# Scheduling Domestic Shiftable Loads in Smart Grids: A Learning Automata-Based Scheme

Rajan Thapa[1], Lei Jiao[1](✉), B. John Oommen[2], and Anis Yazidi[3]

[1] Department of ICT, University of Agder, Grimstad, Norway
lei.jiao@uia.no

[2] School of Computer Science, Carleton University, Ottawa, Canada

[3] Department of Computer Science,
Oslo and Akershus University College of Applied Sciences, Oslo, Norway

**Abstract.** In this paper, we consider the problem of scheduling shiftable loads, over multiple users, in smart grids. We approach the problem, which is becoming increasingly pertinent in our present energy-thirsty society, using a novel *distributed* game-theoretic framework. 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 is the pioneering solution that incorporates 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 (BLA). Numerical results (The algorithmic details and the experimental results presented here are limited in the interest of space. More detailed explanations of these are found in [13]), obtained from testing the schemes on numerous simulated datasets, demonstrate the speed and the accuracy of proposed algorithms in terms of their convergence to the game's NE point.

**Keywords:** Smart Grid · Load scheduling · Potential Game · Nash Equilibrium · Learning Automata

## 1 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. Power utility companies often

*Chancellor's Professor*; *Fellow: IEEE* and *Fellow: IAPR*. This author is also an *Adjunct Professor* with the University of Agder, Norway. The work of this author was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada.

E.T. Lau et al. (Eds.): SmartGIFT 2017, LNICST 203, pp. 58–68, 2017.
DOI: 10.1007/978-3-319-61813-5_6

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 (which are every-day, commonplace appliances in countries with colder climates), can tolerate a certain amount of delay, permitting the users the possibility to schedule them *when* are turned on. Since these shiftable loads can be adaptively scheduled, the system is capable of smoothing the power consumption curve.

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 [3,4,11]. 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.

In our study, various customers are allowed to decide by themselves whether they want to turn a load on or not. To achieve this goal, we propose a distributed Learning Automata (LA)-based approach, where each customer is equipped with a LA to learn from the environment and decide whether to turn on its appliances or not.

The application of LA in SGs has been studied a little, including using them in the underlying communication network in SGs and in the power scheduling approaches. 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, explained in the body of the paper. In the case of all these LA-based schemes, the consumers do not need to share information to the provider. Rather, they 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*.

## 2 System Model

### 2.1 Problem Formulation

The research undertaken in this research focuses on the domestic smart-grid subnet. The lowest level, i.e., the local domestic network between the transformers and the end customers, is the subnet that we concentrate on. A typical scenario of this subnet is an apartment building with many 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 various families, and at the same time to maintain the overall power consumption below a given power budget. Following the common practice [6,14], 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 $N$ customers with their individual 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 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:

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

To simplify the notation, unless explicitly stated, we denote the term $\sum_{i=1}^{N}(d_i L_i)$ 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, i.e., $\sum_{i=1}^{N} L_i \leq C_{SL}$. 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, i.e., $\sum_{i=1}^{N} L_i > C_{SL}$. 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[1] 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 deduced to a subset-sum problem. 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 interactions between the customers is modeled as a game, detailed presently.

## 3 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. (2):

$$U_i(d_i, \mathbf{d}_{-i}) = \begin{cases} \frac{1}{L_i d_i + \sum_{j \in I \setminus i} L_j d_j + C_{SL}}, & C_{SL} < L_i d_i + \sum_{j \in I \setminus i} L_j d_j, \\ \frac{1}{C_{SL} - L_i d_i - \sum_{j \in I \setminus i} L_j d_j}, & C_{SL} \geq L_i d_i + \sum_{j \in I \setminus i} L_j d_j, \end{cases} \quad (2)$$

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 $C_{SL}$ 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 [8] and the reasons are as follows: According to the definition of a Potential Game, the payoff of any player

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

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 [8]. From the properties of Potential Games, we see that the game has at least one pure strategy Nash Equilibrium (NE) point. The fact that a global optimal point of the problem is a NE point of the game, and *vice versa*, is proven in [13]. Also, 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.

## 4 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.

### 4.1 Decisions of Users and Their Effects on Total Load

We invoke a typical working scenario for a LA [9]. It consists of a sequence of interaction cycles between the LA and its environment. In each iteration, the LA selects an action ($\alpha_i$), which is either rewarded ($R = 1$) or penalized ($R = 0$) by the environment as a response. 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.

### 4.2 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. Otherwise, a Penalty is applied (i.e., when $L_T(s)$ is either greater than $C_{SL}$, or less than the previous value $L_T(s-1)$). 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, $s$
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
                other users
 4:         if $L_T(s) \leq C_{SL}$ and $L_T(s) \geq L_T(s-1)$ then
 5:             Decision $d_i(s)$ leads to a Reward to user $i$
 6:         else
 7:             Decision $d_i(s)$ leads to a Penalty to user $i$
 8:         end if
 9:     end for
10: 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.

### 4.3 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 Linear Reward-Inaction (LRI) scheme [9], the Coordination-game Learning Automata (CLA) [7], and the more-recently introduced Bayesian Learning Automata (BLA) [5]. The details of the relevant steps for each of these schemes are included in the formal algorithms described in [13], and omitted here in the interest of space.

**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, $\lambda$, 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 actual decision for scheduling communicated to the SG's power provider will occur after a final decision is attained by the LA-based scheme.

**Decision Making for the CLA.** The CLA is similar to the LRI since it involves the action probabilities and a learning parameter (denoted by $\lambda$), 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 (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 [7]. In our particular problem, we need a normalization of the values of the utility to ensure that the feedback is in the interval $[0, 1]$ [7].

Using such a mapping and updating rule, it is possible to prove that the CLA will converge to the pure equilibrium 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.* [12] since the NE of our Potential Game corresponds to the mode of the payoff matrix. Again, the actual decision communicated to the SG's power provider will occur after a final decision is attained by the CLA-based scheme.

**BLA Based Decision Making Process.** In the BLA-based 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 value of $b_{i,j}$ is increased if the decision leads to a penalty, as shown in Table 1.

**Table 1.** 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 | $b_{i,1} = b_{i,1} + 1$ | $b_{i,0} = b_{i,0} + 1$ |

## 5 Simulation and Experimental Results

To evaluate the performance of the LA-based schemes, we carried out simulations 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[2]. The experiments were conducted to capture two important metrics, i.e., the accuracy of the convergence and the speed of the convergence.

### 5.1 The Data Sets

The simulation configuration was derived based on the real-life measurements of the electricity consumption for 28 domestic users [2,10,15], as shown in Table 2.

**Table 2.** This table lists the demands of 28 user (in KWh) used in our experiments.

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

A word about how the data in Table 2 was obtained is not out of place. Specifically, the annual power consumption in [15] was converted to yield the average consumption for every 15 min timespan [2] considering the scheduling interval in real life. Half of these customers' demands were considered as shiftable loads [10]. 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 [1], 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.

### 5.2 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 3 and 4. Table 3 illustrates the simulation results with all 28 users while Table 4 summarizes the results when the last 15 users in Table 2 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 and the LRI,

[2] Additional results can be seen from the technical report of the First Author, and can be made available on request.

the results are illustrated for different values of $\lambda$. 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.

**Table 3.** Simulation results from the *BLA* 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 |
| $\lambda = 0.03$ | 1352.000 | 1352.000 | 136737 | 141411 |
| $\lambda = 0.05$ | 1352.000 | 1352.000 | 60045 | 61853 |
| $\lambda = 0.08$ | 1352.000 | 1352.000 | 25690 | 26986 |
| $\lambda = 0.10$ | 1352.000 | 1352.000 | 16906 | 17863 |
| $\lambda = 0.12$ | 1352.000 | 1352.000 | 11458 | 12231 |
| $\lambda = 0.14$ | 1352.000 | 1352.000 | 7797 | 8460 |
| $\lambda = 0.16$ | 1351.998 | 1352.000 | 5391 | 6409 |
| $\lambda = 0.17$ | 1351.998 | 1351.998 | 4772 | 5333 |
| $\lambda = 0.2$ | 1351.946 | 1351.988 | 2933 | 3417 |
| $\lambda = 0.3$ | 1349.608 | 1351.408 | 581 | 911 |
| $\lambda = 0.4$ | 1335.400 | 1348.530 | 181 | 300 |
| $\lambda = 0.5$ | 1289.444 | 1342.843 | 92 | 135 |

The performances of the LRI and the CLA depend fundamentally on the value of the LA's parameter, $\lambda$. With a sufficiently small value for $\lambda$, 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 $\lambda$ is relative large, but the convergence accuracy is compromised. If we compare the $\lambda$-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 $\lambda$-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 $\lambda$ led to a superior performance for the LRI and the

**Table 4.** Simulation results from the *BLA* and *LRI*-based algorithms with $i = 15$ and $C_{SL} = 300\,\text{KWh}$.

| Method | Sum of selected loads | | Iterations | |
|---|---|---|---|---|
| BLA | 299.872 | | 1101 | |
| | LRI | CLA | LRI | CLA |
| $\lambda = 0.06$ | 300.000 | 300.000 | 5764 | 6082 |
| $\lambda = 0.08$ | 300.000 | 300.000 | 3763 | 3767 |
| $\lambda = 0.09$ | 300.000 | 300.000 | 3056 | 3148 |
| $\lambda = 0.10$ | 299.997 | 299.997 | 2630 | 2537 |
| $\lambda = 0.12$ | 299.985 | 299.995 | 1772 | 1930 |
| $\lambda = 0.14$ | 299.975 | 299.975 | 1357 | 1496 |
| $\lambda = 0.16$ | 299.917 | 299.947 | 1010 | 1142 |
| $\lambda = 0.2$ | 299.785 | 299.722 | 613 | 612 |
| $\lambda = 0.3$ | 298.215 | 298.347 | 223 | 222 |
| $\lambda = 0.4$ | 294.497 | 294.882 | 114 | 113 |
| $\lambda = 0.5$ | 285.892 | 289.745 | 76 | 78 |

CLA, than for the BLA, the $\lambda$ values could be quite different depending on the system's configurations. Consequently, the issue of determining the ideal value of $\lambda$ 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.

## 6 Conclusions

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. 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 convergence of all the schemes is comparable.

## References

1. Batterberry, T., Miller, M., Jaskolka, K., Toll, R.: Smart grid price response service for dynamically balancing energy supply and demand, June 12 2009. US Patent Ap. 12/483,975

2. Chao, H.-L., Tsai, C.-C., Hsiung, P.-A., Chou, I., et al.: Smart grid as a service: a discussion on design issues. Sci. World J. **2014**, 11 (2014)
3. Chavali, P., Yang, P., Nehorai, A.: A distributed algorithm of appliance scheduling for home energy management system. IEEE Trans. Smart Grid **5**(1), 282–290 (2014)
4. Fan, Z.: A distributed demand response algorithm and its application to phev charging in smart grids. IEEE Trans. Smart Grid **3**(3), 1280–1290 (2012)
5. Granmo, O.-C., Glimsdal, S.: Accelerated Bayesian learning for decentralized two-armed bandit based decision making with applications to the goore game. Appl. Intell. **38**(4), 479–488 (2013)
6. Liu, Y., Hassan, N.U., Huang, S., Yuen, C.: Electricity cost minimization for a residential smart grid with distributed generation and bidirectional power transactions. In: 2013 IEEE PES Innovative Smart Grid Technologies (ISGT), pp. 1–6. IEEE (2013)
7. Mason, L.: An optimal learning algorithm for s-model environments. IEEE Trans. Autom. Control **18**(5), 493–496 (1973)
8. Monderer, D., Shapley, L.S.: Potential games. Games Econ. Behav. **14**(1), 124–143 (1996)
9. Narendra, K.S., Thathachar, M.A.: Learning automata: an introduction. Courier Corporation (2012)
10. Rajakaruna, S., Shahnia, F., Ghosh, A.: Plug in Electric Vehicles in Smart Grids. Springer, Germany (2015)
11. Saad, W., Han, Z., Poor, H.V., Basar, T.: Game-theoretic methods for the smart grid: an overview of microgrid systems, demand-side management, and smart grid communications. IEEE Sig. Process. Mag. **29**(5), 86–105 (2012)
12. Sastry, P., Phansalkar, V., Thathachar, M.: Decentralized learning of Nash equilibria in multi-person stochastic games with incomplete information. IEEE Trans. Syst. Man Cybern. **24**(5), 769–777 (1994)
13. Thapa, R., Jiao, L., Oommen, B.J., Yazidi, A.: A learning automaton-based scheme for scheduling domestic shiftable loads in smart grids. Unabridged Version of this Paper (2016)
14. Zhu, Z., Tang, J., Lambotharan, S., Chin, W.H., Fan, Z.: 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 (2012)
15. Zimmermann, J.-P., Evans, M., Griggs, J., King, N., Harding, L., Roberts, P., Evans, C.: Household electricity survey: A study of domestic electrical product usage. Intertek Testing & Certification Ltd. (2012)