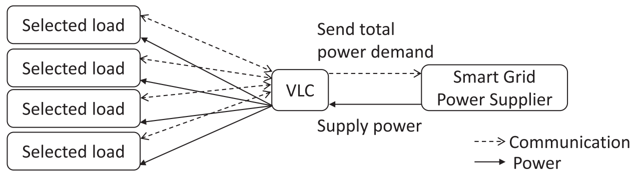


FIGURE 1. Limiting the role of the Smart Grid's CC and protecting privacy of users.

In addition, it would be desirable if all the users are satisfied with the SG's service by serving their important demands, while their less-important demands are either selected or rejected in the overall scheduling process. Indeed, we shall demonstrate that it is possible to generate decisions solely from users' perspective by the application of a rather simple mathematical model, and without the CC bearing any influence in the overall power distribution. This will also guarantee the protection of the privacy of the users' information – which is a facet that has not been adequately addressed in the literature before.

From Figure 1, we can clearly observe that the demand scheduling, denoted by the dotted lines, is done by communication between the users and the Virtual Logical Controller (VLC)[5] only. The CC of the SG has its limited role, receiving the total request from the VLC according to decision generated at the users' side, and supplying the power accordingly, which is shown by two arrows between the VLC and the SG's CC. According to the requests (from the loads that have been selected already), the users will be served with power through the VLC, again restricting the communication between CC and the users.

IV. SYSTEM MODEL

A. PROBLEM FORMULATION

The research undertaken in this paper focuses on the domestic smart-grid subnet. Figure 2 illustrates a hierarchical structure of the power grid which includes several levels. The lowest level, i.e., the local domestic network between the transformers and the households, is the subnet that we shall concentrate on here, which is a micro-smart grid. A typical scenario of this subnet, as illustrated in Figure 2, is an apartment building with a few families which play the role of the customers, and where the building is connected to a main power source. This power source is provided and installed by the power supplier, and it obtains its power budget from the upper levels of the power network based on the scheduling of the supplier. The objective of the power source is to provide power to the

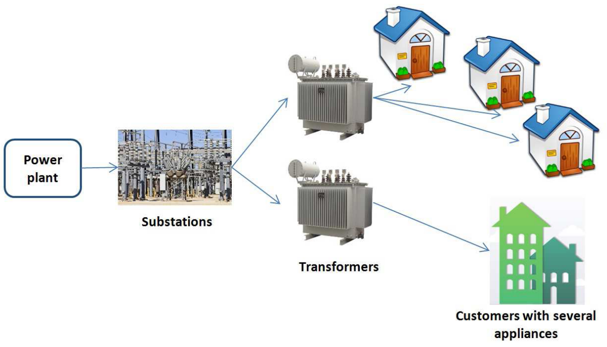


FIGURE 2. A high level structure of the power grid.

various families, and at the same time to maintain the overall power consumption below a given power budget.

Following the common practice [36]–[39], the overall budget for the shiftable load can be suggested by the source to the customers, and this quantity is denoted by C_{SL}. Typically, the time index is segmented into slots, indexed by t, which are of the order of several minutes long. In the beginning of each time slot, the budget for the shiftable load is offered to all the customers. But once the customers obtain this budget, they will have to compete with each other for their own loads. Once a consumer wins the competition, his load will be served within this time slot. The competition among the various customers is carried out through mutual communications and information exchange.

Suppose there are, in total N customers, indexed by $i \in \{1, 2, \ldots, N\}$, who have their individual aggregated demands $\{L_1, L_2, \ldots, L_N\}$ for their respective shiftable loads at time slot t. C_{SL}, referred to above, may not be sufficient to serve all the $\{L_i\}$ loads for all the customers. It would thus be necessary for the system to figure out which users can be served such that $\sum_{i=1}^{N} L_i d_i \leq C_{SL}$, where $d_i \in \{1, 0\}$ denotes whether customer i is to be served or not. In other words, a decision of 1 for a particular customer implies that the specific customer's demand is to be served by the grid in the current time slot, while the decision of 0 means that the corresponding load demand will not be served by the grid in the current time slot. Thus, clearly, all the the users who attain the decision 1 accomplish the sharing of the total shiftable loads' budget, C_{SL}. However, a customer that is not served in the current time slot will eventually be served in the future time slots. The objective of the distributed scheduling problem is to determine a proper sub-group of customers whose aggregated demand is as close to the budget as possible, although it is not allowed to exceed the budget. Formally, the problem is formulated as follows:

$$\begin{aligned} &\max_{\{d_i\}} \sum_{i=1}^{N} (d_i L_i) \\ &\text{s.t.} \sum_{i=1}^{N} (d_i L_i) \leq C_{SL}, \\ &\quad d_i \in \{1, 0\}. \end{aligned} \tag{1}$$

[5]Note that this VLC is used as a virtual module only for the purpose of illustration. However, it does not exist as a device/entity in the network. Its functionality can be carried out through mutual communication between the users.

To simplify the notation, unless explicitly stated, we shall denote the quantity $\sum_{i=1}^{N}(d_i L_i)$ as L_T.

Comparing the values of $\sum_{i=1}^{N} L_i$ and C_{SL}, we highlight the following variations of the problem:

i) Total shiftable demand is less than shiftable capacity:

$$\sum_{i=1}^{N} L_i \leq C_{SL}. \tag{2}$$

Clearly, in this scenario, since the capacity permitted is more than the demand, all the loads can be served by the source.

ii) Total shiftable demand is greater than shiftable capacity:

$$\sum_{i=1}^{N} L_i > C_{SL}. \tag{3}$$

This is the condition of greatest concern for both the supplier and the set of customers, because, clearly all the demands cannot be served by C_{SL}. Note that within this case, there is also a special case where the load of certain customers is greater[6] than the available shiftable capacity. Indeed, the scheme that we propose presently can also be applied to it.

The problem described in Case ii) above (except for the special case where $\forall i, L_i > C_{SL}$), is NP-Hard because it can be reduced to a subset-sum problem. As we know, for such an NP-Hard problem, the complexity increases dramatically when the number of customers increases. As the main focus of this study is the scheduling of aggregated shiftable loads of households in domestic networks between transformers and the households, in our case, the total number of customers is limited.

To solve this problem in a distributed manner, the customers need to communicate with each other and to send their respective decisions $\{d_i\}$ (i.e., their decision values of either 1 or 0 , meaning "YES" and "NO" respectively), along with their demands $\{L_i\}$. This process is carried out iteratively until a proper common consensus is attained on the usage of the available capacity for all customers, through each customer's individual decision. Once the common consensus is reached, the power source can provide the corresponding power accordingly. Thus, the interaction between the customers is modeled as a game, detailed presently.

B. AN EXAMPLE FOR LOAD SELECTION

The decision process modeled as a game can be explained by an example. Suppose there are 5 users with shiftable loads 1, 3, 4, 4 and 5 units respectively, and where the capacity of the SG available for shiftable loads at this timeslot is $C_{SL} = 10$ units.[7] The users will make a decision of either 1 or 0 repeatedly until a final decision is reached. Let us suppose that the initial set of decisions, made randomly, is $\{1, 1, 0, 1, 0\}$ respectively. These decisions from the set of 5 loads are shared among each other (or equivalently, sent to the VLC) that is aware of the fact that $C_{SL} = 10$ units. These decisions mean that only loads L_1, L_2 and L_4 are requesting to be served. This is not the optimal decision because it will not utilize the full capacity of 10 units as $d_1\, L_1 + d_2\, L_2 + d_4\, L_4 = 8$ units. Since this set of decisions is non-optimal, each user will either be rewarded or penalized collectively by the VLC for their good or bad decisions respectively. This Reward/Penalty is to be explained in next section. Based on the Reward/Penalty feedback responses and the previous decisions, each user will arrive at a new decision set using the corresponding LA. Let us assume that the new decision is $\{0, 1, 0, 1, 1\}$. This set of decisions leads to the new demand of $d_2\, L_2 + d_4\, L_4 + d_5\, L_5 = 12$ units. Again, a Reward/Penalty is calculated by the VLC and sent back to users. By repeating the process and the feedback between the users and the VLC, one can arrive at a final decision set, say $\{1, 0, 1, 0, 1\}$ which leads to the total load consumption of $\sum_{i=\{1,3,5\}} L_i d_i = 10$ units, which equals to C_{SL}. The consequence of this decision is that loads L_1, L_3 and L_5 will be served. Observe that another decision set $\{1, 0, 0, 1, 1\}$ is equally possible, and so if the algorithm could attain to either of these decisions, it will confirm that it is unbiased.

V. MODELING AND ANALYSIS OF THE GAME

The distributed decision-making problem can be formulated as a game denoted by $\mathfrak{G} = [I, \{d_i\}_{i\in I}, \{U_i\}_{i\in I}]$, where:

- I is the set of customers with shiftable loads $\{1, 2, 3, ...N\}$, with any specific customer being indexed by i.
- $\{d_i\}$ is the set of decision actions taken by the customers, i.e., $D = \{d_1, d_2, ..., d_N\}$, where $d_i \in D$ is the decision/action of customer i. Decision $d_i = 0$ represents the event that customer i does not turn on his load, while $d_i = 1$ represents the condition when customer i does turn it on.
- $\{U_i\}_{i\in I}$ is the utility function of user i, and can be expressed in terms of C_{SL} as in Eq. (4):

$$U_i(d_i, \mathbf{d}_{-i}) = \begin{cases} \dfrac{1}{L_i d_i + \sum\limits_{j\in I\setminus i} L_j d_j + C_{SL}}, & C_{SL} < L_i d_i + \sum\limits_{j\in I\setminus i} L_j d_j, \\ \dfrac{1}{C_{SL} - L_i d_i - \sum\limits_{j\in I\setminus i} L_j d_j}, & C_{SL} \geq L_i d_i + \sum\limits_{j\in I\setminus i} L_j d_j, \end{cases} \tag{4}$$

where $\mathbf{d}_{-i}$ denotes the set of decisions taken by users other than user i.

The utility function of an individual user is defined from the perspective of the overall system. More specifically, it is beneficial for a user if the sum of the loads based on the current decision of all the users approaches C_{SL}

[6]If $\exists i, s.t. L_i > C_{SL}$, our solution is applicable by excluding those users whose demands exceed the shiftable capacity. Thus, we will not elaborate on this scenario in any greater detail.

[7] In this example, since the number of users is so small, we can, almost trivially, see that there exists a combination of user demands that yields $\sum_{i=1}^{5} L_i d_i = C_{SL}$, which, of course, will not be possible if the number of users is large.

from the left, i.e., whenever the value approaches C_{SL} although it is less than or equal to it. Otherwise, the value of the utility function of each user is reduced.

The formulated game is an exact Potential Game [40] and the reasons are as follows: According to the definition of a Potential Game, the payoff of any player by changing its strategy can be expressed using a single global function, i.e., a so-called potential function. In this particular game, the utility function for each player is defined as a global function, which can be considered to be the potential function itself. Therefore, this game is indeed a Potential Game. Understandably, this game is an exact Potential Game because if a player switches from one action (decision) to another, the change in the potential equals to the change in the utility of that player [40]. From the properties of Potential Games, we see that the game has at least one pure strategy Nash Equilibrium (NE) point.

Based on the utility function U_i, the NE point of the game can be defined as follows. The solution point attained by the selection of the decisions $(d_1^*, d_2^*, ..., d_N^*)$ by the N users is a NE point of $\mathfrak{G}$ if no user can improve his utility function by deviating from it unilaterally. Formally, the NE point satisfies: $(d_i^*, \mathbf{d}_{-i}^*)$ as $U_i(d_i^*, \mathbf{d}_{-i}^*) \geq U_i(d_i, \mathbf{d}_{-i}^*)$, $\forall i \in I$ and $\forall d_i \in \{0, 1\} \backslash \{d_i^*\}$.

Before we proceed, we will demonstrate that a global optimal point of the problem is a NE point of the game, and *vice versa*. As a global optimal point, it is obvious that any unilateral change of decision of any user at the point will result in a decrease in the utility function. In other words, any user's deviation from 0 to 1 will result in grid overload because the sum of the loads will be over C_{SL}. Similarly, any change from 1 to 0 will move the sum of the loads farther away from C_{SL}, resulting in a decreased value of its utility function. Thus, the global optimal solution is, indeed, a NE point of $\mathfrak{G}$. Arguing in the converse manner, for any NE point, if it is not a global optimum, it implies that we could shift at least one user with decision 0, who was not already selected into the set of selected users, to decision 1, leading to a positive increase towards C_{SL}, which is, indeed, the consequence of a *unilateral* change. But such a unilateral user-shifting cannot occur because it contradicts the fundamental definition of the NE point. A similar argument can be presented when a single user's decision is changed from 1 to 0. We can, thus, conclude that the NE point is, indeed, a global optimal point in $\mathfrak{G}$, and *vice versa*.

In the next section, we shall propose a LA-based learning algorithm that assigns a LA to each customer in order to allow their collective decisions to converge to the game's NE point.

VI. IMPLEMENTATION OF LA IN DEMAND SCHEDULING

In our design, the users improve their decisions based on the rewards/penalties received for the decisions they made in previous iterations, and after sufficient number of iterations, users will, hopefully, converge to the NE point, which is the globally optimal solution of the problem.

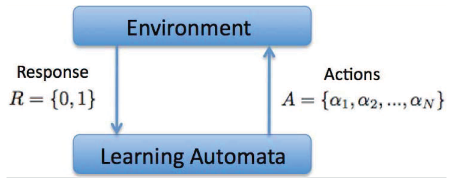


FIGURE 3. The schematic of how a LA interacts with the Environment.

A. DECISIONS OF USERS AND THEIR EFFECTS ON TOTAL LOAD

Figure 3 demonstrates a typical working scenario for a LA. It consists of a sequence of interaction cycles between the LA and its environment. In each iteration, the LA selects an action (α_i), which is either rewarded $(R = 1)$ or penalized $(R = 0)$ by the environment as a response.

Based on the response and the knowledge acquired from the previous iterations, the LA adjusts its strategy of selecting actions in order to make a "wiser" decision in the next iteration. The optimal action is the one with the largest reward probability. The way by which the LA adjusts its strategy depends on whether it has a fixed or variable structure, and if the actions are chosen as per an action probability rule. The latter action probability rule is updated based on the Reward/Penalty response that was received from the Environment, and/or the estimates of the set of reward probabilities obtained from the previous responses. The goal of a LA is to maximize the total probability of receiving rewards during and subsequent to the learning.

The most difficult part of designing a LA-based solution for a new application domain is that of determining what the "Environment" is, and then of knowing how the LA itself is "Rewarded" or "Penalized".

In our specific SG-based domain, since a users' decision d_i is either 0 and 1, the load for this user will be either 0 or L_i respectively, and so the total load "L_T" can be calculated by summing up these individual contributions in each iteration. Every iteration yields a new value of L_T. The decision-making process will go through an iterative process so that "L_T" will approach the global optimal C_{SL}. To capture the number of iterations, we denote a new index, $s \in \{1, 2, ..., M\}$ for the number of iterations, where M is the maximum number of iterations permitted. Correspondingly, the decision of user i at the iteration s is denoted by $d_i(s)$, $i \in \{1, 2, ..., N\}$. Similarly, we denote $L_T(s)$ as the current value of the total of the load values at iteration s. Obviously, the value of $L_T(s)$ differs as the values of the decisions $\{d_i(s)\}$ change. As the aim of the game is to achieve a value that is as close as possible to C_{SL} after every iteration, our task is to define the current Reward/Penalty so as to guide the users' decision-making process towards the optimal point.

B. CALCULATION OF REWARD AND PENALTY

We shall now consider the intricate problem of determining when the LA should be rewarded or penalized.

In order to reach the closest possible value of C_{SL}, it is beneficial if the value of $L_T(s)$ approaches C_{SL} but, at the same time, that it is less than or equal to C_{SL}, as the iterations proceed. In this case, a Reward is applied to all the users. In more details, a Reward is given if $L_T(s)$ is less than or equal to C_{SL} and at the same time larger than or equal to C_{max}, where C_{max} maintains the maximum of the valid[8] sum of loads obtained until the last iteration. Otherwise, a Penalty is applied (i.e., when $L_T(s)$ is either greater than C_{SL}, or less than the maximum valid load sum C_{max}). The procedure for deciding on a Reward/Penalty is formally outlined in Algorithm 1.

Algorithm 1 Reward/Penalty Assignments

Input:

- The loads of all the users and their decisions, $\{d_i(s)\}$ at a time instant, s.
- The maximum of the valid sum of loads among all the previous iterations, C_{max}.

Output:

- The assignment of a Reward or a Penalty to all users at the time instant, s.

```
1:  begin
2:     for every user $i$ do
3:        Calculate $L_T(s) = \sum_{i=1}^{N} d_i(s)L_i$ based on the information obtained from the other users.
4:        if $L_T(s) \leq C_{SL}$ and $L_T(s) \geq C_{max}$ then
5:           Decision $d_i(s)$ leads to a Reward to user $i$.
6:           $C_{max} = L_T(s)$.          ▷ Update the load sum.
7:        else
8:           Decision $d_i(s)$ leads to a Penalty to user $i$.
9:        end if
10:    end for
11: end
```

Algorithm 1 is carried out for every decision-making iteration and stopped when the decision-making process ends. This termination phase will be discussed presently. Note that by embarking on this mutual information sharing, each user will be able to *individually* calculate the Reward or Penalty that *he* receives.

C. DECISION MAKING ON THE ACTIONS IN THE ITERATION

Once the Reward/Penalty for each user has been assigned, we need to specify a learning scheme for deciding the action (0 or 1) that he has to make in the next iteration. As mentioned earlier, in this work, we opt to use LA to achieve this, and in this regard, we select three well-established LA to do this learning, namely, the LRI scheme [16], the CLA [20], and the more-recently introduced BLA [21].

1) DECISION MAKING FOR THE LRI

The way the decisions are made for the LRI scheme, is straightforward. The action that each LA makes is based on the action selection probability. Each LA maintains two parameters p_0 and p_1 representing the probability of selecting 0 and 1 respectively, with $p_0 + p_1 = 1$. The quantities are initialized to 0.5. If the chosen action (either 0 or 1) is rewarded, the probability of the alternate action (p_1 or p_0 respectively) is decreased using a user-defined parameter, λ, and thus the probability associated with the chosen action is increased. The LA keeps the action probabilities unchanged in the case of a penalty feedback.

The formal algorithmic version is straightforward, and is thus omitted. We mention, though, that the actual decision communicated to the SG's CC will occur after a final decision is attained by the LA-based scheme, and concluded by the stopping criteria specified in Section VI-D for the corresponding time slot.

2) DECISION MAKING FOR THE CLA

The CLA is similar to the LRI scheme due to the fact that it involves the action probabilities and a learning parameter (denoted by λ), whose value affects the convergence speed and the proximity of the final solution to the optimal point. The CLA-based scheme is different from the LRI scheme (and the BLA) due to the fact that it explicitly uses a *continuous* utility function in the update equation. It is based on the work of Mason on LA with a continuous feedback response [20]. The rationale for using this type of LA is the fact that the utility function is continuous. In our particular problem, we need a normalization of the values of the utility in order to ensure that the feedback is in the interval [0, 1] [20]. These steps are specified in Lines 13-32 in the Algorithm 2 below.

Using such a mapping and updating rule, it is possible to prove that the CLA will converge to the pure equilibrium of the game with a probability that approaches unity, as the update parameter is made arbitrarily small. This is a consequence of the work due to Sastry *et al.* [41] since the NE of our Potential Game corresponds to the mode of the payoff matrix.

The formal steps in the CLA-based decision making process is detailed in Algorithm 2.

Again, the actual decision communicated to the SG's CC will occur after a final decision is attained by the CLA-based scheme, and concluded by the stopping criteria specified in Section VI-D for the corresponding time slot.

3) BLA BASED DECISION MAKING PROCESS

In the BLA-based learning scheme, each LA maintains two hyper-parameters $a_{i,j}$ and $b_{i,j}$. These are introduced to count the number of rewards and penalties respectively, where index i is the index of the user and $j \in \{0, 1\}$ denotes the decisions that the LA has made. In each iteration, the LA makes a decision about the choice of the action. Thereafter, the value of $a_{i,j}$ is increased if the decision leads to a reward, and the

[8]Here "valid" means that $C_{max} \leq C_{SL}$ holds.

value of $b_{i,j}$ is increased if the decision leads to a penalty, as formalized in Table 2.

Algorithm 2 CLA-based Decision Making

Input:

- Initialize $s = 0, p_{i,0} = p_{i,1} = 0.5$, where i is the index of users. $MaxSoFar = 0.01$.
- Declare variables $Utility$ and $NormUtility$ where $Utility$ gives the utility function of the corresponding iteration, $MaxSoFar$ is the maximum utility value until the current iteration, and $NormUtility$ is the ratio of $Utility$ and $MaxSoFar$.

Output:

- Sequence of decisions made as per the CLA.
- Sequence of Action Probability values.

```
1:  begin
2:      $C_{max} = 0$. ▷ Maintain the maximum valid load sum.
3:      repeat
4:          for every user $i$ do
5:              Generate a random number $Random_i = uniform(0, 1)$.
6:              if $p_{i,0}$ of user $i \leq Random_i$ then
7:                  Decision $d_i(s) = 0$.
8:              else
9:                  Decision $d_i(s) = 1$.
10:             end if
11:         end for
12:         Calculate the Reward/Penalty received due to decision as per Step 3-9 of Algorithm 1.
13:         for every user $i$ do
14:             if Reward is received then
15:                 $Reward = true$.
16:             end if
17:             Calculation of $Utility$.
18:             if $Utility \geq MaxSoFar$ then
19:                 $MaxSoFar = Utility$.
20:             end if
21:             $NormUtility = \frac{Utility}{MaxSoFar}$.
22:             if $Reward$ then
23:                 if $d_i(s) = 0$ then
24:                     $p_{i,0} = p_{i,0} + \lambda \cdot NormUtility \cdot (1 - p_{i,0})$.
25:                     $p_{i,1} = p_{i,1} - \lambda \cdot NormUtility \cdot p_{i,1}$.
26:                 end if
27:                 if $d_i(s) = 1$ then
28:                     $p_{i,1} = p_{i,1} + \lambda \cdot NormUtility \cdot (1 - p_{i,1})$.
29:                     $p_{i,0} = p_{i,0} - \lambda \cdot NormUtility \cdot p_{i,0}$.
30:                 end if
31:             end if
32:         end for
33:         $s = s + 1$.
34:     until Either Stopping criterion in Section VI-D satisfied.
35: end
```

We detail the BLA-based algorithm for each user i in Algorithm 3.

TABLE 2. The effect of the Reward/Penalty responses on the BLA's decision (for user i at iteration s) on its parameters.

	$d_i(s)$=1	$d_i(s)$=0
Reward	$a_{i,1} = a_{i,1} + 1$	$a_{i,0} = a_{i,0} + 1$
Penalty	$b_{i,1} = b_{i,1} + 1$	$b_{i,0} = b_{i,0} + 1$

Algorithm 3 BLA-based Decision Making

Input:

- Initialize $s = 0, a_{i,0} = b_{i,0} = a_{i,1} = b_{i,1} = 1$ where s is the index of iterations.

Output:

- Sequence of decisions made as per the BLA.
- Sequence of values of $\{a_i\}$ and $\{b_i\}$ for all the users.

```
1:  begin
2:      $C_{max} = 0$. ▷ Maintain the maximum valid load sum.
3:      repeat
4:          $s = s + 1$.
5:          for each user $i$ do
6:              draw $x_{i,0}, x_{i,1}$, from Beta distribution given by $\beta(a_{i,0}, b_{i,0})$ and $\beta(a_{i,1}, b_{i,1})$, where:
                $$\beta(a_{i,j}, b_{i,j}) = \frac{\int_0^{x_{i,j}} v^{(a_{i,j}-1)}(1-v)^{(b_{i,j}-1)}dv}{\int_0^1 u^{(a_{i,j}-1)}(1-u)^{(b_{i,j}-1)}du}. \quad (5)$$
7:              if $x_{i,0} \leq x_{i,1}$ then
8:                  Decision $d_i(s) = 1$.
9:              else
10:                 Decision $d_i(s) = 0$.
11:             end if
12:         end for
13:         Based on $d_i(s)$, calculate the Reward/Penalty received as per Step 3-9 of Algorithm 1.
14:         for each user $i$ do
15:             if (Reward is received) then
16:                 if ($d_i(s) = 0$) then
17:                     $a_{i,0} = a_{i,0} + 1$.
18:                 else
19:                     $a_{i,1} = a_{i,1} + 1$.
20:                 end if
21:             else                          ▷ i.e., Penalty is received
22:                 if ($d_i(s) = 0$) then
23:                     $b_{i,0} = b_{i,0} + 1$.
24:                 else
25:                     $b_{i,1} = b_{i,1} + 1$.
26:                 end if
27:             end if
28:         end for
29:     until Either Stopping criterion in Section VI-D satisfied.
30: end
```

D. STOPPING CRITERIA FOR THE NUMBER OF ITERATIONS

The problem that we are studying is NP-Hard. It is, probably, thus not possible to attain to the optimal solution in polynomial time. The best that we can hope for using heuristic solutions is an approximately-optimal solution as the search for the truly-optimal solution, in a reasonable amount of time, may be infeasible. To enable the algorithm to terminate with a reasonably-accurate solution, and to prevent it from looping unnecessarily, we have opted to include a terminating condition. The algorithm will stop when one of the following criteria is satisfied:

1) If the number of iterations has reached maximum, i.e., $s = M$, or
2) The decisions for all the users, that lead to a Reward response, remains unchanged for a certain number of iterations, Z (implying that the iteration has converged). In this case, we know that we have attained a solution that is, with a very high likelihood, a near-optimal one.

These criteria can be applied to any of the LA-based solutions proposed in Section VI-C.

E. OVERALL LA-BASED SG ALGORITHM

Till now, the distributed LA-based load scheduling algorithm has been explained step-by-step. We shall now explain how the LA-based algorithms fit into the overall scheme of things. To do this, we shall use the BLA to encapsulate the learning process, although the exact same principles will be true for the other two LA we have described above.

In the beginning of time slot t, the power source will announce its power budget for the shiftable loads to all users. Once the users receive this budget, they will check if the budget is sufficient to accommodate all the users' demands via mutual communications. If all their loads can be accommodated, the learning process is not necessary and the users can just turn on their loads. If their cumulative loads cannot be accommodated, they will initiate the learning process so that they will be able to converge to a combination that will yield the closest value to the budget. In each iteration, each user will broadcast its power demand and current decision to all the other users. Based on this mutual information sharing, all the users will possess a global view of the game so that each of them can decide on their rewards/penalties, and make a decision for the next iteration. When the stopping criteria is fulfilled at any iteration, the iterations for that time slot will terminate and the users who decide "Yes" (i.e., who converge to the action '1') will turn on their loads. One must observe that during the process, the information is exchanged between their users only, and the power appliances will not be turned on or off until the iteration terminates.

The sequence of steps mentioned above is formalized below in Algorithm 4.

Algorithm 4 The Overall SG Algorithm

Input:
- The total load that the SG can provide and the individual loads of the users.
- The parameters for the Stopping Criteria.

Output:
- The users whose loads can be turned on.

```
1:  begin
2:      Every user broadcasts his demand to all the other users.
3:      if $\sum_{i=1}^{N} L_i < C_{SL}$ then
4:          All the users can turn their loads on.
5:          Exit.
6:      end if
7:      Initialize $s = 0$, $a_{i,j} = b_{i,j} = 1$, $C_{max} = 0$.
8:      $s = s + 1$.
9:      for each user $i$ do
10:         Draw a random value $x_{i,j}$ from $\beta(a_{i,j}, b_{i,j})$ obeying Eq. (5).
11:         $x_{i,0}$ from $\beta(a_{i,0}, b_{i,0})$ for $j = 0$.
12:         $x_{i,1}$ from $\beta(a_{i,1}, b_{i,1})$ for $j = 1$.
13:         if $x_{i,0} < x_{i,1}$ then
14:             Decision $d_i(s) = 1$.
15:         else
16:             Decision $d_i(s) = 0$.
17:         end if
18:         Broadcast the user's decision to all other users.
19:     end for
20:     for each user $i$ do
21:         Calculate $L_T(s) = \sum_{i=1}^{N}(d_i(s)L_i)$.
22:         if $L_T(s) \leq C_{SL}$ and $L_T(s) \geq C_{max}$ then
23:             Decision $d_i(s)$ leads to a Reward to user $i$.
24:             $C_{max} = L_T(s)$.
25:         else
26:             $d_i(s)$ leads to a Penalty to user $i$.
27:         end if
28:         if (Reward is received) then
29:             if ($d_i(s) = 0$) then $a_{i,0} = a_{i,0} + 1$.
30:             else  $a_{i,1} = a_{i,1} + 1$.
31:             end if
32:         else                              ▷ i.e., Penalty is received
33:             if ($d_i(s) = 0$) then $b_{i,0} = b_{i,0} + 1$.
34:             else  $b_{i,1} = b_{i,1} + 1$.
35:             end if
36:         end if
37:     end for
38:     if Either Stopping Criterion Satisfied then
39:         The users who have "converged" to 1 turn on their loads.
40:     else
41:         Go to Step 8 and repeat the BLA choice.
42:     end if
43: end
```

VII. SIMULATION AND EXPERIMENTAL RESULTS

To evaluate the performance of the LA-based schemes, we carried out simulations[9] on numerous SGs, where the number of users and the parameters were varied. However, in the interest of brevity and space, we merely cite the results obtained from a subset of these experiments.[10] The experiments were conducted to capture two important

[9]The simulations were done in MATLAB, but on "real-life" data, as explained below.

[10]Additional results can be seen from the Masters' thesis of the First Author, and can be made available on request.

metrics, namely, the accuracy of the convergence of the scheme used, and its speed of the convergence.

A. THE DATA SETS

The simulation configuration was derived based on the real-life measurements of the electricity consumption for 28 domestic users [42]–[44], as shown in Table 3.

TABLE 3. This table lists the demands of 28 user (in KWh) used in our experiments.

L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}
242	146	131	111	97	95	92	82	75	74
L_{11}	L_{12}	L_{13}	L_{14}	L_{15}	L_{16}	L_{17}	L_{18}	L_{19}	L_{20}
74	74	71	59	57	55	51	49	42	41
L_{21}	L_{22}	L_{23}	L_{24}	L_{25}	L_{26}	L_{27}	L_{28}		
39	37	35	35	31	15	11	11		

A word about how the data in Table 3 was obtained is not out of place. Specifically, the annual power consumption in [42] was converted to yield the average consumption for every 15 minutes timespan [43] considering the scheduling interval in real life. Half of these customers' demands were considered as shiftable loads [44]. In our simulation, we considered the scenario where only a subset of all the users could be selected for the given capacity, i.e., $0 < C_{SL} < \sum L_i$. Although the shiftable capacity of the SG and the shiftable demands were subject to change due to various reasons [34], without loss of generality, we assumed that the capacity of the shiftable load, C_{SL}, was about 70% of the total demand for the shiftable load.

B. CONVERGENCE TO 0 AND 1 DECISIONS

To illustrate how the learning process takes place, consider Figure 4. This figure illustrates the respective decisions, between 1 and 0 given in y-axis, made by a specific user plotted as a function of the number of iterations for a single trial in the BLA-based algorithm.

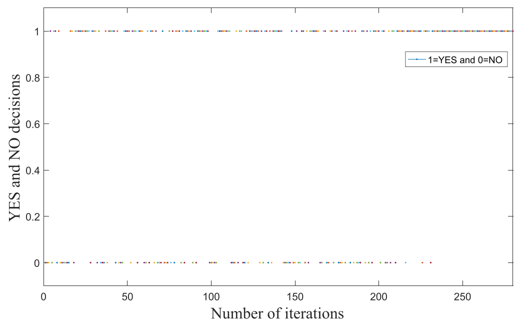


FIGURE 4. The pattern of 1 and 0 decisions for a specific user who is playing the game.

From this figure, we can clearly see that before the 240th iteration, the decision of the user changed between the decisions of 1 and 0, implying that the user was able to "learn" from the environment by trying (using the BLA scheme) different options. After 240 iterations, the user converged to the decision 1 as it had consecutively made decision 1 from the 240th iteration onwards.

Figure 5 depicts the same phenomenon but from a different perspective as the y-axis indicates the number of decisions of 1 and 0 that were made as the iteration continued. Before 240 iterations, the number of decisions of 0 and 1 increased monotonically. On the other hand, after this juncture, the number of 0's stopped increasing because the decision had converged to 1. Indeed, after a certain number of consecutive decision of 1, the second stopping criteria of Section VI-D was fulfilled, and so the user terminated the iteration and froze its decision. A similar convergence procedure can be observed for CLA and LRI-based learning algorithms, but are not included here to avoid repetition.

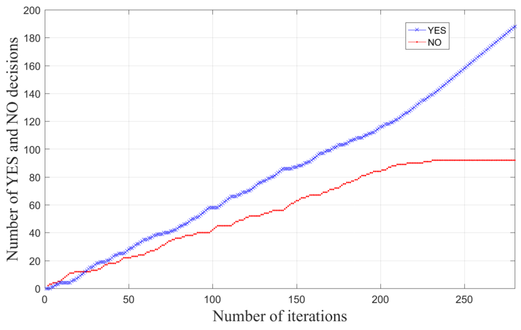


FIGURE 5. The number of 1 and 0 decisions for a specific user who is playing the game.

TABLE 4. Simulation results from the BLA, CLA and *LRI*-based algorithms with $i = 28$ and $C_{SL} = 1352$ KWh.

Method	Sum of selected loads		Iterations	
BLA	1351.998		20306	
	LRI	CLA	LRI	CLA
λ=0.01	-	-	300000+	300000+
λ=0.03	1352.000	1352.000	136737	141411
λ=0.05	1352.000	1352.000	60045	61853
λ=0.08	1352.000	1352.000	25690	26986
λ=0.10	1352.000	1352.000	16906	17863
λ=0.12	1352.000	1352.000	11458	12231
λ=0.13	1352.000	1352.000	9091	10259
λ=0.14	1352.000	1352.000	7797	8460
λ=0.15	1352.000	1352.000	6781	7517
λ=0.16	1351.998	1352.000	5391	6409
λ=0.17	1351.998	1351.998	4772	5333
λ=0.2	1351.946	1351.988	2933	3417
λ=0.3	1349.608	1351.408	581	911
λ=0.4	1335.400	1348.530	181	300
λ=0.5	1289.444	1342.843	92	135

C. AVERAGE CONVERGENCE CHARACTERISITICS

To illustrate the average number of iterations before convergence and the average value of the total selected power demands, we present the simulation results of the experiments in Tables 4 and 5. Table 4 illustrates the simulation results with all 28 users while Table 5 summarizes the results when the last 15 users in Table 3 are used. All the results presented in these tables are averaged values of over an ensemble of 400 independent replications. For the cases of the CLA

TABLE 5. Simulation results from the BLA, CLA and *LRI*-based algorithms with $i = 15$ and $C_{SL} = 300$ KWh.

Method	Sum of selected loads		Iterations	
BLA	299.872		1101	
	LRI	CLA	LRI	CLA
λ=0.01	-	-	300000+	300000+
λ=0.06	300.000	300.000	5764	6082
λ=0.07	300.000	300.000	4600	4596
λ=0.08	300.000	300.000	3763	3767
λ=0.09	300.000	300.000	3056	3148
λ=0.10	299.997	299.997	2630	2537
λ=0.11	299.990	299.995	2168	2139
λ=0.12	299.985	299.995	1772	1930
λ=0.13	299.975	299.970	1509	1528
λ=0.14	299.975	299.975	1357	1496
λ=0.16	299.917	299.947	1010	1142
λ=0.20	299.785	299.722	613	612
λ=0.30	298.215	298.347	223	222
λ=0.40	294.497	294.882	114	113
λ=0.50	285.892	289.745	76	78

and the LRI, the results are illustrated for different values of λ. As opposed to this, since the BLA does not depend on any parameter, the results for the BLA are presented, in those tables, in a single line.

As an overall observation, we mention that for the parameter-free BLA-based algorithm, the average value of the selected load is very close to the optimal point in both tables. This indicates that the convergence accuracy of the algorithm is close to the NE point. Further, unlike the other LA-based schemes, there is no speed/accuracy conflict. Understandably, the average number of iterations, before convergence, for the scenario when there are 28 users is much more than the corresponding figure that we encounter when we deal with only 15 users. The reason is for this is obvious: Because the problem is NP-Hard, the complexity of the problem increases super-linearly with the number of users. That being said, it is still worth mentioning that the BLA can efficiently solve the problem after a fairly reasonable number of iterations.

The performances of the LRI and the CLA depend fundamentally on the value of the LA's parameter, λ. With a sufficiently small value for λ, the algorithms can converge to the NE point with high precision at a cost of executing a large number of iterations. Of course, the number of iteration is smaller when λ is relative large, but the convergence accuracy is compromised. If we compare the λ-dependent schemes (i.e., the CLA and the LRI) with the BLA-based algorithms, the number of iterations for the former were smaller than that for the latter, i.e., if the average value of the selected loads was almost identical. For example, when $\lambda = 0.16$ for $i = 15$, the average load was 299.917 after 1,010 iterations for LRI, and 299.947 after 1,142 iterations for the CLA. As opposed to this, the BLA yielded a lower load value of 299.872 after 1,101 iterations. Interestingly, the traditional age-old LRI with $\lambda = 0.16$ was superior to the CLA and the BLA in such a configuration. Similarly, when $i = 28$, both the CLA and the LRI with the parametric setting of $\lambda = 0.17$ yielded a better performance than the BLA. Comparing the λ-dependent schemes, arguably the LRI yielded a slightly better performance than the CLA in most cases, as the CLA needed more iterations to converge.

Although, as demonstrated by the results presented in the tables, comparatively smaller values of λ led to a superior performance for the LRI and the CLA, than for the BLA, the λ values could be quite different depending on the system's configurations. Consequently, the issue of determining the ideal value of λ was mandatory for a certain system configuration, whenever the LRI or the CLA was applied. However, the BLA-based approach did not require the setting of any *a priori* configurations, which renders it to be a more practical option in this application domain.

The general conclusion of our experiments can be stated as follows:

- If one wants a "quick-and-dirty" solution to the problem, it is better to use a large value of λ for either of the λ-dependent schemes, and to converge to a reasonably accurate solution, very quickly. Thus, as we can see from Table 5, we can converge to a fairly good solution in less than 80 iterations by using a value of λ that is as large as 0.50.
- If one wants to get a fine and accurate solution and has the time available to test the SG using different λ-dependent schemes, it is advisable to use various values of λ during a training-like phase, and to then use the best value of λ in the actual real-life execution of the schemes.
- Finally, if one is not willing to go through a rigorous phase where the schemes are tested for various values of λ, we recommend that one utilizes the BLA scheme, which, being parameter-free, is best suited for such scenarios.

VIII. CONCLUSIONS AND FUTURE WORK

In this work, we have studied the problem of the scheduling of loads for domestic users in Smart Grids (SGs) in a distributed manner. This load scheduling problem is NP-Hard, and the distributed scheduling process is formulated as an exact Potential Game that has at least one pure strategy NE point. In this paper, we proposed a LA-based algorithm which utilized three distinct LA alternatives, i.e., the LRI, the CLA and the BLA. Each of the multiple users utilized a LA to achieve the decision making process, and to thus solve the problem in a distributed manner. The simulations results show that the proposed approaches converge to a solution close to the NE point of the game, which is also the global optimal point. The advantage of the BLA-based approach is that without any pre-configuration (i.e., the setting of any user-defined parameter), the algorithm can converge to the NE point with a very high probability. As opposed to this, the LRI and CLA-based approaches possess the advantage that their accuracy and the number of iterations can be compromised by tuning the learning parameter λ. Otherwise, the convergence of all the schemes is comparable.

A. FUTURE WORK

In spite of the fact that our schemes permits a randomized schedule, there is no guarantee that this schedule is "fair" over time. This is because there is the possibility that some customers might randomly get favored. A possible research direction is to consider providing fairness to the scheduling process by considering the *history* of the assignments. One possible solution would be to alter the initial probabilities of the actions, so as to increase the likelihood of the system to converge to one of the actions. Another option would be to decide, at each epoch, which customers are allowed to enter the game and to exclude the rest.

Another research direction worth investigating when facing a large-scale game with a large number of customers, is that of dividing the customers into k groups where each group plays exactly the same game with a uniform constraint on the load $\frac{C_{SL}}{k}$. Based on the concept of creating groups and of dividing the loads between the groups, one could devise more sophisticated approaches that do not necessarily distribute the load in a uniform manner. Finally, it is also possible to allow communication between these groups themselves so that the groups that have complementary loads can enter into alliances and be scheduled simultaneously.

ACKNOWLEDGMENT

This paper was presented in part at the 2nd EAI International Conference on Smart Grid Inspired Future Technologies, London, U.K., in 2017.

REFERENCES

[1] R. Thapa, L. Jiao, B. J. Oommen, and A. Yazidi, "Scheduling domestic shiftable loads in smart grids: A learning automata-based scheme," in *Proc. Int. Conf. Smart Grid Inspired Future Technol. (SmartGIFT)*, Mar. 2017, pp. 58–68.

[2] H. Farhangi, "The path of the smart grid," *IEEE Power Energy Mag.*, vol. 8, no. 1, pp. 18–28, Jan./Feb. 2010.

[3] V. C. Gungor *et al.*, "Smart grid technologies: Communication technologies and standards," *IEEE Trans. Ind. Informat.*, vol. 7, no. 4, pp. 529–539, Nov. 2011.

[4] V. C. Gungor, B. Lu, and G. P. Hancke, "Opportunities and challenges of wireless sensor networks in smart grid," *IEEE Trans. Ind. Electron.*, vol. 57, no. 10, pp. 3557–3564, Oct. 2010.

[5] *Semiah: Scalable Energy Management Infrastructure for Aggregation of Households*. Accessed: Sep. 17, 2016. [Online]. Available: http://semiah.eu/

[6] A. J. Conejo, J. M. Morales, and L. Baringo, "Real-time demand response model," *IEEE Trans. Smart Grid*, vol. 1, no. 3, pp. 236–242, Dec. 2010.

[7] M. Roozbehani, M. Dahleh, and S. Mitter, "Dynamic pricing and stabilization of supply and demand in modern electric power grids," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Oct. 2010, pp. 543–548.

[8] A.-H. Mohsenian-Rad and A. Leon-Garcia, "Optimal residential load control with price prediction in real-time electricity pricing environments," *IEEE Trans. Smart Grid*, vol. 1, no. 2, pp. 120–133, Sep. 2010.

[9] T. Logenthiran, D. Srinivasan, and T. Z. Shun, "Demand side management in smart grid using heuristic optimization," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1244–1252, Sep. 2012.

[10] A.-H. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, and R. Schober, "Optimal and autonomous incentive-based energy consumption scheduling algorithm for smart grid," in *Proc. Innov. Smart Grid Technol. (ISGT)*, Jan. 2010, pp. 1–6.

[11] A.-H. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Trans. Smart Grid*, vol. 1, no. 3, pp. 320–331, Dec. 2010.

[12] P. Chavali, P. Yang, and A. Nehorai, "A distributed algorithm of appliance scheduling for home energy management system," *IEEE Trans. Smart Grid*, vol. 5, no. 1, pp. 282–290, Jan. 2014.

[13] Z. Fan, "A distributed demand response algorithm and its application to PHEV charging in smart grids," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1280–1290, Sep. 2012.

[14] W. Saad, Z. Han, H. V. Poor, and T. Basar, "Game-theoretic methods for the smart grid: An overview of microgrid systems, demand-side management, and smart grid communications," *IEEE Signal Process. Mag.*, vol. 29, no. 5, pp. 86–105, Sep. 2012.

[15] M. A. L. Thathachar and B. R. Harita, "Learning automata with changing number of actions," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 17, no. 6, pp. 1095–1100, Nov. 1987.

[16] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. North Chelmsford, MA, USA: Courier Corporation, 2012.

[17] N. Kumar, S. Misra, and M. S. Obaidat, "Routing as a Bayesian coalition game in smart grid neighborhood area networks: Learning automata-based approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jan. 2014, pp. 1502–1507.

[18] S. Misra, P. V. Krishna, V. Saritha, H. Agarwal, and A. Ahuja, "Learning automata-based multi-constrained fault-tolerance approach for effective energy management in smart grid communication network," *J. Netw. Comput. Appl.*, vol. 44, pp. 212–219, Sep. 2014.

[19] S. Misra, P. V. Krishna, V. Saritha, and M. S. Obaidat, "Learning automata as a utility for power management in smart grids," *IEEE Commun. Mag.*, vol. 51, no. 1, pp. 98–104, Jan. 2013.

[20] L. G. Mason, "An optimal learning algorithm for S-model environments," *IEEE Trans. Autom. Control*, vol. AC-18, no. 5, pp. 493–496, Oct. 1973.

[21] O.-C. Granmo and S. Glimsdal, "Accelerated Bayesian learning for decentralized two-armed bandit based decision making with applications to the Goore Game," *Appl. Intell.*, vol. 38, no. 4, pp. 479–488, 2013.

[22] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid—The new and improved power grid: A survey," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 944–980, 4th Quart., 2012.

[23] W. Wang, Y. Xu, and M. Khanna, "A survey on the communication architectures in smart grid," *Comput. Netw.*, vol. 55, no. 15, pp. 3604–3629, 2011.

[24] K. Moslehi and R. Kumar, "Smart grid—A reliability perspective," in *Proc. Innov. Smart Grid Technol. (ISGT)*, 2010, pp. 1–8.

[25] Q. Qdr, "Benefits of demand response in electricity markets and recommendations for achieving them: A report to the united states congress pursuant to section 1253 of the energy policy act of 2005," U.S. Dept. Energy, Washington, DC, USA, Tech. Rep., 2006.

[26] M. Parvania and M. Fotuhi-Firuzabad, "Demand response scheduling by stochastic SCUC," *IEEE Trans. Smart Grid*, vol. 1, no. 1, pp. 89–98, Jun. 2010.

[27] P. Palensky and D. Dietrich, "Demand side management: Demand response, intelligent energy systems, and smart loads," *IEEE Trans. Ind. Informat.*, vol. 7, no. 3, pp. 381–388, Aug. 2011.

[28] S. K. Vuppala, K. Padmanabh, S. K. Bose, and S. Paul, "Incorporating fairness within demand response programs in smart grid," in *Proc. IEEE Innov. Smart Grid Technol. (ISGT)*, Jan. 2011, pp. 1–9.

[29] X. Liang, X. Li, R. Lu, X. Lin, and X. Shen, "UDP: Usage-based dynamic pricing with privacy preservation for smart grid," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 141–150, Mar. 2013.

[30] K. Mets, T. Verschueren, W. Haerick, C. Develder, and F. de Turck, "Optimizing smart energy control strategies for plug-in hybrid electric vehicle charging," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. Workshops*, Apr. 2010, pp. 293–299.

[31] J. Zheng, D. W. Gao, and L. Lin, "Smart meters in smart grid: An overview," in *Proc. IEEE Green Technol. Conf.*, Apr. 2013, pp. 57–64.

[32] I. Koutsopoulos and L. Tassiulas, "Optimal control policies for power demand scheduling in the smart grid," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 6, pp. 1049–1060, Jul. 2012.

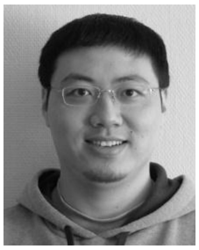
[33] K. Kursawe, G. Danezis, and M. Kohlweiss, "Privacy-friendly aggregation for the smart-grid," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2011, pp. 175–191.

[34] T. Batterberry, M. Miller, K. Jaskolka, and R. Toll, "Smart grid price response service for dynamically balancing energy supply and demand," U.S. Patent 0 138 363 A1, Jun. 3, 2010.

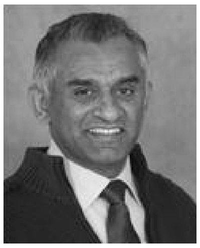
[35] H. Wang, T. Huang, X. Liao, H. Abu-Rub, and G. Chen, "Reinforcement learning in energy trading game among smart microgrids," *IEEE Trans. Ind. Informat.*, vol. 63, no. 8, pp. 5109–5119, Aug. 2016.

[36] S. Bakr and S. Cranefield, "Optimizing shiftable appliance schedules across residential neighbourhoods for lower energy costs and fair billing," in *Proc. 2nd Australasian Workshop Artif. Intell. Health (AIH)*, 2013, pp. 45–52.
[37] Y. Liu, N. U. Hassan, S. Huang, and C. Yuen, "Electricity cost minimization for a residential smart grid with distributed generation and bidirectional power transactions," in *Proc. IEEE Innov. Smart Grid Technol. (ISGT)*, Feb. 2013, pp. 1–6.
[38] Z. Zhu, J. Tang, S. Lambotharan, W. H. Chin, and Z. Fan, "An integer linear programming based optimization for home demand-side management in smart grid," in *Proc. IEEE Innov. Smart Grid Technol. (ISGT)*, Jan. 2012, pp. 1–5.
[39] T. Kato, K. Yuasa, and T. Matsuyama, "Energy on demand: Efficient and versatile energy control system for home energy management," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Oct. 2011, pp. 392–397.
[40] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behavior*, vol. 14, no. 1, pp. 124–143, 1996.
[41] P. S. Sastry, V. V. Phansalkar, and M. Thathachar, "Decentralized learning of Nash equilibria in multi-person stochastic games with incomplete information," *IEEE Trans. Syst., Man and*, vol. 24, no. 5, pp. 769–777, May 1994.
[42] J.-P. Zimmermann *et al.*, "Household electricity survey: A study of domestic electrical product usage," Intertek Testing Certification Ltd, Milton Keynes, U.K., Tech. Rep. R66141, 2012.
[43] H.-L. Chao, C.-C. Tsai, P.-A. Hsiung, and I.-H. Chou, "Smart grid as a service: A discussion on design issues," *Sci. World J.*, vol. 2014, Aug. 2014, Art. no. 535308.
[44] S. Rajakaruna, F. Shahnia, and A. Ghosh, *Plug in Electric Vehicles in Smart Grids*. Springer, 2015.

RAJAN THAPA received the bachelor's degree in electronics and communication engineering from Tribhuwan University, Nepal, and the master's degree in information and communication technology from the University of Agder, Norway, in 2010 and 2016, respectively. He was a site engineer at mobile company in Nepal for two years.



LEI JIAO received the B.E. degree in telecommunication engineering from Hunan University and the M.E. degree in communication and information system from Shandong University, China, in 2005 and 2008, respectively, and the Ph.D. degree in information and communication technology from the University of Agder, Norway, in 2012. He is currently an Associate Professor with the Department of Information and Communication Technology, University of Agder. His research interests include resource allocation in wireless communications, smart grid, and artificial intelligence.



B. JOHN OOMMEN (F'03) was born in Coonoor, India, in 1953. He received the B.Tech. degree from IIT Madras, India, in 1975, the M.E. degree from the Indian Institute of Science, Bengaluru, India, in 1977, and the M.S. and Ph.D. degrees from Purdue University, West Lafayettte, IN, USA, in 1979 and 1982, respectively. He joined the School of Computer Science, Carleton University, Ottawa, ON, Canada, from 1981 to 1982, where he is currently a Full Professor. He has authored over 455 refereed journals and conference publications. His research interests include automata learning, adaptive data structures, statistical and syntactic pattern recognition, stochastic algorithms, and partitioning algorithms. He is a fellow of an IAPR. Since 2006, he has held the honorary rank of Chancellor's Professor, which is a lifetime award from Carleton University. He has served on the Editorial Board for the IEEE Transactions on Systems, Man and Cybernetics, and *Pattern Recognition*.

ANIS YAZIDI received the M.Sc. and Ph.D. degrees from the University of Agder, Grimstad, Norway, in 2008 and 2012, respectively. He is currently an Associate Professor with the Department of Computer Science, Oslo and Akershus University College of Applied Sciences. He was a Researcher with Teknova AS, Norway. His research interests include machine learning, learning automata, stochastic optimization, recommendation systems, pervasive computing, and the applications of these areas in industrial applications.