Reducing Electricity Costs in a Dynamic Pricing Environment

Edgar Galvan, Colin Harris, Ivana Dusparic, Siobhán Clarke and Vinny Cahill
Distributed Systems Group, School of Computer Science and Statistics, Trinity College Dublin
Email: { edgar.galvan, colin.harris, ivana.dusparic, siobhan.clarke, vinny.cahill } @ scss.tcd.ie

Abstract—**Smart Grid technologies are becoming increasingly dynamic, so the use of computational intelligence is becoming more and more common to support the grid to *automatically* and *intelligently* respond to certain requests (e.g., reducing electricity costs giving a pricing history). In this work, we propose the use of a particular computational intelligence approach, denominated Distributed W-Learning, that aims to reduce electricity costs in a dynamic environment (e.g., changing prices over a period of time) by turning electric devices on (i.e., clothes dryer, electric vehicle) at residential level, at times when the electricity price is the lowest, while also, balancing the use of energy by avoiding turning on the devices at the same time. We make this problem as realistic as possible, by considering the use of real-world constraints (e.g., time to complete a task, boundary times within which a device can be used). Our results clearly indicate that the use of computational intelligence can be beneficial in this type of dynamic and complex problems.**

I. Introduction

A Smart Grid (SG) is defined as a type of electrical power grid whose goal is to respond to the behaviour and actions of energy suppliers and consumers to efficiently deliver economic, reliable and sustainable electricity services.

Computational Intelligence (CI) has successfully been used in tackling different and challenging problems in SGs (e.g., disturbance diagnosis [8], power secondary voltage control [16], load restoration [19]). One element that underlies all these works is the use of Multi-Agent Systems [14] (MAS). The main idea in MAS is to break down a complex and dynamic problem handled by a centralised system into a smaller and more manageable problems controlled by several independent entities (distributed system). We further discuss the use of CI, in particular the of MAS in SGS in Section II

There are still many challenges and opportunities for CI to support the SG to *predict* and *intelligently* respond to certain requests. In this work, we are interested precisely to do address this by means of MAS.

More specifically, this work intends to use MAS in SGs to learn the optimal times to switch two electric devices on or off (i.e., clothes dryer and electric vehicle) in a *dynamic environment* (e.g., price rates changing over time) with the ultimate goal of reducing costs. Moreover, to make this problem realistic, we take into considerations real-world constraints (e.g., boundaries times that specify when a device can be used) simulating typical scenarios that might arise at a residential level. Finally, we also aim to balance the use of energy by avoiding turning the devices on at the same. We further describe this in detail in Section V.

This paper is organised as follows. In the next section, we set the foundations of our work by presenting MAS along with previous works that have used it in Smart Grids. In Section III, we present in detail our approach, named Distributed W-Learning (DWL). In Section IV, we give details on how we modeled the explained problem using our DWL approach. Section V presents the experimental setup used to conduct our experiments. In Section VI we present and discuss our findings. Finally, in Section VII we draw some conclusions and discuss some potential future work.

II. Related Work

As indicated previously, the use of Computation Intelligence (CI) in SGs has increased significantly over the last few years. Indeed a recent issue of CI in the IEEE Transactions in Smart Grids was recently devoted to this topic. In the following paragraphs, we summarise some CI works in SGs that have inspired, to some extent, the work presented in this paper.

Molderink et al. [9] proposed a three-step approach to manage the cooperation of distributed generation, distributed storage and demand-side load managements applied to domestic energy streams. The authors pointed that these three elements are relevant to each other. To this end, they proposed a control strategy consisting of three steps. In the first step, a system located at the consumers level predicts the production and consumption pattern for all appliances for the upcoming day. In the second step, these optimisation processes can be used by a central planner to exploit the potential to reach a global objective. It is here where the hierarchy is designed. That is, the global controller consists of multiple nodes connected in a tree structure (i.e., each house sends its profile to its parent node). The result of this step is planning for each household for the upcoming day. Finally, in the last step, a real time algorithm determines at which times appliances are switched on/off, when and how much energy flows from or to the buffers and when and which generators are switched on.

The notion of distributed agents in SGs has also been explored recently by Ramchurn et al. [12] inspired, according to the authors, by the limitations present in Smart Meters (these aims to control the devices in the home to minimise inefficiencies in usage and maximise savings to the user) and Demand Side Management (DSM) technologies (which have been developed to alter the behaviour of users). One major limitation in both technologies, according to the authors, is that it is unclear how these can be rolled out to millions of houses or buildings nationwide. To address this problem, the authors proposed a novel approach denominated Decentralised DSM, where the main idea is to allow multiple homogeneous agents (smart meters) to coordinate in a decentralised way. The authors reported a reduction of demand peak of up to 17% and 6% reduction of carbon emissions.

MAS have also been used for load restoration. In [19], Xu and Liu presented a distributed multi-agent based load restoration algorithm. Similarly to [9], the authors considered a local and global approach. Global information that is needed for distributed load restoration can be achieved based on the Average Consensus Theorem [18] (this relied on local information to guarantee that the important information can be shared in a distributed way). Local communication is achieved by the interaction of agents that are neighbours. The design and implementation of a MAS that provides intelligence to a distributed smart grids has also been explored in [10]. Other interesting works using MAS in SGs include the use to disturbance diagnosis [8], power secondary voltage control [16]. A more comprehensive summary of works using MAS in SGs can be found in [10].

As can be seen from the previous paragraphs, it is clear that CI has been used in different problems in SGs. Particularly, MASs have attracted the attention of many researchers in SGs due to its success in challenging problems. In the next section we present our approach based on heterogeneous MASs.

III. Distributed W-Learning

The main goal of this work is to use decentralised and collaborative techniques to learn optimal times in a *dynamic environment* (e.g., changing prices over time) to turn electric devices within a household on and off, so as to satisfy device constraints, user preferences, minimise overall electricity cost of the household and balance energy requirements.

To simulate the problem, we model the electricity needs of two household devices: a clothes dryer and an electric vehicle. Each of these devices has a set of associated constraints: (a) each device can be used or charged within some time boundaries (i.e., earliest start time and latest finish time), (b) each device has to finish its task given within certain time (e.g., a clothes dryer is given two hours to finish drying the clothes), and (c) the balance of energy is managed by avoiding turning devices on at the same time, whenever possible.

A. Distributed W-Learning

Distributed W-Learning (DWL) [1] is a learning-based algorithm for agent-based self-optimisation that enables collaboration between heterogeneous agents in order to simultaneously satisfy multiple heterogeneous system policies. DWL learns and exploits the dependencies between agents and between policies to improve performance while respecting the relative priorities of the policies.

DWL is based on Reinforcement Learning (RL) [13], which is considered particularly suitable for implementation of self-organising optimisation behaviours in large-scale systems, as it does not require a predefined model of the environment, which, due to the scale and complexity of such systems, is time-consuming and complex to construct [15]. DWL works by using Q-Learning and W-Learning, as follows.

B. Q-Learning and W-Learning Methods

In DWL, each agent uses a single Q-learning [17] process to implement each of its own local goals (policies) that it is tasked with meeting. This Q-learning process is a RL technique that works by learning an action-value function that gives the expected utility of taking a given action in a given state and following a policy thereafter.

The problem model consists of an agent, states and a set of actions per state. So, by performing an action, the agent (in our case, agents controlling an electric vehicle and a clothes dryer) can move from state to state, where each state provides the agent a reward (a real or natural number). The goal of this process is for the agent to maximise its total reward. Q-Learning is highly influenced by the values assigned to the discount factor γ and to the learning rate α.

In any RL method, an agent wanders in an unknown environment and tries to maximise its long term return by performing actions and receiving rewards. These variables (learning rate and discount factor) balance this process. The discount factor determines the present value of future rewards. That is, the lower the value of the discount factor is (e.g., close to 0), the more the emphasis is put by the agent to maximise immediate rewards compared to long term returns. In contrast, when this value is close to 1, the agent takes future rewards into account more strongly. The latter variable (learning rate) controls how fast we modify our estimates.

To arbitrate between different policies, an agent uses W-learning [7] which learns the relative importance of an agent's policies. In W-learning, for each of the states that each policy can be in, the agent learns how the performance of that policy is affected should its preferred action not get executed. This difference between the reward the agent would receive if its preferred action is executed and the reward the agent receives when another policy's action is executed, is learned as a W-value. The agent then executes the action suggested by the policy that would be the most negatively affected if its suggested action is not respected.

In this way, dependencies between local policies are learned: if policies are compatible, actions suggested by policies will be the same/similar, and execution of one will not negatively affect the other; when policies conflict, an action preferred by one policy will have a larger negative impact on the performance of the other.

C. The Use of Q-Learning and W-Learning in DWL

In DWL, as well as Q-values and W-values for all of their local policies, all agents also learn Q-values and W-values for all of the policies that their immediate neighbours implement (so-called remote policies), i.e., they learn how their local actions affect their neighbours' performance. In such a way dependencies between local and remote policies are learned, similar to learning of the dependencies between local policies.

At each time step, each agent considers the W-values for the current state of each of its local and remote policies. If any of the immediate neighbours' policies has a higher W-value than the agent's local W-values, the action suggested by that neighbour can be executed.

Simultaneous optimisation towards multiple heterogeneous policies on multiple heterogeneous agents can be enabled while priorities of the policies are respected both locally and within the overall system through use of a flexible cooperation mechanism. All these features, as explained previously, are present in our DWL approach.

IV. Modelling the Problem in DWL

As described in Section III, DWL works by using both Q-Learning and W-Learning. Thus, it is necessary to define actions, states and rewards for each of our devices: the clothes dryer and the electric vehicle.

The simplest element is the action that each agent (device) can take: either the device can be turned on or turned off. The states and rewards are more complex and each is described in the following paragraphs.

A. Modelling the States

The definition of the state is one of the main elements in any multi-agent system as it represents the "environment" where the agents make decisions (i.e., actions). To make this problem more interesting, we have defined an scenario where there is an overlap on the hours where the devices can start working, as we are also interested in seeing if our approach is able to avoid turning the devices on at the same time, as a way to balance energy usage. We describe this formally in Section V.

The relevant state for the clothes dryer is the time t where an agent makes a decision, three prices p_1, p_2, p_3, the action a (i.e, turn on/off) and the time left tl to finish the task.

The state is similar for the electric vehicle with one difference. Here, we are dealing with two forms of time left: time left until the vehicle is fully charged and the time left for the vehicle to be partially charged. So, we need to add an extra time left to the design of the state space.

An additional element indicates the status of both devices (on/off) when there is an overlap of hours, which is used to balance the use of energy. We will discuss more of this in the following section.

B. Defining Rewards

The reward mechanism varies according to the device. For example, when using the clothes dryer, we simply reward the agent 1 if it turns on the device at the time where the price is the lowest and reward it 0, otherwise.

For the electric vehicle the situation is slightly more complex. The agent is rewarded 1 if it decides to start charging the vehicle at the time when the price is the lowest. Only if this happens, the agent is rewarded another 0.5 units if the minimum time of charge was achieved (partially charged) and it is rewarded another 0.5 units if the agent manages to charge the optimal time of the vehicle (fully charged). So, in principle, an agent that is able to start charging the electric vehicle at the time with the lowest price and charge it fully for the optimal time, gets a reward of 2 units.

As indicated before, we are also interested in seeing if one can control the agents by avoiding turning the devices on at the same time. For this, we assigned an extra unit to the agent if it manages to turn any of the devices on when the other is turned off. By doing so, we are trying to balance the use of energy. We further discuss this in Section VI.

V. Experimental Setup

A. Dynamic Environment

To test our DWL approach in SGs to learn the optimal times to switch two electric devices on or off in a dynamic environment, we used two different pricing history files, where each of them follow a given pattern (i.e., [low, medium, high]; [medium, low, high]).

More specifically, the dynamic environment is simulated by using two pricing history data, let us call them data A and data B, where each contains the hour with the associated electricity price. We guarantee a challenging dynamic environment by having a different price at each time in the two pricing history data used in this work. For example, when using history data A, we could have something like Time 17:00, Price = 100, and when using history data B for the same Time (17:00), we could have Price = 200. Then we switch n times, from data A to data B and *vice versa* every 30000 time steps[1] (see Table I for more details).

B. Constraints

The use of each of these devices is subject to a start time that lies within the boundaries of an earliest start time (EST) and a latest finish time (LFT). Moreover, an action (i.e., turn on/off) can take place every 30 minutes (i.e., we simulate that the price changes every 30 mins). It is also worth mentioning that we also took into consideration the time to complete a task, either for the clothes dryer to finish drying the clothes (T_{CD}) or for the electric vehicle to be partially ($TMin_{EV}$) or fully ($TIdeal_{EV}$) charged.

More formally, we have that the constraint for the clothes dryer can be expressed in the following terms,

$$EST_{CD} \leq ST_{CD} \leq LFT_{CD}$$

where $EST_{CD} = 17:30$, $LFT_{CD} = 23:00$, $ST_{CD} = EST_{CD}$, and the time allowed to finish drying the clothes $T_{CD} = \{2\}$.

Similarly, we have that the constraint for the electric vehicle can be expressed as follows:

[1] We chose 30000 time steps to allow our multi-agent system to learn the pattern and give it opportunity to predict prices in this dynamic environment.

TABLE I
SUMMARY OF PARAMETERS USED IN OUR EXPERIMENTS.

Parameter	*Value*
Learning Rate (α)	0.4
Discount Factor (γ)	0.4
n Steps to Switch from Price to Price	$2, 3, 4, 5, 6, 7, 8$
Time Steps	$30000 * n$
Selection	Boltzmann (temperature = 2)

$$EST_{EV} \leq ST_{EV} \leq LFT_{EV}$$

where $EST_{EV} = 17 : 30$, $LFT_{EV} = 3 : 30$, $ST_{EV} = EST_{EV}$, and the times allowed to fully charge the electric vehicle and partially charge it at $TIdeal_{EV} = \{4\}$, $TMin_{EV} = \{2\}$.

C. Parameters

To study our approach in a dynamic environment, along with the constraints associated to the problem, as described before, we run 10 independent runs for each of the values associated to the n steps to switch from price to price. The rest of the parameters we have used are summarised in Table I. In the following section we present and describe the results obtained by our approach (DWL) using the described scenario.

VI. RESULTS AND DISCUSSION

We simulate price rates changing over time to simulate a dynamic environment, similarly as the one can that could be found in a SG scenario, as explained in Section V. To measure how well or bad our approach behaves in this dynamic-constrained problem, we performed an extensive empirical experimentation (10 * 7 runs in total[2], each run with 30000 time steps * n steps to switch from price to price).

A. Overview Performance of DWL

We are interested in seeing if it is possible to reduce electricity costs by predicting prices and switching on electric devices at the lowest price possible.

To measure this, we simulated a similar scenario that a user faces, that is, turning devices on or off without the user knowing the price at a given time (let us call this approach "random approach"). Thus, we performed a random action-evaluation process. That is, we selected an action (i.e., either turn on/off the electric device) at any given time within the boundaries set by EST and LST, and kept record of the prices associated to those particular times when the electric devices was turned on. Then, we averaged this and used it for comparison purposed.

First, let us focus our attention to the clothes dryer (left-hand side of Figure 1) to see if there was an increase or decrease on electricity costs. It is clear that the agent fails to switch on the clothes dryer at the lowest possible prices, regardless of the number of changes during a period of time (the higher the number of changes, the more challenging the problem is). Interestingly, it seems like the performance of our multi-agent system approach remains more or less the same, regardless the number of changes that takes place during a period of time. To understand why this agent performs poor in this dynamic-constraint problem, one really needs to consider the constraints imposed to the problem, as introduced in the previous section. We will discuss more about it later in this section.

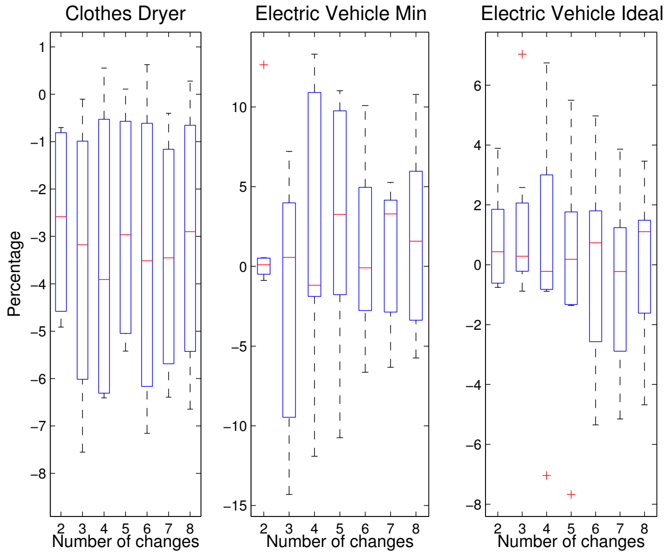

Fig. 1. Increase or decrease achieved by our DWL approach compared to a random approach, expressed in percentages, for each of the electric device present in a household (i.e., clothes dryer and electric vehicle) taking into consideration, n, which is the number of changes ($n = \{2, \cdots, 8\}$ over a period of time.

If we, now, focus our attention in the other agent (i.e., electric vehicle – middle and right-hand side of Figure 1), we can see a better (positive) trend compared to the clothes dryer, especially when the vehicle is partially charged (i.e., middle of Figure 1), and, to a lesser degree when the electric vehicle is fully charged (right-hand side of Figure 1). When the electric vehicle is partially charged, the electricity cost reduction is fairly consistent regardless of the number of changes in prices (expect when the number of changes is three). For example, if one considers the mean values, the best result is around 4% electricity cost saving (this occurred when there were five changes). The situation is more or less similar when the electric vehicle is fully charged, but only when the number of changes is less than four. From this point onwards, the agent fails to predict the lowest possible price resulting in an increase to the user's electricity costs.

So far, we have seen, how the proposed multi-agent system behaves more or less well in this complex and dynamic problem, but we do not how this was achieved. To explain this, one really needs to break down this result and analyse it for time periods. This is discussed in the following section.

[2] 10 independent runs and 7 different changes over a period of time, as indicated by n in Table I.

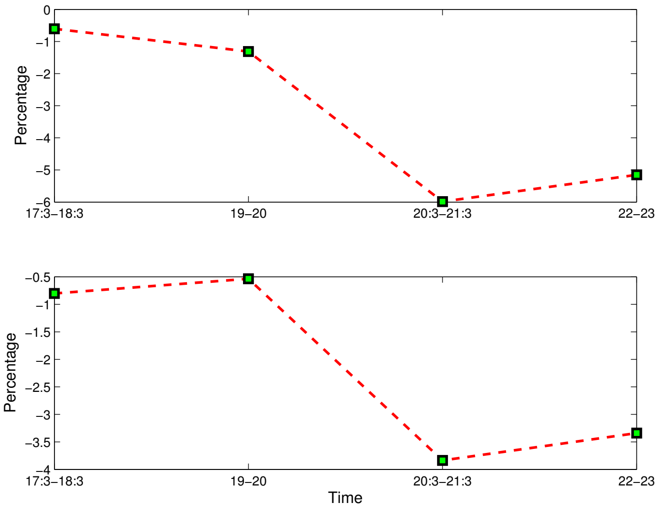

Fig. 2. Percentage increase/decrease achieved by our proposed multi-agents system (DWL) when controlling the **clothes dryer** and in the presence of two pricing history changes.

B. Detail Analysis of Performance

In the previous paragraphs, we have analysed the overall performance achieved by our approach in a dynamic environment (e.g., changing prices). It is fair to say, however, that it is unclear how this was achieved. Thus, to have a better understanding of this, it is necessary to perform a deeper analysis of what happen at each hour for each electric device along with the constraints and features assigned to each of them (e.g., time needed to complete a task).

Due to space restrictions, we focus our attention in the presence of two pricing history changes (not to be confused with the number of pricing history data used in this work, see Section V for details). However, it is worth mentioning that the same trend is observed when using more changes, as can be inferred by the overall performance shown in Figure 1.

Let us start our analysis with the clothes dryer. As discussed in the previous paragraphs, our model fails to turn on this device at the lowest possible price. To explain why this happens, one really needs to considers all the constraints and preferences, as defined in Section V: (a) the time that the clothes dryer has to finish its task (two hours), (b) there is an overlap in the time where both devices can operate, and finally (c) the system tries both: to turn on the device at the lowest price, and it also tries to balance energy by avoiding that both devices are switched on at the same time.

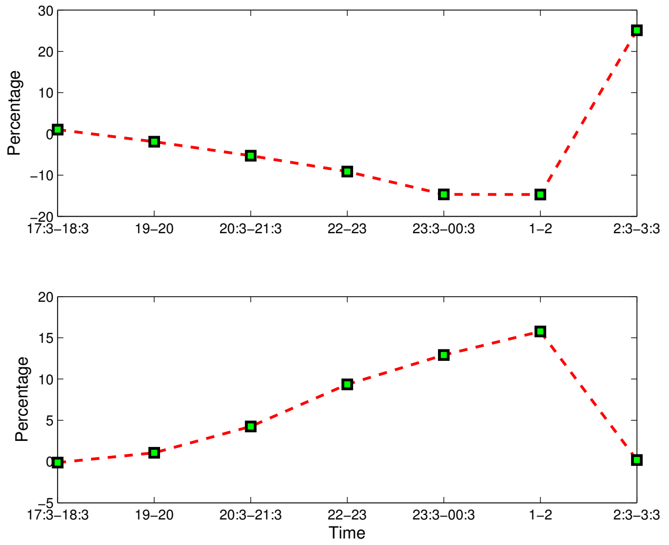

Fig. 3. Percentage increase/decrease achieved by our proposed multi-agents system (DWL) when controlling the **electric vehicle** (minimum charge) and in the presence of two pricing history changes.

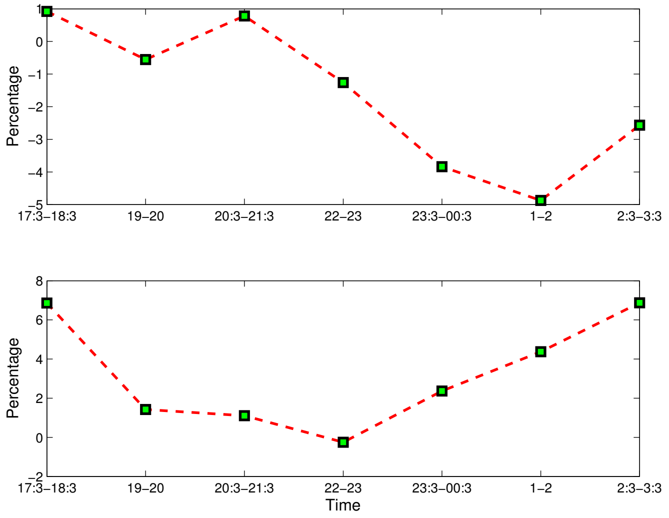

Fig. 4. Percentage increase/decrease achieved by our proposed multi-agents system (DWL) when controlling the **electric vehicle** (ideal charge) and in the presence of two pricing history changes.

Taking all these elements into consideration, we can now take a look to what happen when the system controls the clothes dryer (see Figure 2). We know that each pricing history data follows a given pattern (e.g., [medium, low, high] price), so for clarity purposes we grouped these patterns in period ranges (e.g., 17:30 – 18:30). By doing so, it is clear that there are four periods and in each of these periods, there is only one time with the lowest price. Now, given that the clothes dryer needs two hours to finish its task, that will mean that the system will need to turn on the clothes dryer at the lowest price in these four periods. Because of the constraints specified in Section V and discussed in the previous paragraphs, it is clear that this highly unlikely to happen due to the presence of the electric vehicle that can operate at the same time, along with the type of preference given to the vehicle (e.g., the reward for the vehicle is twice higher compared to the clothes dryer). This is confirmed in Figure 2 (again, in this example we used two changes). From this, it is clear that regardless of the history pricing data used (i.e., A or B, top and bottom of Figure 2, respectively), the system turned on the clothes dryer at times where the price was not the cheapest.

Let us continue our analysis by turning our attention to the other device: the electric vehicle in both situations: when it is partially charged (Figure 3) and when it is fully charged (Figure 4).

As indicated in Section V, we know that the electric vehicle needs two hours to be partially charged (Figure 3) which is the same time that the clothes dryer needs to finish its task. In this case, however, our proposed system started to learn to charge the electric vehicle at times when the price was the cheapest (from 2:30 to 3:30, as seen at the top of Figure 3), and this trend continued when there was a change in the data price used (bottom of Figure 3). It is worth pointing out that this saving in electricity costs is higher when there is no overlap of hours compared to the clothes dryer (i.e., from 23:30 to 3:30). This indicates that while the system is able to save some electricity costs, it also learns to avoid turning on both devices (clothes dryer and electric vehicle) at the same time, allowing to balance the use of energy.

Finally, let us consider when our system tries to fully charge the electric vehicle (twice the time needed to charge the vehicle compared to the partially charge scenario – Figure 4). The situation is more or less similar compared when the system tries to partially charge it, as explained in the previous paragraph. That is, there is a tendency to start charging it at times when the electricity price is the cheapest (bottom of Figure 4), even in period of hours where there is an overlap with the clothes dryer (i.e., from 17:30 to 23:00, and from 17:30 to 3:30, for the clothes dryer and electric vehicle, respectively).

VII. Conclusions and Future Work

In this paper, we propose the use of Distributed W-Learning, which is a computational intelligence method, in Smart Grid technologies with the ultimate goal of learning the optimal times to switch electric devices on or off to minimise electricity costs, by *learning* and *predicting* the electricity price, based on a pricing history, in a dynamic price environment (e.g., prices rates changing over a period of time), while also, trying to balance the use of energy by avoiding turning on electric devices at the same time.

To make this problem as realistic as possible, we considered the use of constraints (e.g., time needed for the devices to finish their tasks, boundary and overlap times within which a device can be used).

We show how our proposed approach is able to reduce electricity costs in some scenarios (e.g., electric device partially charged, and in a lesser degree, when the system aims to fully charge the electric vehicle). More importantly, we showed how computational intelligence can successfully been used in dynamic environments, which is precisely the type of environment present in many real-world Smart Grid problems.

We are planning to extend this work considerably. For example, we are considering the use of more devices (agents), we are also planning in using other forms on CI. In particular we are interested in using Evolutionary Algorithms and their novel research on problem hardness (e.g., locality [3], [4], [5], neutrality [2], [6], [11]) to, for example, speed the learning process up.

Acknowledgments

This research was supported by Science Foundation Ireland (SFI) under the Principal Investigator research program 10/IN.1/I2980 "Self-organizing Architectures for Autonomic Management of Smart Cities" and by SFI grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie).

References

[1] I. Dusparic and V. Cahill. Autonomic multi-policy optimization in pervasive systems: Overview and evaluation. *Transactions on Autonomous and Adaptive Systems. (In Press).*

[2] E. Galvan-Lopez. *An analysis of the effects of neutrality on problem hardness for evolutionary algorithms*. PhD thesis, University of Essex, 2009.

[3] E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon. Defining locality in genetic programming to predict performance. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[4] E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon. Towards an understanding of locality in genetic programming. In M. Pelikan and J. Branke, editors, *GECCO*, pages 901–908. ACM, 2010.

[5] E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon. Defining locality as a problem difficulty measure in genetic programming. *Genetic Programming and Evolvable Machines*, 12(4):365–401, 2011.

[6] E. Galván-López, R. Poli, A. Kattan, M. ONeill, and A. Brabazon. Neutrality in evolutionary algorithms... what do we know? *Evolving Systems*, 2:145–163, 2011. 10.1007/s12530-011-9030-5.

[7] M. Humphrys. Action selection methods using reinforcement learning. In *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 135–144. MIT Press, 1996.

[8] S. McArthur, E. Davidson, J. Hossack, and J. McDonald. Automating power system fault diagnosis through multi-agent system technology. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, page 8 pp., jan. 2004.

[9] A. Molderink, V. Bakker, M. Bosman, J. Hurink, and G. Smit. Management and control of domestic smart grid technology. *Smart Grid, IEEE Transactions on*, 1(2):109 –119, sept. 2010.

[10] M. Pipattanasomporn, H. Feroze, and S. Rahman. Multi-agent systems in a distributed smart grid: Design and implementation. In *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, pages 1 –8, march 2009.

[11] R. Poli and E. Galván-López. The effects of constant and bit-wise neutrality on problem hardness, fitness distance correlation and phenotypic mutation rates. *IEEE Trans. Evolutionary Computation*, 16(2):279–300, 2012.

[12] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. Jennings. Agent-based control for decentralised demand side management in the smart grid. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, pages 5–12, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[13] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, Mar. 1998.

[14] K. P. Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.

[15] G. Tesauro. Reinforcement learning in autonomic computing: A manifesto and case studies. *IEEE Internet Computing*, 11(1):22–30, 2007.

[16] H. Wang. Multi-agent co-ordination for the secondary voltage control in power-system contingencies. *IEE Proceedings - Generation, Transmission and Distribution*, 148(1):61–66, 2001.

[17] C. J. C. H. Watkins and P. Dayan. *Machine Learning*, (3):279–292, May.

[18] F. Xiao, L. Wang, and Y. Jia. Fast information sharing in networks of autonomous agents. In *American Control Conference, 2008*, pages 4388 –4393, june 2008.

[19] Y. Xu and W. Liu. Novel multiagent based load restoration algorithm for microgrids. *Smart Grid, IEEE Transactions on*, 2(1):152 –161, march 2011.