Prediction-Based Multi-Agent Reinforcement Learning in Inherently Non-Stationary Environments

ANDREI MARINESCU, IVANA DUSPARIC, and SIOBHÁN CLARKE, Trinity College Dublin

Multi-agent reinforcement learning (MARL) is a widely researched technique for decentralised control in complex large-scale autonomous systems. Such systems often operate in environments that are continuously evolving and where agents' actions are non-deterministic, so called inherently non-stationary environments. When there are inconsistent results for agents acting on such an environment, learning and adapting is challenging. In this article, we propose P-MARL, an approach that integrates prediction and pattern change detection abilities into MARL and thus minimises the effect of non-stationarity in the environment. The environment is modelled as a time-series, with future estimates provided using prediction techniques. Learning is based on the predicted environment behaviour, with agents employing this knowledge to improve their performance in realtime. We illustrate P-MARL's performance in a real-world smart grid scenario, where the environment is heavily influenced by non-stationary power demand patterns from residential consumers. We evaluate P-MARL in three different situations, where agents' action decisions are independent, simultaneous, and sequential. Results show that all methods outperform traditional MARL, with sequential P-MARL achieving best results.

CCS Concepts: • **Theory of computation** → **Multi-agent reinforcement learning**; • **Computing methodologies** → **Reinforcement learning**; **Multi-agent reinforcement learning**; *Multi-agent systems*; • **Hardware** → *Smart grid;*

Additional Key Words and Phrases: Multi-agent systems, reinforcement learning, environment prediction, smart grids

ACM Reference Format:
Andrei Marinescu, Ivana Dusparic, and Siobhán Clarke. 2017. Prediction-based multi-agent reinforcement learning in inherently non-stationary environments. ACM Trans. Auton. Adapt. Syst. 12, 2, Article 9 (May 2017), 23 pages.
DOI: http://dx.doi.org/10.1145/3070861

1. INTRODUCTION

Advances in multi-agent algorithms have enabled large-scale systems to perform complex tasks without requiring human assistance [Huebscher and McCann 2008]. Multi-agent systems (MAS) comprise multiple autonomous entities, known as agents, and can be used to solve problems in a distributed manner when centralised control becomes infeasible. Often, system autonomy is achieved by agents continuously learning from their interaction with the environment and each other rather than relying on predefined behaviours [Stone and Veloso 2000]. One commonly-used computational approach to learning is reinforcement learning (RL) [Sutton and Barto 1998]. Agents in Multi-agent RL (MARL) systems learn suitable actions by trial and

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie).
Authors' addresses: A. Marinescu, I. Dusparic, and S. Clarke, Distributed Systems Group, Trinity College, Dublin, Ireland; emails: marinesa@tcd.ie, ivana.dusparic@scss.tcd.ie, siobhan.clarke@scss.tcd.ie.

error, interacting with their environment to reach their goal, which is to maximise their cumulative rewards (i.e., the total reward they receive in the long run [Sutton and Barto 1998]), and can achieve this by collaborating with other agents.

However, an environment in which agents operate can be non-stationary, i.e., it might evolve so that selected actions have an effect different from the previously learned one. Once previously unencountered situations occur, agents need to re-restart the learning process (i.e., re-learn) in order to learn the new suitable actions to adapt to new situations, performing sub-optimally while doing so. Our hypothesis is that prediction of future environment behaviour can provide agents with a sufficiently good *a priori* training model for off-line learning to improve their performance in online mode [Dusparic et al. 2013]. This article proposes Predictive-MARL (P-MARL), a MARL technique augmented with environment prediction and pattern change detection capabilities to counter inherently non-stationary environments that can be described as a time series. We demonstrate the implementation of P-MARL in a smart grid case study, with non-stationary energy usage patterns. We show that P-MARL obtains improved performance over traditional MARL by using environment forecasts to prepare for upcoming changes.

The rest of this article is organised as follows: Section 2 introduces RL and reviews existing work of MARL in non-stationary environments. Section 3 presents P-MARL, our proposed approach for optimisation in non-stationary environments. Section 4 describes the application of P-MARL in a smart grid case study, while Section 5 presents and analyses the results obtained by our approach. Finally, Section 6 presents our concluding remarks and avenues for future work.

2. BACKGROUND AND RELATED WORK

2.1. Reinforcement Learning

Reinforcement learning (RL) is a computational approach to learning from interaction with the environment that enables an agent to learn how to perform well in an unknown environment through a process of trial and error [Sutton and Barto 1998]. A reward is provided to the agent for each action taken in a given state. This process is pictured as a Markov Decision Process (MDP) in Figure 1, where x is the state of the environment, s is the state of the agent, a is the action taken by the agent, and r is the reward received. The values in the brackets represent the timestep (e.g., $s(t-1)$ is the state s at timestep $t-1$). The next state depends only on the agent's current state s and action a, and not on any previous actions taken or states encountered; this is known as the Markov property. Even though some immediate actions may lead to higher rewards than others, an agent's goal is to maximise its long term reward. An agent needs to explore all states and actions combinations in order to learn the highest rewarding sequence of actions that can be taken from a particular state. This sequence is known as the agent's policy. RL agent performance is optimised once sufficient exploration through different states and actions is performed.

There are two main types of RL methods: ones that use models of the environment and rely on planning, called model-based (e.g., Dyna-Q [Sutton 1990]), and ones that learn through trial and error, called model-free (e.g., Q-learning [Watkins and Dayan 1992]).

If external changes occur in an environment, it will react differently to an agent's actions; therefore, an agent's learned information can become outdated. In such cases, we say that the environment is non-stationary. When an environment in which an agent is acting is non-stationary, an agent needs to re-learn in order to adapt to the new changes.

Model-based RL approaches use experience to model the environment's reaction to an agent's actions, and thus, agents are able to implement strategies based on the

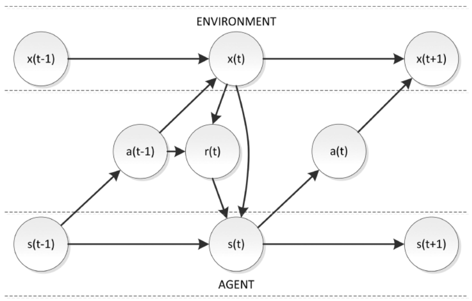

Fig. 1. RL: Agent and Stationary Environment.

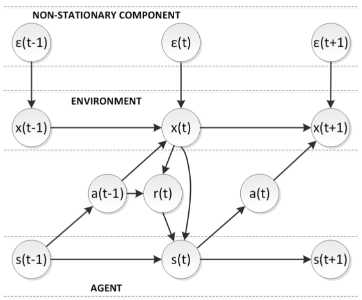

Fig. 2. RL: Agent and Non-Stationary Environment.

developed model to maximise their performance. However, when the environment is inherently non-stationary, the accumulated experience becomes misleading, even if the Markov property holds. Agents need to acquire new experience in order to learn a new model, and while doing so, the system underperforms. Model-free RL is not held back by an inaccurate model of the environment and does not need to accumulate experience for modelling it; however, it will still perform sub-optimally throughout the learning stage.

2.2. MARL in Non-Stationary Environments

When several RL agents interact within the same environment, the single agent RL problem is extended to MARL. However, when such multiple self-interested learning agents interact with each other, the result of their actions on the environment becomes non-stationary, as each agent acts in a non-stationary manner during the learning process (i.e., while it is trying different actions in the same state). In addition, non-stationarity in the environment can arise independently from agents' behaviours, i.e., through independent changes in environment variables. Therefore, non-stationarity in MARL can arise for two reasons:

A. Agent-contributed non-stationarity: several agents are simultaneously exploring within the same environment; therefore, the effect of their individual actions is non-deterministic since the environment reacts differently for each combination of actions.
B. Environment-induced non-stationarity: the environment continuously evolves by itself; therefore, the result of agents' actions are also affected by the evolution of the environment independent from agents' actions. Figure 2 illustrates how this affects the typical RL process (Section 2.1) from an MDP perspective, where environment state $x(t)$ is now also influenced by the non-stationary component $\epsilon(t)$, which represents the non-stationary independent evolution of the environment.

Techniques that address agent-contributed non-stationarity mostly focus on modelling other agents' behaviours in order to predict their actions and their joint impact on the environment [Claus and Boutilier 1998; Hu and Wellman 2003; Shoham et al. 2003; Laumônier and Chaib-draa 2005; Elidrisi et al. 2014; Hernandez et al. 2013]. However, modelling the behaviour of an opponent is not feasible in complex non-stationary environments, where many agents act upon the environment and the effect of a single agent is not transparent.

In situations where the environment itself is complex and non-stationary, agents' learning cannot converge to stationary policies; therefore, gaining knowledge about the environment can be key to successful implementation of MARL [Klügl et al. 2005].

Techniques that consider non-stationary environments do so by creating partial models of the environment [Doya et al. 2002; Choi et al. 2001]. However, these do not address continuously evolving environment dynamics, where previously defined models are not reliable. RL with Context Detection (RL-CD) [Silva et al. 2006] attempts to address environment dynamics with a context detection-based approach that builds on multiple partial models. Multiple models are continuously evaluated against a specific environment dynamic, and the model performing best is chosen to further dictate actions. If no model performs satisfactorily, RL-CD triggers learning a new model for the particular environment dynamics. The new model is triggered only when an agent consistently performs sub-optimally. In SOILSE [Salkham and Cahill 2010], fluctuations in environment behaviour are continuously monitored by a moving average filter. Once a change is detected, SOILSE's learning parameters are changed in order to allow agents to adapt to the new conditions.

In each of these solutions, the system needs to explore the suitability of actions while learning. Overall system performance is affected during this time, which is not suitable in many real-world applications such as industrial processes or vehicular traffic. Training offline can improve performance [Tesauro et al. 2006] but is directly affected by the quality of information provided. Such information can be provided by using prediction, as discussed next.

2.3. Prediction in Non-Stationary Environment

P-MARL is aimed at non-stationary environments whose behaviour can be modelled as a time series, and as such can benefit from advanced time-series analysis techniques. Such environments can be found in financial markets (e.g., predicting stock trends, detecting financial market meltdowns), meteorology (e.g., detecting seasonality in weather patterns), process monitoring (e.g., detecting abnormal operating performance), control engineering (e.g., load balancing based on model predictive control), signal processing, and electric energy consumption (e.g., estimating future energy needs of end-users). A time series is a discrete sequence of points set at fixed time intervals whose values represent successive measurements of a specific variable. This variable, denoted here as property x of an environment behaviour over an interval of length t, can be defined as $(x_1, x_2, \ldots, x_i, \ldots, x_t)$, where x_i is the measurement of the property x taken at time i. Widely-used approaches for predicting future values of time series (from the example, values $x_{t+1}, x_{t+2}, \ldots$) include statistical auto-regressive methods [Box and Jenkins 1970], Artificial Neural Networks (ANNs) [Zhang et al. 1998], and fuzzy logic [Brown and Harris 1994].

Statistical auto-regressive methods are established methods for time-series prediction in weakly stationary or non-stationary time series, with the auto-regressive moving average (ARMA) model being first introduced [Whittle 1951], and later popularised together with auto-regressive integrated moving average (ARIMA) as Box-Jenkins approaches [Box and Jenkins 1970]. The models analyse random processes and linearly relate the output of the prediction system based on previous values of the time series. The series is decomposed through a formula that relates individual coefficients with the former chosen n values. ARMA and ARIMA additionally have a moving average part where another set of coefficients is considered for the moving average model component. While ARMA can be used with weak-stationary systems, ARIMA applies differencing on a non-stationary time series, thus removing the non-stationary component and treating the result as stationary.

While directly predicting future behaviour can be very efficient, most successful techniques combine several methods to improve overall results. Among them, one of the most noticeable and effective additions is the classification of historical behaviour into different sets, generally accomplished using self-organising maps (SOM) [Kohonen

Table I. Adaptation Under Environment-Induced Non-Stationarity

	Previously encountered dynamics	New dynamics
Behaviour Change Detection	Detected through change in rewards [Choi et al. 2001; Doya et al. 2002]	RL-CD: Uses memory for known models, learns new models online [Silva et al. 2006] SOILSE: Trigger re-learning when change detected [Salkham and Cahill 2010]
Behaviour Prediction	Memory-based approaches (keep previous models of environments and switch based on change in reward) [Choi et al. 2001; Doya et al. 2002]	X

1990], k-means clustering [Hartigan and Wong 1979], or support vector machines [Cortes and Vapnik 1995]. These techniques rely heavily on past behaviour, but key environmental variables (i.e., factors that impact the behaviour of the environment, such as the passing of a cloud affecting the current temperature) can also play an important role. Monitoring the key variables can help estimate future changes in the environment.

So far, as presented in Section 2.2, most of the focus in MARL has been on predicting agent behaviour instead of environment behaviour. In this article, the main contribution stems from the use of advanced prediction techniques for determining future non-stationary environment behaviour in the learning process of MARL. Previous methods that include environment-related information in MARL only address change-detection, but not future behaviour prediction, i.e., they do not provide any information about the future evolution of the environment when it evolves towards previously unencountered situations.

2.4. Requirements for MARL in Non-Stationary Environments

A summary of the state-of-the-art in non-stationary environments is presented in Table I. Current optimisation techniques are designed to address previously encountered environment dynamics; agents can only adapt to new environment dynamics by learning online (e.g., RL-CD, SOILSE), and have no means to predict the behaviour of new environment dynamics or pretrain for it. To address these limitations, without losing any of the functionalities current techniques can provide, a multi-agent system deployed in non-stationary environments should:

—**Req. 1** support agents taking optimal actions in the current environment state (i.e., fulfil the basic MARL requirement of learning optimal actions);
—**Req. 2** detect environment behaviour changes;
—**Req. 3** predict environment behaviour: if a change in environment behaviour is detected, it should identify the type of change occurring, and re-predict the environment's future dynamics;
—**Req. 4** support agents pretraining (i.e., learning offline) for possible future environment dynamics even if these have not been encountered previously.

In the next section, we describe P-MARL's design and how it meets these requirements.

3. P-MARL

P-MARL's goal is to improve MARL performance when the environment is continuously evolving and learned information becomes outdated. Unlike solutions that react after changes in the environment are detected (e.g., RL-CD and SOILSE), P-MARL allows agents to adapt to new situations, minimising underperformance that might arise from such outdated learned information. By predicting future environment behaviour, agents can learn optimal policies offline, using a simulation of the environment. This is a key difference from RL-CD, which is model-based, i.e., it learns a new model at runtime, and therefore performs sub-optimally while doing so. In P-MARL, agents are better prepared to handle upcoming changes in the runtime environment after training in a simulated environment, before applying this synthetic knowledge at runtime.

Our solution employs multiple autonomous RL agents, and is, therefore, a decentralised approach. Centralised solutions to multi-agent optimisation in resource allocation problems are not feasible, as the time needed to solve a problem grows exponentially with the number of agents involved [Busoniu et al. 2008]. Furthermore, such solutions rely on one unit, with corresponding single point of failure issues. In P-MARL, agents are agnostic with respect to from where the prediction comes. Depending on the scale and communication capabilities of the underlying system, it is possible to perform simulations/predictions centrally, on a dedicated agent, or on any of the participating agents, or have multiple agents take responsibility for their sub-regions.

3.1. P-MARL Design

P-MARL uses a reinforcement learning algorithm known as Q-Learning [Watkins and Dayan 1992], which was chosen for its simplicity as it does not require predefined models of the environment. Q-Learning is formally defined as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big], \tag{1}$$

where $Q(s_t, a_t)$, the Q-value, is the expected discounted reward in state s_t after taking action a_t and following a certain policy afterwards [Watkins and Dayan 1992], α is the learning rate, r_{t+1} is the immediate reward received at the next timestep for taking action a_t in state s_t, γ is the discount factor, and $max_a\ Q(s_{t+1}, a)$ is the maximum expected Q-value that can be achieved from state s_{t+1} after taking an action a. The two parameters α and γ influence how much the algorithm learns from new experience and how much weight a delayed reward holds, respectively. In this article, we use the classical tabular format of Q-learning, based on Equation (1), where the Q-value of each state-action combination is stored in a table. Furthermore, we also employ softmax action selection [Sutton and Barto 1998], where the RL algorithm gradually moves from exploration to exploitation as a parameter entitled temperature is decremented. When the temperature is high, actions are almost equiprobable, but as the temperature value decreases, softmax action selection becomes similar to greedy selection.

In Q-Learning with softmax action-selection, once an agent moves towards the exploitation stage, it attempts to maximise its overall reward (i.e., Q-Value) by choosing the most long-term rewarding action from state s_t. Once sufficient exploration is performed, action a_t is considered to be the optimal action from this state. In the case of stationary environments, the transition to state s_{t+1} is a function f of the action taken at time t, assuming that we already are in a known state s_t:

$$s_{t+1} = f(s_t, a_t) \tag{2}$$

In a non-stationary environment where a change has just occurred, an agent takes action a_t in state s_t, expecting to maximise its long term reward by moving into state s_{t+1}. We consider a fully observable environment at state s_t, therefore, within the MDP framework, but which, at the same time, has a non-stationary evolution and therefore is non-deterministic. Environments that are not fully observable should be addressed under the POMDP (Partially Observable Markov Decision Process) framework

[Kaelbling et al. 1998].[1] Since the environment considered in this article is non-stationary, it influences the result of action a_t, and the new state reached is instead s'_{t+1}. Therefore, s'_{t+1} is a function of both the agent's action and the environment's independent behaviour:

$$s'_{t+1} = f(a_t, x_{t+1}) \tag{3}$$

Thus, value x_{t+1} affects the outcome s'_{t+1} together with action a_t. In stationary environments, x_{t+1} can be learned and integrated in the rewarding scheme, so Equation (3) can be reduced to Equation (2). However, when the environment changes, the previously integrated value x_{t+1} can be radically different from the actual state of the environment. In this case, action a_t can lead towards a sub-optimal state with a different reward than expected, so Q-values need to be readjusted. In such situations, further exploration is needed for the Q-Values to be re-learned until an optimal set of actions is found again. Here, environment evolution needs to be taken into account separately to improve the agent's performance. As such, in an environment defined as time series, we have:

$$x_{t+1} = \bar{x}_{t+1} + \varepsilon_{t+1}, \tag{4}$$

where $\bar{x}_{t+1}$ is the already integrated value in Q-Learning, and ε_{t+1} is the difference between the integrated value of the environment and the one actually occurring due to the environment's independent evolution. From Equation (3) and Equation (4), we can deduce that:

$$s'_{t+1} = f(a_t, \bar{x}_{t+1} + \varepsilon_{t+1}), \tag{5}$$

where ε_{t+1} impacts the time-series evolution as well. However, since ε_{t+1} is a scalar value, as we are addressing environments that can be defined as time series, and considering the function f as being of additive nature, Equation (5) can be written as:

$$s'_{t+1} = f(a_t, \bar{x}_{t+1}) + \varepsilon_{t+1} \tag{6}$$

If $|\varepsilon_{t+1}| \ll |\bar{x}_{t+1}|$ (i.e., the estimation error of the time-series behaviour is very small), the Q-values obtained by employing $\bar{x}_{t+1}$ should not affect the choice of actions as the state reached $s'_{t+1} = s_{t+1}$, but this assumption does not usually hold for non-stationary environments, where ε_{t+1} can be large. This fact is key to our approach. We want to provide an estimate $\hat{x}_{t+1}$ to closely match x_{t+1}, the future environment behaviour. This can be used to train an RL agent offline, within a simulation of the environment based on the estimate. Note that there is an important difference between our simulation of the environment and the model of the environment in model-based approaches, for example in Dyna-Q [Sutton 1990]. Dyna-Q builds a model of the environment from experience and it is able to predict the next state and reward based on current state and action. Our approach is model-free, and does not build a model or explicitly predict the next state or reward; we instead simulate the environment outside of the agent framework, and let the RL agent implicitly learn suitable actions for this simulation.

P-MARL employs advanced forecasting techniques for non-stationary environments, where a close estimate $\hat{x}_{t+1}$ can be predicted, so that $\hat{x}_{t+1} \approx x_{t+1}$.

Our chosen application environment is inherently non-stationary. In mathematical terms, this means that the underlying generating function of the environment changes over time. The problem is simplified by modelling the outcome of the generating function of the environment based on recently observed behaviour. At every timestep t,

[1]POMDPs are outside the scope of this article, and the application of P-MARL to POMDPs needs to be evaluated separately.

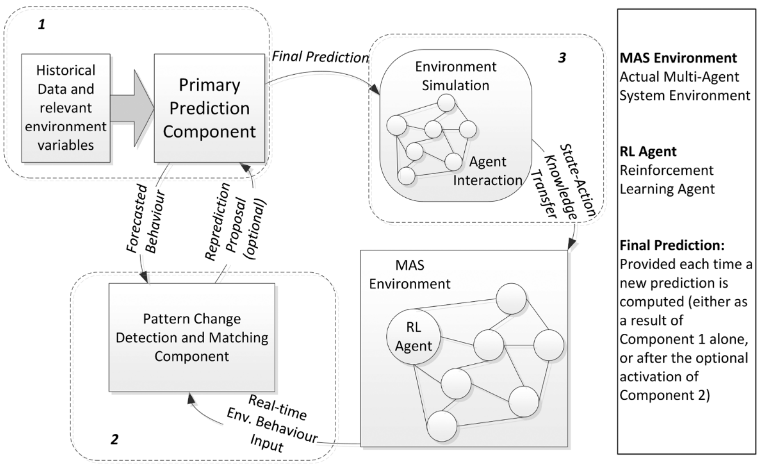
1
Historical Data and relevant environment variables
Primary Prediction Component
Final Prediction
3
Environment Simulation
Agent Interaction
State-Action Knowledge Transfer
Forecasted Behaviour
Reprediction Proposal (optional)
Pattern Change Detection and Matching Component
Real-time Env. Behaviour Input
2
MAS Environment
RL Agent
MAS Environment
Actual Multi-Agent System Environment
RL Agent
Reinforcement Learning Agent
Final Prediction:
Provided each time a new prediction is computed (either as a result of Component 1 alone, or after the optional activation of Component 2)

Fig. 3. P-MARL algorithm structure.

there is historic data available:

$$x^H = (x_1, x_2, \ldots, x_t),$$

where x_i is the state of the environment x at time i. We predict x_{t+1} considering past observations. This is a difficult task, since a non-stationary environment does not present fully repeatable patterns. Moreover, the uncertainty present in the environment needs to be accounted for separately, as anomalous events in the environment can lead to unexpected changes. The accuracy of predictions is critical in the training process of agents, as inaccurate predictions can make the agents take sub-optimal actions.

Depending on the speed of changes occurring in the environment, prediction systems need to:

(1) Periodically update the model of the future states of the time series based on the most recently observed samples (we have selected this approach as inspired by work in Widmer and Kubat [1996]);
(2) Trigger the generation of a new model of future time-series states once a pattern change detection mechanism notices an unexpected shift from the model's expected behaviour (as suggested in Alippi and Roveri [2008]).

The second part is more challenging. The new model is proposed based on a pattern-matching mechanism that should provide a close match, based on similar previously encountered behaviour. This, in turn, provides additional information in generating a better model.

Figure 3 illustrates the P-MARL architecture, which integrates environment prediction (via Primary Prediction and Pattern Change Detection/Matching components) with off-line simulation, to provide additional knowledge to the MAS component, as follows.

3.2. Prediction

P-MARL's time-series prediction system has two main components. The primary prediction component provides environment estimates periodically, forwarded to the MARL component as a training base. In parallel, once an estimate has been provided by the primary prediction component, its accuracy is continuously evaluated against the real-time behaviour of the environment by the Pattern Change Detection and Matching Component.

3.2.1. Primary Prediction Component. The primary prediction component uses both recent historic values of previous environment behaviour and key environment variables that influenced the environment's historic behaviour in order to provide an estimate of future behaviour. The model is a hybrid solution mainly comprising ANNs and ARIMA, taking advantage of these techniques' strengths for time-series prediction. A specific implementation of the hybrid solution for the smart grid domain is presented in Section 4, where further details are given on each of these techniques. As determined in our previous study [Marinescu et al. 2014b], some methods are more effective at specific sub-components of time series than others, and a combined solution is more accurate overall at predicting the complete time series. For example, given an estimate with a horizon of five time-points $(\hat{x}_{t+1}, \ldots, \hat{x}_{t+5})$, sub-technique A could be suited for elements $\hat{x}_{t+1}$ and $\hat{x}_{t+3}$, while sub-technique B could be suited for elements $\hat{x}_{t+2}$, $\hat{x}_{t+4}$, and $\hat{x}_{t+5}$. The best performance of sub-techniques needs to be investigated on the specific training dataset to achieve maximum possible accuracy [Marinescu et al. 2014b]. In general, a method consistently achieving higher accuracy than others over a time-series segment would be selected to forecast it. Considering a forecast of a set length, its comprising segments can be forecasted individually by the most suitable methods and then combined to provide an ensemble forecast. For example, in our initial experiments on the energy demand-side management prediction data, ANN performed well on non-linear data, such as morning/evening energy demand peaks, while ARIMA performed well on linear data, such as mid-day/late night energy demand. The prediction algorithm continues to monitor the suitability of the methods used for different segments of data, and evaluates their accuracy and switches methods as appropriate at fixed time intervals.

3.2.2. Pattern Change Detection and Matching Component. This component detects when the prediction model fails to provide accurate estimates of the future state of the environment. The current behaviour of the environment is evaluated by a SOM [Kohonen 1990]. SOMs are a form of ANN trained through unsupervised learning. Data is clustered into separate classes based on similarity. An initial set of previously encountered environment values are used for the training of the map.

A SOM has a number of classes available for classification, which are decided at the implementation stage. These depend on the approximate number of different types of samples. Once the training process is finished, each new sample fed into the SOM is assigned into one of the classes. Since there can be many different classes within a SOM, a design choice was to group these into two main categories: normal classes, encompassing most of the samples, and anomalous classes, comprising samples that are outside of the ordinary [Marinescu et al. 2014a]. The SOM is initially trained with historical data; new samples are classified by the SOM based on similarity with existing clusters. If the new sample is classified into an anomalous category, further measures need to be taken.

A predicted horizon $(\hat{x}_t, \hat{x}_{t+1}, \ldots, \hat{x}_{t+h})$, provided by the primary prediction component, is used to detect anomalies. The initial assumption is that for this horizon, the environment behaviour will not be different to the predicted behaviour, i.e., anomalous. As the environment evolves from time t to time $t + i$ between $(t, \ldots, t + h)$, we overlap the recently acquired information about the environment over the predicted horizon. Therefore, the sequence becomes $(x_t, x_{t+1}, \ldots, x_{t+i}, \hat{x}_{t+i+1}, \ldots, \hat{x}_{t+h})$. At each time $t + i$, the newly obtained sequence is inserted into the SOM. If the sequence is deemed to be anomalous, further adjustments need to be made since the previous historic data can result in inaccurate predictions. The sequence obtained so far is then compared with similar sequences from the corresponding anomalous class, and a closest match is found based on the k-nearest algorithm; this match is used to fill in the remaining time points in the sequence from the current time $t+i$. Fill-in points are adjusted according to the correlated current environment variables. If the behaviour is not deemed anomalous, no re-prediction will be executed. If the environment behaviour identified as anomalous does not lead to an actual anomaly (i.e., it was a false-positive detection), the re-prediction and matching processes will still be executed, but the resulting prediction will not be significantly different from the one provided before anomaly detection. Therefore, the performance of the agents will not be negatively affected.

In short, the hypothesis is that the closest previously encountered sample from the matched class of anomalies will increase prediction accuracy by triggering a new estimate, which is particular to anomalous cases.

3.3. The Multi-Agent System

As illustrated in Figure 3, and detailed in Algorithm 1, a final estimate of the environment's future expected behaviour is provided by the prediction module to the MAS agents. The MAS agents evaluate their performance based on this expected future behaviour and recalibrate their state-action values. This is a simulated, off-line process of exploration-exploitation, repeated by the agents until solutions converge, and they are ready to operate in online mode. The simulation process means they are better prepared to achieve a best-response function to the expected future behaviour of the environment. Even though the actual environment they face may be somewhat different from the estimate, the hypothesis is that the previously obtained knowledge will help them perform well in conditions similar to the ones provided by the estimate. In the case of Q-Learning, this translates to obtaining Q-values that allow agents to maintain the same policy both for the estimate and for the actual environment, since the environment states are expected to be approximated in a similar way.

ALGORITHM 1: P-MARL Algorithm

```
foreach new prediction do
    envVars ← updateEnvData();
    histData ← updateHistoricBehaviour();
    prediction ← makeInitialPrediction(envVars,histData);
    evaluatePrediction(prediction,currentEnvBehaviour);
    if no significant change detected then
        finalPrediction ← prediction
    else
        anomalyType ← matchChangeType();
        finalPrediction ← repredict(anomalyType,envVars,histData);
    end
    for each agent do in parallel
        reward agent based on decision and influence on env status;
        update Q-Values;
    end
    exploit learned info on actual env;
end
```

Once each agent receives a prediction of the environment behaviour, several types of learning can occur during the exploration stage of agents, depending on the communication restrictions imposed on the system:

1. *Single agents acting separately on the environment*: during exploration, each agent reaches a policy solely based on the effects occurring on the environment due to its own actions. An agent cannot see any of the effects of other agents on the environment. While the environment is updated only locally, the final state of the

environment will aggregate all actions taken, but this aggregated state is not accessible to agents (e.g., several slot-machine players attempting to win money from a casino). This is the case with most severe communication restrictions.
2. *Multiple agents acting simultaneously on the environment*: during exploration, an agent's action on the environment is taken into account together with the cumulative effect of all agents' actions. The agent can see this cumulative effect of all agents' actions only after it has chosen an action; therefore, agents choose actions simultaneously. This case assumes that there is only post-communication towards the agents, coming from an environment update once all agents have taken action (e.g., the stock market, where brokers take actions simultaneously).
3. *Multiple agents acting sequentially on the environment*: during exploration, agents take decisions at each time step in a sequential manner (e.g., an auction, where only the current bid matters and not previous ones). An agent will see the cumulative result of the previous agents' actions on the environment, and then decide which action to take. Once it has taken a decision, the updated cumulative effect on the environment is passed on to the following agent. The order of this sequence is random at each timestep; therefore, no agent is favoured during the length of a learning episode. This form of interaction requires an underlying protocol to maintain ordering between the agents and to avoid overlapping decisions.

3.4. P-MARL and the Requirements for Optimisation in Non-Stationary Environments

Section 2.4 listed a set of requirements for optimisation in non-stationary requirements, based on the existing approaches and the gaps in their applicability. P-MARL, as presented in this section, meets those requirements as follows:

—**Req. 1** Taking optimal actions in the environment is enabled by the Multi-Agent System component (Section 3.3). Agents use Q-learning to learn suitable actions for each of their states. Actions are pre-learned offline in a simulation facilitated by the prediction and simulation components, and further adjusted to new environment conditions online.
—**Req. 2** Detecting changes in the environment is done by the Pattern Change Detection and Matching Component (Section 3.2.2). This enables P-MARL to detect when environment prediction (as provided by the Primary Prediction Component, Section 3.2.1) differs from the actual environment behaviour.
—**Req. 3** Future environment behaviour is estimated by the Primary Prediction Component (Section 3.2.1). When changes from expected environment behaviour occur, the Pattern Change Detection and Matching Component is able to detect and classify the type of change in the environment and trigger reprediction, which is executed by the Primary Prediction Component. This is done to enable prediction of environment dynamics even when changes from expectations occur.
—**Req. 4** Pre-training for new dynamics (i.e., previously unencountered) is enabled by the Environment Simulation component (Figure 3). Re-predicted behaviour is simulated within this component and enables agents to adjust their preferred actions before acting online.

P-MARL detects changes in the environment (Behaviour Change Detection) and predicts environment behaviour (Behaviour Prediction) for both previously encountered dynamics and new dynamics without losing any of the features of the existing approaches to optimisation in non-stationary environments. Behaviour prediction in the new environment dynamics is enabled, which was not possible with other existing MARL approaches to non-stationary environments (see Table I). Optimisation in newly-encountered dynamics is further improved by enabling off-line instead of online learning implemented by the existing approaches, which reduces the negative impact on the environment during the training phase.

4. SMART GRID CASE STUDY

Smart energy grids are a good example of systems operating in non-stationary environments, which can be modelled as a time series. Environment non-stationarity in energy grids is caused both by non-stationary energy usage (e.g., energy use depends on the time of day, day of the week, weather) and non-stationary renewable energy generation, which depends on weather. In this case study, we apply P-MARL specifically to Demand Side Management (DSM). DSM techniques attempt to reduce peak-time energy usage by re-scheduling deferrable loads to times when the baseload (i.e., non-deferrable energy load) is lowest. These deferrable loads can be controlled by intelligent agents organised in an MAS and learn how to schedule energy usage to achieve their objectives, while avoiding peak-time energy usage.

4.1. Multi-Agent Systems and the Smart Grid

Several DSM solutions using MAS have been proposed, with particular focus on electric vehicle (EV) charging, which have high power consumption but charging time flexibility. Examples include: agent-based control algorithms based on grid prices and emergent coordination of agents [Ramchurn et al. 2011]; imbalance cost reductions through a multi-agent solution for coordinated plug-in hybrid EVs [Vandael et al. 2011]; EV fleets as virtual power plants for energy trading in wholesale markets [Kahlen et al. 2014]; and research into individual EVs, which benefit from smart charging strategies based on learning agents [Valogianni et al. 2014]. However, the proposed approaches do not account for potential changes in the baseload (e.g., unexpectedly cold weather leading to additional electric heating in the home), and so do not adapt, leading to sub-optimal performance if the baseload does change.

We evaluate P-MARL in a standard DSM scenario, where EVs in a community of households try to optimise their battery charging. The EVs coordinate their usage with other EVs to decrease peak-energy usage and best utilise nighttime energy usage troughs. We monitor and predict energy baseload and aim to improve agent charging performance by providing accurate baseload predictions.

4.2. Primary Prediction Component Configuration

In the smart grid scenario, EVs arrive home in the evening and depart in the morning. EV agents need to be aware of periods of low demand while they are home in order to be as cost effective as possible. In a non-stationary environment, though, such *a priori* knowledge is not available, as environment conditions may directly influence demand. An initial analysis of how the environment evolves needs to be performed to provide the input required to configure the primary prediction component, and thereby provide the environment predictions that the EV agents need. In general, the defining characteristic of a relevant environment is examined in relation to other environment variables, and any dependent variables found should also be employed in the prediction component. Furthermore, seasonal patterns and trend lines in the time-series representation need to be addressed when customising the prediction model.

Short-term load forecasting (STLF) focuses on day-ahead demand forecasting, providing an estimate every 24 hours. This is appropriate for power demand estimates, as energy demand is seasonal at day level. Such forecasts rely on historic values of previous power demands, and also on other data such as weather variables (temperature, humidity), day of the week, and public holidays [Gross and Galiana 1987]. While weekdays and weekends differ significantly in terms of demand patterns, each weekday also poses somewhat different characteristics. Anomalous days (from a demand perspective) also occur, mostly because of public holidays but also because of other unexpected reasons, such as snap cold periods. These anomalous days need to be accounted for separately.

P-MARL's forecasting method is a hybrid solution exploiting the best features of several forecasting techniques: ANN, Neuro-Fuzzy Networks, and auto-regression. The hybrid solution uses as input previously recorded power demands, past day's temperature and humidity information, temperature and humidity forecasts for the day to be predicted, and day of the week information. The output is the next day's power demand estimate, provided as a sequence of 24 data points, one for each hour of the day.

In the neural network, the input neurons are divided as follows:

—24 neurons used for previous load input, one for each hour of the day
—5 neurons are used for the day code input, for each day of the week (weekends are excluded)
—14 neurons are used for weather forecast input along the day, 8 for temperature and 6 for humidity; more inputs were chosen for temperature because of its higher correlation with energy consumption

The output layer comprises 24 neurons, which represent the short term load forecast. Each of the output neurons provides a demand estimate belonging to an hour of the day, in consecutive order.

Baseload demand prediction is based on the weather forecast for the day to be predicted, together with the historical recorded demand occurring over the same day of the previous week. The reason for choosing the load belonging to the same day of the week for prediction (e.g., previous Tuesday is used for predicting the following Tuesday) is that each day of the week tends to have a particular demand shape.

The ARIMA model, another key component of the prediction system, is based on a whole week's recorded demand, occurring before the day to be forecasted. Thus, a total of 120 (5×24) coefficients are taken into account in the model, one for each hour of the week [Marinescu et al. 2014b].

This technique relies only on the analysis of the time-series representation of the environment's behaviour. In this particular case study, the forecasted energy load is provided as input. However, the technique can be similarly used for other domains, for example: in vehicular traffic control, by providing traffic flow forecasts to traffic light agents; in cloud computing, by employing a daily forecast of expected computational load to resource allocation agents; or data traffic information forecasts provided to telecommunications node agents for these to decide routing tables. Prediction of future behaviour from any given dataset can be provided by a hybrid combining several best-performing techniques specific to a domain, as described here for the smart grid scenario. Alternatively, it can be provided using only a single technique, if, based on the data sample available, it provides a suitable level of accuracy. Any of the state-of-the-art time-series analysis techniques, as discussed in Section 2.3, can be integrated into P-MARL, as only the behaviour prediction is needed, regardless of the techniques and algorithms used to obtain it.

4.3. Pattern Change Detection and Matching Component Configuration

In situations involving uncertainty, quality of service guarantees cannot be provided. Even if previous models generate accurate predictions, there are particular times when forecasts fail to closely match actual behaviour. In smart grids, anomalous situations are caused by anomalous events such as unexpected climate phenomena or particular socio-economic events. The pattern change detection component makes the system more robust in the face of such events.

Once the prediction model provides a forecasting estimate for the next day, the actual demand is compared with the estimate along the first few hours of the morning (the evening part is the period of highest demand and needs to be accommodated for). The custom SOM used for classification in this case comprises four classes: two classes for normal days (one for cold/winter days and one for warm/summer days), and two anomalous classes (one for public holidays and one for other particular days).

If significant deflections from the actual demand occur, the day will be classified in one of the anomalous classes. In this case, the pattern change detection mechanism triggers reprediction, since the demand estimate is regarded as flawed. A match is chosen based on similar patterns encountered previously, which are found in a database of historic recordings. After the self-organising map classifies the type of uncertainty detected, it provides the closest previously encountered match from the fitting class.

The reprediction component is based on an ANN which adjusts its historic load input part (24 neurons, one for each hour of the day) to a combination between the actual demand observed so far of the anomalous day (7:00-14:30 interval) and the remaining demand from the closest match provided by the SOM component (14:30-23:30 interval). This constructed sequence is input to the ANN, which also takes into account current weather and day of the week conditions to provide another forecast of the day. It is expected that the results will be more accurate for the evening demand interval.

This technique can be used to detect changes and adjust independently of the application domain. For example, when new vehicle traffic patterns are detected, these can be matched with some known situations (e.g., traffic in rainy conditions) and this information can be further used by agents for calibration; when high traffic is generated by a particular webpage, this can be matched with patterns occurring after exposure to social media, for agents to allocate additional server resources; or when there is high traffic in a particular telecommunication network area due to a social event, additional telecom stations can be temporarily dispatched. Selection of normal and anomalous classes for the SOM is straightforward, with a little knowledge of the domain.

4.4. P-MARL System Implementation

P-MARL integrates the prediction and pattern change components with a MARL system. The RL process is implemented using W-Learning [Humphrys 1995], which combines two Q-Learning objectives: (1) sufficiently charge an EV for the next day's trip considering the time available for charging, and (2) avoid generating peaks in demand. W-learning is a technique for multi-policy action selection, where each policy is implemented as a Q-learning process, and which works by minimising the loss that any of its comprising Q-learning process can incur when their nominated action is not being executed.

In our case, the agent is rewarded more for the first objective than for the second one, as otherwise, the agent would avoid charging at all. By combining both objectives, the agent should only generate demand during times of low energy usage. Once the agent achieves the minimum charge required for the daily trip, the reward for charging decreases, so that peak-avoidance can be prioritised.

The environment forecast provided by the prediction components is used to find the times of lowest energy usage. Since we employ tabular Q-learning, for the agent's state aggregation, the forecasted load is analysed, and afterwards, discretised into 10 different levels (or RL states) depending on the amount of power used. State aggregation is necessary because otherwise there are an infinite number of possible states between minimum load and maximum load (e.g., the [0-300]kW interval). The lowest discretised states will present the highest reward to the agent, therefore encouraging it to charge during intervals of low demand. However, the load levels are expected to change also due to the action of other agents in the environment; therefore, the reward will also depend on their actions.

In general, when employing an environment described through time series, rewards for actions should be given depending on the state of the environment (i.e., depending on whether an action was taken during low/high load).

4.5. Benchmark: Optimal Centralised Solution

We have defined the Pareto optimal performance of this problem to evaluate against P-MARL. Our optimal centralised solution aims to optimise the following cost function:

$$\min F(x) = \min \sum_{j=1}^{m} \left[\sum_{i=1}^{n} \left(x_{ij} + C_j \right) \right] x_{ij}, \tag{7}$$

where $F(x)$ is the cost function, n the total number of EVs, m the total hours available for charging (assuming the same availability schedules for EVs), x_{ij} the charging decision of vehicle i at time j (0 for not charging and 1 for charging), and C_j the initial cost of energy at time j (based on baseload). This is a binary integer programming problem, therefore NP-complete [Karp 1972]. While a centralised solution is not feasible in most situations when a problem is NP-complete, it is guaranteed to be optimal with respect to a defined set of constraints, and so is an interesting benchmark for comparison purposes. We solve this as a valley-filling problem, using the approach presented in Gan et al. [2011]. The problem is solved centrally, by pre-calculating the static solution for the day ahead. Each EV is taken in turn, the minimum amount of charging slots required is computed, and then charging slots are allocated in the periods of low demand. Each EV incrementally updates the overall demand until all EV charging slots are allocated.

The best performance that can be achieved by the MARL technique should have the same aggregated effect as the valley-filling algorithm. In order to evaluate the optimality of the MAS solution we used a formula derived from the mean absolute percentile error (MAPE). This is shown in Equation (8).

$$M = \frac{1}{m} \sum_{j=1}^{m} \left(1 - \frac{|X_j - \hat{X}_j|}{TotalNoOfEVs} \right), \tag{8}$$

where X_j is the total number of EVs charging at time-slot j, and $\hat{X}_j$ is the optimal amount of EVs that should be charging at time-slot j. Note that this formula actually represents the complement of error, as opposed to MAPE.

5. EXPERIMENTAL RESULTS

Power demands from a community of 230 households are employed, as recorded by an Irish smart meter trial [Comission for Energy Regulation, Ireland 2011]. This is roughly the number of houses provided for by a typical 630kVA residential transformer [Marinescu et al. 2013]. We have also assumed an EV penetration rate of 40% [Nemry and Brons 2010], resulting in a total of 90 EVs. A daily trip is considered to be 50km [EPA 2008], while the EV specifications are based on the Nissan Leaf characteristics [U.S. EPA Fuel Economy Information 2014]. The vehicles can choose 15 minutes charging slots anytime between 18:00 and 09:00. This smart grid scenario was implemented in GridLAB-D, a power distribution system simulator [U.S. Department of Energy at Pacific Northwest National Laboratory 2014], where a standard charging rate of 1.4kW was used.

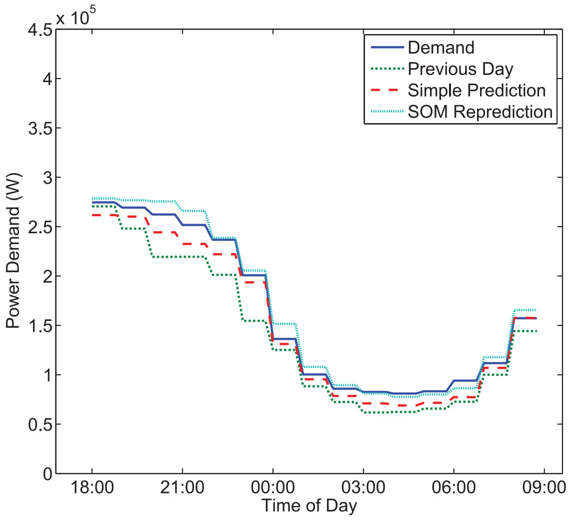

Fig. 4. Predictions of an anomalous day.

Three sets of experiments are presented here, with different levels of interaction between agents and environment. These reflect the cases presented in Section 3.3. For these experiments, we chose an anomalous day from the dataset in order to identify P-MARL performance differences between employing only the primary prediction component and a combined system that also involves pattern-change detection and matching abilities.

All 90 EV agents begin the training sessions with a state of charge (SOC) of 0% (i.e., empty battery). Their purpose is to sufficiently charge for the next day's trip. Agents are trained offline over a period of 100 days, in a simulation of the environment. The simulation is based on a prediction of the environment's future behaviour. After the training session, agents are tested on the actual environment. Each set of experiments is performed 10 times and averaged.

The load from the same day of the previous week and three different load predictions are provided in each set of experiments as input to P-MARL: simple prediction (no pattern matching), anomaly-matching prediction (SOM Prediction), and perfect prediction (i.e., the estimate is the same as the actual environment behaviour). The load of the previous week is used in a traditional MARL implementation, which is based only on previously encountered situations. Perfect prediction represents an idealised situation, where the environment induced non-stationarity is removed from the MARL problem. This latter case is used only for comparison purposes, to differentiate between the levels of performance achievable by the first two prediction types.

The evening and early morning periods of an anomalous day used, together with its predictions and the same day of the previous week, are illustrated in Figure 4. The anomalous day occurs in December 2010; a comparison with other winter days reveals a higher amount of energy usage in the anomalous day. This anomaly occurs because of an unexpected cold front advancing from the North Atlantic, which caused increased power usage. The previous day and simple prediction underestimate the actual demand, while the SOM reprediction overestimates the demand. In this case, SOM reprediction takes a more conservative measure. The previous day has a forecasting error of 14.87% MAPE, simple prediction achieves a forecasting error of 7.65% MAPE, while SOM reprediction achieves 4.66% MAPE. These results show that significant increase in

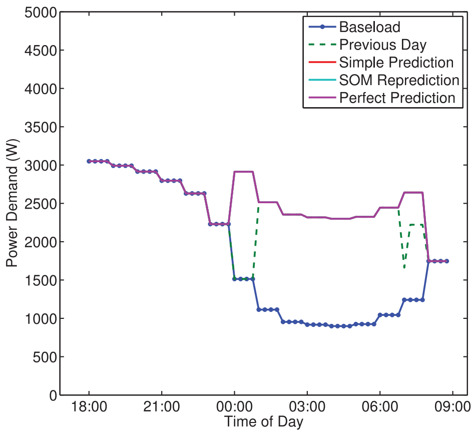

Fig. 5. Single EV charging.

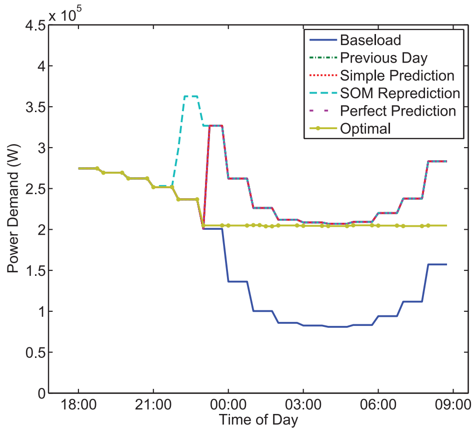

Fig. 6. EVs charging separately.

Table II. Algorithms Efficiency in Different Prediction Conditions

	Pareto Performance			
	MARL	**P-MARL**		
MARL Agent Interaction	Prev. Day	Simple Pred.	SOM Repred.	Perfect Pred.
1. Separate Actions	83.22%	83.22%	75.78%	83.22%
2. Simultaneous Actions	78.94%	86.95%	85.88%	86.71%
3. Sequential Actions	97.76%	94.24%	93.68%	95.20%

forecasting accuracy can be obtained using P-MARL's primary prediction component, and even further improvements by using pattern change detection, matching, and re-prediction.

Before considering the multi-agent scenario, as a preliminary study, we have first validated our algorithm's ability to address a non-stationary environment when applied at single-agent level (i.e., focusing only on environment-induced non-stationarity without the impact of agent-contributed non-stationarity). We investigated the impact of different levels of accuracy of non-stationary environment prediction, considering only one EV agent within a single household. For this, we have scaled down the community's demand to approximate the demand of one house. Figure 5 presents the results obtained by the EV agent while training on the four different types of estimation for the day's demand. Note that the curves when employing the three predictions (i.e., simple prediction, SOM reprediction, and perfect prediction) fully overlap.

When training on the previous day, which has lower demand, the EV agent fails to charge sufficiently 60% of the time given 10 trials. This can be noticed also in Figure 5, as the EV trained on the previous day starts charging too late and stops charging too early, since it incorrectly assumes that the demand is too high at those times. On the other hand, accuracy differences between the three predictions in this particular scenario do not impact the performance of a single agent. When using any of the predictions for training, the EV agent will charge sufficiently 100% of the time.

In the following sections, we investigate the impact of prediction accuracy on overall P-MARL performance. The summary of the results are presented in two tables. Table II shows the Pareto efficiency of the MARL solutions with regard to an optimal cumulated demand pattern; the optimal performance is defined by an aggregate behaviour where agents charge precisely the amount necessary for their daily trip while generating

Table III. Percentage of EVs Charged Before Departure

	Percentage of Charged EVs			
	MARL	**P-MARL**		
MARL Agent Interaction	Prev. Day	Simple Pred.	SOM Repred.	Perfect Pred.
1. Separate Actions	100%	100%	100%	100%
2. Simultaneous Actions	16.4%	95.7%	100%	100%
3. Sequential Actions	82.3%	83.9%	100%	100%

perfectly smoothed demand on aggregate. Table III presents the percentage of EVs achieving their target charge during the allocated charging interval.

Case 1: EVs Charging Separately (Figure 6). Each agent has a view only of its local environment, and is only aware of its own impact on the overall system performance. As each EV adds a relatively small load to the overall energy load, regardless of the load and load prediction, each EV perceives that there is enough capacity for it to charge. Therefore, in all prediction method cases, all EVs end up fully charged 100% of the time (Table III). As a consequence, multiple vehicles end up charging simultaneously, increasing the overall load, which achieves only up to 83% Pareto optimality regardless of the prediction method used (Table II). The SOM method generates a peak earlier, which affects its Pareto efficiency even more, since it tends to overestimate the demand (see Figure 4). EVs therefore start charging once the load goes under their initial expectations. We have performed additional tests on statistical signficance through two-tailed t-tests (where results were considered to be statistically significant if their p-value is lower than 0.05). The performance differences in terms of Pareto performance between simple prediction/perfect prediction and SOM reprediction are statistically significant, with a p-value of $1.21e-11$ over the 10 different runs. Since all vehicles end up charged all the time, there is no statistical difference between numbers of charged vehicles.

Case 2: EVs Charging Simultaneously (Figure 7). In this case, all agents are aware of their aggregated impact on the environment, and all agents take actions simultaneously, taking the overall impact in to account. The results show the impact and importance of accuracy of prediction on the end performance in such a scenario. In terms of Pareto optimality, both simple and SOM reprediction result in 7% improvement over using the previous day's load as prediction (Table II). In terms of battery charge, when using previous day's load, only 16.4% of EVs achieve the required charge. Using simple prediction significantly improves that to 95.7%, while SOM reprediction achieves the required charge on 100% of EVs, same as the perfect prediction (Table III). The difference in Pareto performance between perfect prediction and SOM reprediction runs is statistically significant, with a p-value of 0.0058, while simple prediction and SOM reprediction runs failed to show any significant differences, with a p-value of 0.10. In terms of percentage of vehicles charged, the difference was noticed to be statistically significant between simple prediction and the SOM reprediction/perfect prediction cases, with a p-value of 0.0002.

Case 3: EVs Charging Sequentially (Figure 8). In this case, agents take actions one after another, so each EV is aware of the aggregated impact of all of the previous agents on the power demand, before deciding whether to charge or not. Accuracy of prediction does not have a substantial impact on Pareto optimality of the solution, and less accurate prediction sometimes even results in better performance (Table II). Since simple prediction underestimates demand, and some agents charge more than required, the aggregate demand will get closer to the optimal line. However, in terms of

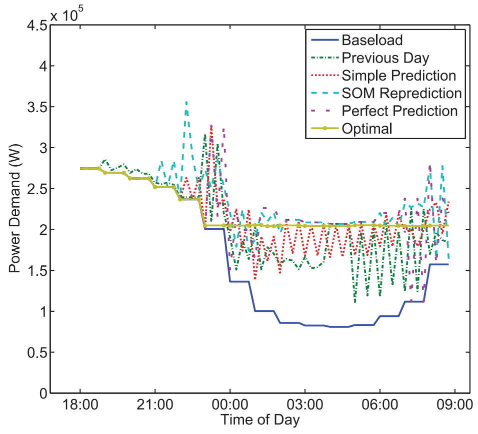

Fig. 7. EVs charging simultaneously.

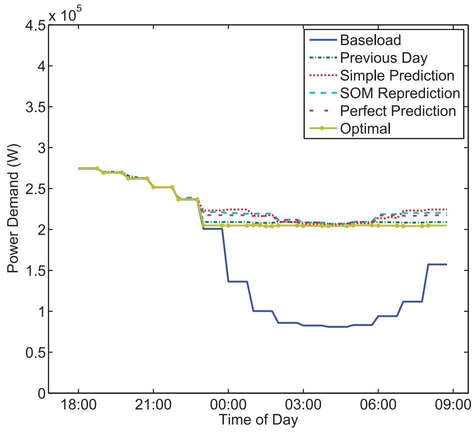

Fig. 8. EVs charging sequentially.

meeting agents' objectives, again, only SOM reprediction achieves the required charge on 100% of EVs, while using previous day load and simple reprediction do so only for 82.3% and 83.9% of EVs, respectively (Table III). The difference between SOM results and the ones obtained by using simple prediction are statistically significant, with a p-value of $1.25e-7$. The Pareto optimality differences between the two situations are very small, however, still statistically significant with a p-value of 0.0005. Also, there is a statistically significant difference between perfect prediction runs and SOM reprediction runs, with a p-value of 2e-9 with regard to Pareto optimality.

We observe the difference between the single-agent and multi-agent cases; in the single-agent case, different predictions used did not have any impact on the charging outcomes, as the EV ended up fully charged during all the trials for all prediction types. However, when multiple agents are involved, the accuracy of prediction can make the difference between agents reaching their charging objective or not, as can be seen in Table III. This confirms the requirement for very accurate predictions of the environment's behaviour, as achieved by complex time-series prediction methods used by P-MARL.

5.1. Results Analysis

The quality of prediction and an agent's level of interaction with the environment and other agents has a high impact on the aggregate P-MARL performance. When agents are not informed of the effect of other agents' decisions (case 1), the aggregate demand has negative consequences on the environment. The peak demand that results in this case (as can be seen in Figure 6 around midnight) forces the scheduling of additional power generation units, or even a blackout at neighbourhood level. This peak effect is noticeable regardless of the level of prediction accuracy.

Once agents have access to aggregate charging decisions (case 2), they notice their cumulative effect. When their aggregated behaviour leads to peaks, agents will back-off at the next timestep. The problem is that they can all resume charging two timesteps later as they register a low demand once all have backed off. This alternating behaviour is illustrated in Figure 7 and is noticeable, in particular, when using the previous day's load as prediction and in the simple prediction situation. Here, since the nighttime prediction is underestimated, agents believe that their aggregate behaviour is generating new peaks. This is not the case in reality, and as a result, some agents do not achieve their target charge because they expect time slots of lower demand to follow (as seen in Table III).

Informing agents of the cumulative effect of other agents before making their own decision leads to a considerable performance improvement (case 3). Peak demand is avoided, and agents' aggregate performance closely matches the optimal line, as illustrated in smooth lines in Figure 8. In this case, prediction quality also affects the number of EVs achieving their target SOC. Some EVs trained on the previous day and simple prediction cases do not charge enough before their departure, despite the high Pareto efficiency achieved on aggregate. This is because some of the first agents to make a decision still decide to charge even though their SOC is sufficient for their trip, as they are rewarded when the load is low enough to allow that. Therefore, some of the following agents that are not sufficiently charged, and that are about to make a decision, will encounter a load state where they believe that charging negatively impacts the grid, resulting in them avoiding charging.

An investigation of the 10 different runs reveals consistent results in case 1 and 3. In case 2, there is a larger amount of variability involved, in particular, during midnight and morning hours. Some time slots with higher variability occur because of the effect agents have when acting in bulk. If all agents charge in these particular time slots, their aggregate demand can result in peak demand, a situation in which all agents are penalised. They will immediately back off in this situation for the next time slot, but will start again for the one after.

In summary, P-MARL provides improved performance over the traditional MARL approach with regard to the most important criteria: agents achieving their charging objectives. When SOM reprediction is provided and agents interaction exists (case 2 and 3), all P-MARL agents achieve their objectives, as opposed to only 16.4% of agents in simultaneous MARL and 82.3% in sequential MARL. In terms of Pareto efficiency, P-MARL performance is similar to MARL, except for the simultaneous actions case: here MARL performs considerably worse. This is because MARL agents consistently back off as they erroneously assume that their bulk charging is generating peaks in demand.

6. CONCLUSIONS AND FUTURE WORK

This article presented P-MARL, a novel approach that extends MARL by integrating advanced time-series analysis techniques to address inherently non-stationary environments. The effect of environment-induced non-stationarity is reduced through prediction and pattern-change detection and matching techniques, which provide agents with additional information about the upcoming behaviour of the environment. In effect, these extensions enable MARL to reduce the time needed for online learning, and minimise the negative impact agents have on the environment while learning how to optimise their behaviour in the new environment conditions.

P-MARL can be applied to non-stationary environments whose behaviour can be characterised as time series, and whose future behaviour is not completely random. In other words, where external influencing factors render the environment's behaviour as somewhat predictable through additional information sources. In this article, we used a smart grid scenario as an example of such an environment and have shown that P-MARL obtains significant improvements over MARL in a simulation of residential energy demand response.

P-MARL has a number of limitations we plan to address in our future work. For example, it includes an online environment simulation component, which assumes an online connection between agents in the simulation environment. If agents cannot interact within a simulation of the environment, their performance is reduced to the first type of interaction presented, single-agents acting separately, reducing the benefits that are achievable by the use of P-MARL. This could be addressed by providing alternative communication abilities, such as agents exchanging messages directly in order to inform others of their decisions.

Another assumption in the evaluation of P-MARL was that all agents have the same global objectives, in addition to an individual local objective. However, the MAS agents could have conflicting objectives, or even involve malicious agents; therefore, the convergence of P-MARL in such conditions should be investigated. We plan to base a solution to this problem on multi-objective RL optimisation techniques such as DWL (Distributed W-Learning) [Dusparic and Cahill 2010], rather than plain Q-learning.

The performance of P-MARL will also be evaluated in other non-stationary environments, for example, as discussed in Section 4.2, vehicular traffic, network traffic, and load balancing. Previous experience with portability of RL-based algorithms across domains with similar characteristics (such as, for example, DWL, which has been applied in smart grids, urban traffic control [Dusparic and Cahill 2010], and smart camera networks [Rudolph et al. 2014]) leads us to expect a relatively similar process.

Finally, as discussed in Section 2.2, non-stationarity in MARL can be environment-induced or contributed by the interacting agents; P-MARL addresses only environment-induced non-stationarity. Integration of P-MARL with the presented state-of-the-art techniques for addressing agent-contributed non-stationarity should be investigated to provide a complete solution to optimisation in non-stationary environments.

REFERENCES

Cesare Alippi and Manuel Roveri. 2008. Just-in-time adaptive classifiers - Part II: Designing the classifier. *IEEE Transactions on Neural Networks* 19, 12 (2008), 2053–2064.

George E. P. Box and Gwilym M. Jenkins. 1970. *Time Series Analysis, Forecasting and Control*. San Francisco, CA: Holden Day.

Martin Brown and Christopher John Harris. 1994. Neurofuzzy Adaptive Modelling and Control. Prentice Hall.

Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 38, 2 (2008), 156–172.

Samuel P. M. Choi, Dit-Yan Yeung, and Nevin L. Zhang. 2001. Hidden-mode Markov decision processes for nonstationary sequential decision making. In *Sequence Learning*. Springer, 264–287.

Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI*. 746–752.

Comission for Energy Regulation, Ireland. 2011. Smart Meter Trial Data. Retrieved from http://www.ucd.ie/issda/data/commissionforenergyregulationcer/.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3 (1995), 273–297.

Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. 2002. Multiple model-based reinforcement learning. *Neural Computation* 14, 6 (2002), 1347–1369.

Ivana Dusparic and Vinny Cahill. 2010. Multi-policy optimization in self-organizing systems. In *Self-Organizing Architectures*. Springer, 101–126.

Ivana Dusparic, Colin Harris, Andrei Marinescu, Vinny Cahill, and Siobhán Clarke. 2013. Multi-agent residential demand response based on load forecasting. In *Proceedings of the 2013 1st IEEE Conference Technologies for Sustainability (SusTech)*. IEEE, 90–96.

Mohamed Elidrisi, Nicholas Johnson, Maria Gini, and Jacob Crandall. 2014. Fast adaptive learning in repeated stochastic games by game abstraction. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'14)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1141–1148.

EPA. 2008. *Average Annual Emissions and Fuel Consumption for Gasoline-fueled Passenger Cars and Light Trucks*. Technical Report. United States Environmental Protection Agency.

Lingwen Gan, Ufuk Topcu, and Steven Low. 2011. Optimal decentralized protocol for electric vehicle charging. In *Proceedings of the 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*. IEEE, 5798–5804.

George Gross and Francisco D. Galiana. 1987. Short-term load forecasting. *Proc. IEEE* 75, 12 (1987), 1558–1573.

John A. Hartigan and Manchek A. Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics* (1979), 100–108.

Pablo Hernandez, E. Munoz de Cote, and L. Enrique Sucar. 2013. Learning Against Non-stationary Opponents. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, Saint Paul, MN*.

Junling Hu and Michael P. Wellman. 2003. Nash Q-learning for general-sum Stochastic games. *The Journal of Machine Learning Research* 4 (2003), 1039–1069.

Markus C. Huebscher and Julie A. McCann. 2008. A survey of autonomic computing: degrees, models, and applications. *ACM Comput. Surv.* 40, 3, Article 7 (Aug 2008), 28 pages.

Mark Humphrys. 1995. W-learning: Competition Among Selfish Q-learners. Departmental Technical Report. University of Cambridge.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable Stochastic domains. *Artificial Intelligence* 101, 1 (1998), 99–134.

Micha Kahlen, Wolfgang Ketter, and Jan van Dalen. 2014. Agent-coordinated virtual power plants of electric vehicles. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1547–1548.

Richard M. Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Springer, 85–103.

Franziska Klügl, Manuel Fehler, and Rainer Herrler. 2005. About the role of the environment in multi-agent simulations. In *Environments for Multi-agent Systems*. Springer, 127–149.

Teuvo Kohonen. 1990. The self-organizing map. *Proc. IEEE* 78, 9 (1990), 1464–1480.

Julien Laumônier and Brahim Chaib-draa. 2005. Multiagent Q-learning: Preliminary study on dominance between the Nash and Stackelberg equilibriums. In *Proceedings of AAAI-2005 Workshop on Multiagent Learning.*

Andrei Marinescu, Ivana Dusparic, Colin Harris, Vinny Cahill, and Siobhán Clarke. 2014a. A dynamic forecasting method for small scale residential electrical demand. In *IJCNN*. 3767–3774. DOI:http://dx.doi.org/10.1109/IJCNN.2014.6889425

Andrei Marinescu, Collin Harris, Ivana Dusparic, Vinny Cahill, and Siobhán Clarke. 2014b. A hybrid approach to very small scale electrical demand forecasting. In *Innovative Smart Grid Technologies (ISGT), 2014 IEEE PES*. 1–5. DOI:http://dx.doi.org/10.1109/ISGT.2014.6816426

Andrei Marinescu, Colin Harris, Ivana Dusparic, Siobhán Clarke, and Vinny Cahill. 2013. Residential electrical demand forecasting in very small scale: An evaluation of forecasting methods. In *Proceedings of the 2013 2nd International Workshop on Software Engineering Challenges for the Smart Grid (SE4SG)*. IEEE, 25–32.

Francoise Nemry and Martijn Brons. 2010. *Plug-in Hybrid and Battery Electric Vehicles. Market Penetration Scenarios of Electric Drive Vehicles*. Technical Report. Institute for Prospective and Technological Studies, Joint Research Centre.

Sarvapali D. Ramchurn, Perukrishnen Vytelingum, Alex Rogers, and Nick Jennings. 2011. Agent-based control for decentralised demand side management in the smart grid. In *Proceedings of the he 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 5–12.

Stefan Rudolph, Sarah Edenhofer, Sven Tomforde, and Jörg Hähner. 2014. Reinforcement learning for coverage optimization through PTZ camera alignment in highly dynamic environments. In *Proceedings of the International Conference on Distributed Smart Cameras (ICDSC'14)*. ACM, New York, NY.

As'ad. Salkham and Vinny Cahill. 2010. Soilse: A decentralized approach to optimization of fluctuating urban traffic using reinforcement learning. In *Proceedings of the 2010 13th International IEEE Conference on Intelligent Transportation Systems (ITSC).* 531–538. DOI:http://dx.doi.org/10.1109/ITSC.2010.5625145

Yoav Shoham, Rob Powers, and Trond Grenager. 2003. Multi-agent Reinforcement Learning: A Critical Survey. Technical report, Stanford University.

Bruno C. Silva, Eduardo W. Basso, Ana L. C. Bazzan, and Paulo M. Engel. 2006. Dealing with non-stationary environments using context detection. In *ICML*. ACM, 217–224.

Peter Stone and Manuela Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots* 8, 3 (2000), 345–383.

Richard S. Sutton. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning*. 216–224.

Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning*. MIT Press.

Gerald Tesauro, Nicholas K. Jong, Rajarshi Das, and Mohamed N. Bennani. 2006. A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation. In *Proceedings of the IEEE International Conference on Autonomic Computing (ICAC'06)*. 65–73. DOI:http://dx.doi.org/10.1109/ICAC.2006.1662383

U.S. Department of Energy at Pacific Northwest National Laboratory. 2014. GridLAB-D. Retrieved from http://www.gridlabd.org/.

U.S. EPA Fuel Economy Information. 2014. Nissan Leaf. Retrieved from http://www.fueleconomy.gov/feg/Find.do?action=sbs&id=32154.

Konstantina Valogianni, Wolfgang Ketter, and John Collins. 2014. Learning to schedule electric vehicle charging given individual customer preferences. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1591–1592.

Stijn Vandael, Nelis Boucké, Tom Holvoet, Klaas De Craemer, and Geert Deconinck. 2011. Decentralized coordination of plug-in hybrid vehicles for imbalance reduction in a smart grid. In *Proceedings of he 10th International Conference on Autonomous Agents and Multi-agent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 803–810.

Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3–4 (1992), 279–292.

Peter Whittle. 1951. *Hypothesis Testing in Time Series Analysis*. Vol. 4. Almqvist & Wiksells.

Gerhard Widmer and Miroslav Kubat. 1996. Learning in the presence of concept drift and hidden contexts. *Machine Learning* 23, 1 (1996), 69–101.

Guoqiang Zhang, B. Eddy Patuwo, and Michael Y. Hu. 1998. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* 14, 1 (1998), 35–62.

Received July 2015; revised March 2017; accepted March 2017