Comparing Policy Gradient and Value Function Based Reinforcement Learning Methods in Simulated Electrical Power Trade

Richard Lincoln, Stuart Galloway, Bruce Stephen, *Member, IEEE*, and Graeme Burt, *Member, IEEE*

***Abstract*—In electrical power engineering, reinforcement learning algorithms can be used to model the strategies of electricity market participants. However, traditional value function based reinforcement learning algorithms suffer from convergence issues when used with value function approximators. Function approximation is required in this domain to capture the characteristics of the complex and continuous multivariate problem space. The contribution of this paper is the comparison of policy gradient reinforcement learning methods, using artificial neural networks for policy function approximation, with traditional value function based methods in simulations of electricity trade. The methods are compared using an AC optimal power flow based power exchange auction market model and a reference electric power system model.**

***Index Terms*—Artificial intelligence, game theory, gradient methods, learning control systems, neural network applications, power system economics.**

I. Introduction

With growing world population comes increasing demand for energy and with it, demand for the fuels used to generate electricity. Competitive markets have an important role to play in the management of this demand and the subsequent price paid for electricity by consumers. Market designs for electricity are unique among commodity markets and new architectures are expensive and risky to implement. The importance of electricity to society makes it impractical to experiment with radical changes to trading arrangements on real systems. As an alternative it is possible to study abstract mathematical models of markets with sets of appropriate simplifying approximations and assumptions applied. Market architecture characteristics and the consequences of proposed changes can be established by simulating the models as digital computer programs. Competition between participants is fundamental to all markets, but the strategies of human participants are a challenge to model mathematically.

Unsupervised reinforcement learning algorithms from the field of artificial intelligence can be used to represent adaptive behavior in competing players [1] and have been shown to be capable of learning highly complex strategies [2]. Individuals participating in an electricity market (be they representing generating companies, load serving entities, or firms of traders) must utilize voluminous multidimensional data to their advantage. Data may be noisy, sparse, corrupt, have a degree of uncertainty (e.g., demand forecasts), or be hidden from the participant (e.g., competitor bids). Reinforcement learning algorithms must also be capable of operating with data of this kind if they are to successfully model participant strategies.

Traditional reinforcement learning algorithms attempt to learn a function that returns the long-term expected reward of each action available in a given state. Generalization techniques can be used to approximate this *value function* and allow continuous multi-variate state and action spaces to be used. However, it has been found that the greedy updates used with most techniques can prevent these algorithms from generalizing even in simple problems [3]–[5].

Policy gradient algorithms are an alternative form of reinforcement learning method that do not learn a value function, but adjust an agent's policy directly [6]. They can be used with function approximation techniques without suffering from the problems that mar value function based methods. They have been successfully applied in several types of operational setting, including robotic control [7], financial trading [8], [9], and network routing [10], but they have not been previously applied in simulations of competitive electricity trade.

In this paper, two policy gradient algorithms are compared with one value-function based method and two variants of the popular Roth-Erev technique [11]. A power exchange auction market model is used to facilitate trade between learning agents and AC optimal power flow solutions are used to clear submitted offers. The IEEE Reliability Test System provides a reference electric power system model that provides dynamic load profiles and a realistic generation mix with associated costs [12]. Learning agents are each endowed with roughly equivalent portfolios of generating stock. The agents can mark-up offer prices above marginal cost and withhold generating capacity within specified limits. The algorithms are compared in their profitability over a simulated year of trade.

The remainder of this paper is organized as follows: Section II provides an introduction to reinforcement learning and the algorithms under test. Related research is reviewed in Section III. In Section IV the power exchange auction market model and the design of a multi-learning-agent system is defined. Details of the simulation setup are given in Section V and the results are

Manuscript received January 16, 2011; revised January 24, 2011, April 28, 2011, and July 18, 2011; accepted August 18, 2011. Date of publication October 06, 2011; date of current version January 20, 2012. This work was supported by the United Kingdom Engineering and Physical Research Council under grant GR/T28836/01. Paper no. TPWRS-01031-2010.

The authors are with the Department of Electronic and Electrical Engineering, The University of Strathclyde, Glasgow, U.K.

Digital Object Identifier 10.1109/TPWRS.2011.2166091

presented in Section VI. Discussion and critical analysis of the results is provided in Section VII before the conclusions and opportunities for further work are described in Section VIII.

II. Reinforcement Learning

Reinforcement learning is learning from reward by mapping situations to actions when interacting with an uncertain environment [13]. In the classical model of agent-environment interaction, at each time step t in a sequence of discrete time steps $t = 1, 2, 3\ldots$, an agent receives as input some form of the environment's state $s_t \in S$, where S is the set of possible states. From a set of actions $A(s_t)$ available to the agent in state s_t, the agent selects an action a_t and performs it in its environment. The environment enters a new state s_{t+1} in the next time step and the agent receives a scalar numerical reward $r_{t+1} \in \mathbb{R}$ in part as a result of its action. The agent then adjusts its policy for selecting actions by learning from the state representation s_t, the chosen action a_t, and the reinforcement signal r_{t+1} before beginning its next interaction.

Traditional reinforcement learning methods, such as Q-learning [14] or Sarsa [15], attempt to learn an action-value function $Q(s_t, a_t)$ that returns the longterm expected reward $v_t^{s_t,a_t}$ for taking action a_t in state s_t. If a discrete number of states n_s and actions n_a are defined, these values may be stored in a look-up table of the form

$$\begin{array}{c} \\ s^1 \\ s^2 \\ \vdots \\ s^{n_s} \end{array} \begin{array}{c} \begin{array}{cccc} a^1 & a^2 & & a^{n_a} \end{array} \\ \begin{bmatrix} v^{1,1} & v^{1,2} & \cdots & v^{1,n_a} \\ v^{2,1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ v^{n_s,1} & \cdots & \cdots & v^{n_s,n_a} \end{bmatrix} \end{array}. \quad (1)$$

In Q-learning the values are updated after each interaction according to the equation

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (2)$$

where γ is a discount factor, with $0 \leq \gamma \leq 1$ that prevents values from going unbounded and represents reduced trust in the reward r_t as discrete time t increases. The learning rate α, where $0 \leq \alpha \leq 1$, controls how much attention is paid to new data when updating.

A balance between exploration of the environment and exploitation of past experience must be struck when selecting actions. The ϵ-greedy approach to action selection is defined by a randomness parameter ϵ, where $0 \leq \epsilon \leq 1$, and a decay parameter d [16]. A random number x_r where $0 \leq x_r \leq 1$ is drawn for each selection. If $x_r < \epsilon$, then a random action is selected; otherwise, the perceived optimal action is chosen. After each selection, the randomness is attenuated by d.

Enumerating high-dimensional state and action spaces can result in impractical memory requirements in all but the simplest of problems [1]. Function approximation techniques, such as artificial neural networks [17], can be used to approximate the value function and allow these methods to be applied to problems with multidimensional and continuous environments [18]. However, it has been found that updates from greedy action selections can cause methods using this technique to fail to generalize [3]–[5].

Policy gradient reinforcement learning methods learn a *policy function* that returns an action given the current perceived state of the environment [6]. Small incremental changes are made to the parameter vector θ of a policy function approximator (connection weights in the case of an artificial neural network) in the direction of steepest ascent of some policy performance measure Y with respect to the parameters

$$\theta_{t+1} = \theta_t + \alpha \frac{\partial Y}{\partial \theta_t} \quad (3)$$

where α is again a positive definite step size learning rate. Policy gradient methods are differentiated largely by the techniques used to obtain an estimate of the gradient $\partial Y/\partial\theta$. Some of the most successful real-world robotics results [7], [19] have been yielded by likelihood ratio methods such as Williams' REINFORCE [20] and natural policy gradient methods, such as the Episodic Natural Actor-Critic (ENAC) [21]. For a concise overview of these methods, the interested reader is referred to [22].

A relatively simple reinforcement learning method, proposed by Roth and Erev [11], has received considerable attention from the agent based electricity market simulation community. The algorithm is based on empirical results obtained by observing how humans learn decision-making strategies in games against multiple strategic players. It learns a policy in which the state of the environment is ignored and each action a is associated with a single value q that is the agent's propensity for selecting it. After each interaction, the propensity for the previous action, that resulted in the reward r_t, is adjusted by an experimentation parameter ϵ and all other action propensities are adjusted by a small proportion of their current value.

Two shortcomings of the original Roth-Erev algorithm were identified in [23] and a modified formulation was proposed. Under this variant, after selecting action a' in interaction t, the propensity to select action a for interaction $t+1$ is

$$q_a(t+1) = \begin{cases} (1-\phi)q_a(t) + r_t(1-\epsilon), & a = a' \\ (1-\phi)q_a(t) + q_a(t)\left(\frac{\epsilon}{A-1}\right), & a \neq a' \end{cases} \quad (4)$$

where A is the total number of feasible actions and ϕ is the *recency* parameter. The recency, or forgetting parameter, degrades the propensities for all actions and prevents propensity values from going unbounded. It is intended to represent the tendency for players to forget older action choices and to prioritize more recent experience. The experimentation parameter prevents the probability of choosing an action from going to zero and encourages exploration of the action space.

This paper proposes a new stateful variant of the Roth-Erev method. Instead of assuming only one state and maintaining only one row of propensities, a multi-row table with one row per environment state, as with Q-learning, is used. Action propensity values are still updated according to (4), but it is the values in the row associated with the previous state that are modified during the learning step. The method allows differentiation between states, but can increase the number of propensity values that require updating.

III. Related Work

When agent based electricity market simulations first emerged, they were driven by heuristics based on domain expert knowledge and intuition, and were implemented as basic trading rules [24]–[27]. Successive publications from the London Business School illustrate a trend in the field towards more complex algorithms and improved behavioral models [28], [29].

More recently, unsupervised reinforcement learning algorithms from the field of artificial intelligence have been used in energy market research [30]. Variations on the Q-learning technique have been used to study congestion management schemes [31], combined electricity and gas markets [32], and emissions allowance trading [33]. The Roth-Erev method has been used to study market power [23], the Italian wholesale electricity market [34], cross-holdings[1] [35], and interrelationships between contracts markets and balancing markets [36], [37].

Policy gradient reinforcement learning methods have not previously been used in agent based electricity market research, but have been successfully applied in other laboratory [6] and operational settings [7], [10]. In [8] and [9] a recurrent gradient method was used to optimize financial investment performance without price forecasting, and in [38], a modified version of REINFORCE is used to simulate a marketplace for grid computing resources.

IV. Modeling Electricity Trade

In this paper a power exchange auction market model is used to provide an electricity trading environment for comparing reinforcement learning algorithms. It is based on the SmartMarket model that is provided with MATPOWER [39] and was developed for the PowerWeb project at Cornell University. Market participants are modeled using software agents from PyBrain [40] that use reinforcement learning algorithms to adjust their behavior. Their interaction with the market is coordinated in multi-agent simulations, the structure of which is derived from PyBrain's single player design. The multi-agent system consists of discrete and continuous market *environments*, specific *tasks* for agents, and *modules* that are used for policy function approximation and storing state-action values or action propensities.

A. Power Exchange Auction Market

In each trading period, the auction accepts offers to sell blocks of capacity from participating agents. A double-sided auction, in which bids to buy blocks of power may be submitted by agents associated with dispatchable loads, is also available, but this feature is not used. Valid offers for each generator are sorted into non-decreasing order with respect to price and converted into corresponding generator capacities and piecewise linear cost functions. The newly configured units form a unit-decommitment optimal power flow problem [39], the solution to which provides generator set-points and nodal marginal prices that are used to determine the proportion of each offer block that should be cleared and the associated clearing price. The cleared offers determine each agent's revenue and hence the profit that is used as the reward signal.

A nodal marginal pricing scheme is used in which the price of each offer is cleared at the value of the Lagrangian multiplier on the power balance constraint for the bus at which the offer's generator is connected.

B. Market Environments for Agents

Each agent has a portfolio of n_g generators associated with their local environment. Each environment is responsible for 1) returning a vector representation of its current state and 2) accepting an action vector which transforms the environment into a new state.

1) Discrete Market Environment: For agents using the Q-learning or Roth-Erev methods, an environment with n_s discrete states and n_a discrete action possibilities is defined. The environment produces a state s, where $s \in \mathbb{Z}^+$ and $0 \leq s < n_s$, at each simulation step and accepts an action a, where $a \in \mathbb{Z}^+$ and $0 \leq a < n_a$. To prevent state space enumeration from exceeding memory limits, the discrete states are derived only from the current total system demand $d = \sum P_d$, where P_d is the vector of active power demand at each bus. Informally, the state space is n_s states between the minimum and maximum demand and the current state s_t is the index of the state to which the present demand, d, relates.

The action space for a discrete environment is defined by a vector m, where $0 \leq m_i \leq 100$, of percentage markups on marginal cost with length n_m, a vector w, where $0 \leq w_i \leq 100$, of percentage capacity withholds with length n_w and a scalar number of offers n_o, where $n_o \in \mathbb{Z}^+$, to be submitted for each generator associated with the environment. Each offer relates to one price-quantity block, where the price is the marginal cost of the associated generator marked up by m_i percent and the quantity is p_g/n_o, where p_g is the rated output of the associated generator, with w_i percent withheld. Increasing n_o provides greater flexibility with regards to how capacity is sold, allowing some capacity to be offered at a low price, ensuring dispatch, and for the remainder to be marked up further, risking non-dispatch for the chance of greater profit.

2) Continuous Market Environment: For agents operating policy gradient learning algorithms, a continuous environment is defined. It outputs a state vector s, where $s_i \in \mathbb{R}$, and accepts an action vector a, where $a_i \in \mathbb{R}$. Scalar variables m_u and w_u define the upper limit on the percentage markups on marginal cost and the upper limit on the percentage of capacity that can be withheld, respectively. Again, n_o defines the number of offers to be submitted for each generator associated with the environment.

The state vector can be any set of variables from the power system or market model, e.g., bus voltages, branch power flows, generator limit Lagrangian multipliers. Each element of the vector provides one input to the neural network used for policy function approximation.

The action vector a has length $2n_g n_o$. Element a_i, where $0 \leq a_i \leq m_u$, corresponds to the percentage price markup and a_{i+1}, where $0 \leq a_{i+1} \leq w_u$, to the percentage of capacity to withhold for the $(i/2)$th offer from the agent, where $i = 0, 2, 4, \ldots, 2n_g n_o$.

[1]Cross-holdings occur when one publicly traded firm owns stock in another publicly traded firm.

C. Agent Tasks

To allow alternative goals (such a profit maximization or meeting some target level for plant utilization) to be associated with a single type of environment, an agent does not interact directly with the environment, but through a particular *task* [40]. Using some measure of risk adjusted return (as in [8]) might be of interest in the context of simulated electricity trade and this would simply involve the definition of a new task and would not require any modification of the environment.

A task defines the reward returned to the agent and thus defines the agent's purpose. For all simulations in this paper, the goal of each agent is to maximize direct financial profit. Rewards are defined as the sum of earnings from the previous period t, as determined by the difference between the revenue from cleared offers and generator marginal costs at their total cleared quantity.

Given a task, an agent may have a restricted view of the environment or be able to only perform certain actions. Thus a task adjusts the state vector before it is passed to the agent and makes adjustments to the action vector before it is passed to the environment. Agents operating policy-gradient learning methods approximate their policy functions using artificial neural networks that are presented with input vector x of length n_s where $x_i \in \mathbb{R}$. The state vector from the environment may consist of values that differ greatly in their relative magnitude. To ensure that all values have a similar influence on the agent's policy, the task produces a normalized vector, with $-1 \leq x_i \leq 1$, for input to the policy function approximator. To accomplish this, vectors of lower and upper sensor limits are defined, s_l and s_u, respectively, and used to calculate the normalized input vector

$$x = 2\left(\frac{s - s_l}{s_u - s_l}\right) - 1. \tag{5}$$

To produce an output vector y, where $-1 \leq y_i \leq 1$, nodes that implement a hyperbolic tangent activation function are used in the output layer of the artificial neural network. Outputs in this range can be denormalized to provide values for markup and capacity withhold that are valid for the associated generator if vectors of lower and upper action limits, a_l and a_u, respectively, are defined. The output and the limit vectors are combined to produce an action vector

$$a = \left(\frac{y+1}{2}\right)(a_u - a_l) + a_l \tag{6}$$

where $0 \leq a_i \leq m_u$ and $0 \leq a_{i+1} \leq w_u$ for $i = 0, 2, 4, \ldots, 2n_g n_o$.

D. Participant Agents

Each agent is defined as an entity capable of producing an action a based on previous observations of its environment s and is associated with a *module*, a *learner*, a *dataset*, and an *explorer*.

The module is used to determine the agent's policy for action selection and returns an action vector a when activated with observation s. For Q-learning, the module is an $n_s \times n_a$ table where each element $v_t^{s_t,a_t}$ is the value in state s_t associated with selecting action a_t at simulation step t. For a standard Roth-Erev learner, the table has one row that stores the propensity for selection of each action.

The module for policy gradient methods is a multi-layer feed-forward artificial neural network that outputs a vector a when presented with an observation vector s.

The learner can be any reinforcement learning algorithm that modifies the values/propensities/parameters of the module to increase expected future reward. The dataset stores state-action-reward triples for each interaction between the agent and its environment. The stored history is used by learners when computing updates to the module.

Each learner is associated with an explorer that adds a degree of exploration to the agents action selection by returning an explorative action a_e when activated with the current state s and action a from the module. Softmax and ϵ-greedy [13] explorers are implemented for discrete action spaces. Policy gradient methods use an additional module that adds Gaussian noise to the output of the neural network. The explorer has a parameter σ that relates to the standard deviation of the Gaussian distribution.

V. IEEE Reliability Test System Simulation

Learning methods are compared by repeating the same simulation of competitive electricity trade and switching the type of algorithm used by the competing agents. The IEEE Reliability Test System (RTS) provides a reference electric power system model, hourly load profiles for a whole year, and a realistic generation mix with costs. Each agent interacts with the market once for each hour of the simulated year, submitting offers for each of the generators in its portfolio, and this is then repeated for each of the learning algorithms.

The model has 24 bus locations that are connected by 32 transmission lines, four transformers, and two underground cables. The transformers tie together a 230-kV area and a 138-kV area. The original model has 32 generators of nine different types with a total capacity of 3.45 GW. To reduce the size of the discrete action domain, five 12-MW and four 20-MW generators are removed. This is a minor alteration as their combined capacity is only 4.1% of the original total generation capacity and the remaining capacity is still sufficient to meet demand. To further reduce action space sizes, all generators of the same type at the same bus are aggregated into one unit. This may be considered to be the representation of each individual power station in the market, rather than each synchronous machine. The model has loads at 17 locations and the total demand at system peak is 2.85 GW. The connectivity of branches and the location of generators and loads is illustrated in Fig. 1. This model was selected because it is well established in the electrical engineering research community, it captures important aspects of a real transmission system (such as topology and plant ratings), and it is suitably sized for repeated simulation.

Generator marginal costs are quadratic functions, of the form $c(p_i) = ap_i^2 + bp_i + c$ where p_i is the output of generator i. The parameters a, b, and c for each generator type are given in Table I. Generator cost function coefficients were taken from an RTS model by Georgia Tech Power Systems Control and

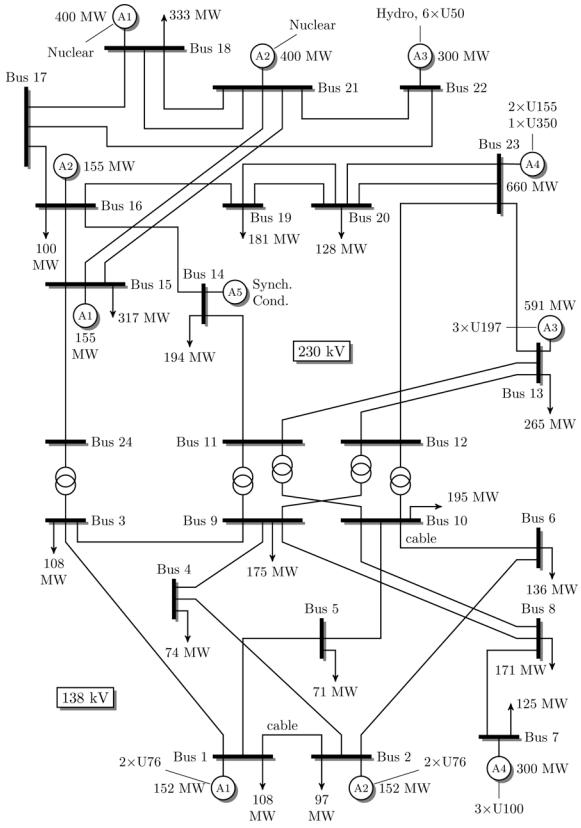

Fig. 1. Single line diagram for the simplified IEEE Reliability Test System.

TABLE I
GENERATOR TYPES AND COST PARAMETERS FOR THE SIMPLIFIED IEEE RELIABILITY TEST SYSTEM

Code	a	b	c	Type
U50	0.0	0.001	0.001	Hydro
U76	0.01414	16.0811	212.308	Coal
U100	0.05267	43.6615	781.521	Oil
U155	0.00834	12.3883	382.239	Coal
U197	0.00717	48.5804	832.758	Oil
U350	0.00490	11.8495	665.109	Coal
U400	0.00021	4.4231	395.375	Nuclear

Automation Laboratory that assumes Coal costs of 1.5 $/MBtu,[2] Oil costs of 5.5 $/MBtu, and Uranium costs of 0.46 $/MBtu.

The generating stock is divided into four portfolios that are each endowed to a learning agent. Table II shows the number of generators of each type in each portfolio. The portfolios were chosen such that each agent has: a mix of base load and peaking plant, approximately the same total generation capacity, generators in different areas of the network, and enough generators to produce a challenging action domain. The generator labels in Fig. 1 denote the associated agent. The synchronous condenser is associated with a passive agent that always offers 0 MW at 0 $/MWh (the unit can be dispatched to provide or absorb reactive power within its limits).

Markups on marginal cost are restricted to a maximum of 30% and discrete markups of 0, 15%, or 30% are defined for value function based methods. Up to 20% of the total capacity

TABLE II
PORTFOLIOS OF GENERATING PLANT ENDOWED TO EACH AGENT

Agent	U50 Hydro	U76 Coal	U100 Oil	U155 Coal	U197 Oil	U350 Coal	U400 Nuclear
1		2×		1×			1×
2		2×		1×			1×
3	6×				3×		
4			3×	2×		1×	

of each generator can be withheld and discrete withholds of 0 or 20% are defined. These values were chosen so as to be similar with those used in [30] to allow the same findings to be reproduced and for existing research to be built upon. Finer discrete action definitions would allow agents to compete more closely to the margins of dispatch, but would greatly increase the size of the action space. However, initially only one offer per generator is required, but this is increased to two in order to explore the effect of increased state space size and offer flexibility.

The environment state for all algorithm tests is derived from a forecast of the total system demand for the next one-hour period. The system demand follows the hourly profile from the RTS which varies according to the day of the week and the time of year. For tests of value function based methods and the Stateful Roth-Erev learning algorithm, the continuous state is divided into three equally sized discrete states between minimum and maximum demand that allow differentiation between low, medium, and peak load.

To investigate exploitation of constraints, AC optimal power flow is used and the state vector for agents using policy gradient methods is optionally enhanced to combine the demand forecast with voltage constraint Lagrangian multipliers of all generator buses and the voltage magnitude at all other buses. Lagrangian multipliers are used as the voltage at generator buses is typically fixed and the multipliers indicate if the constraint is binding. Branch flows are not included in the state vector as the flow limits in the RTS are high and are typically not reached at peak demand. Generator capacity limits are binding in most states of the RTS, but the output of other generators is deemed to be hidden from an agent.

Typical parameter values, either defaults from PyBrain or inspired by the literature, are used for each of the algorithms. Learning rates are set low and exploration parameters decay slowly due to the length and complexity of each simulation. By decaying exploration parameters to a suitably low level, all algorithms converge to a stable policy by the end of the simulated year. No attempt to study parameter sensitivity if undertaken, but Q-learning is typically robust to parameter changes in simulations of this type [31]. For Q-learning, $\alpha = 0.2$, $\gamma = 0.99$, and ϵ-greedy action selection is used with $\epsilon = 0.9$ and $d = 0.999$. For Roth-Erev learning, $\epsilon = 0.55$, $\phi = 0.3$, and Boltzmann action selection [13] is used with $\tau = 100$ and $d = 0.999$.

Two-layer neural networks with linear input and output nodes, no bias nodes, and randomized initial connection weights are used for policy function approximation. The exploration parameter σ for these methods is initialized to zero and adjusted manually after each episode t such that

$$\sigma_t = d(\sigma_{t-1} - \sigma_n) + \sigma_n \tag{7}$$

[2]1 Btu (British thermal unit) ≈ 1055 Joules

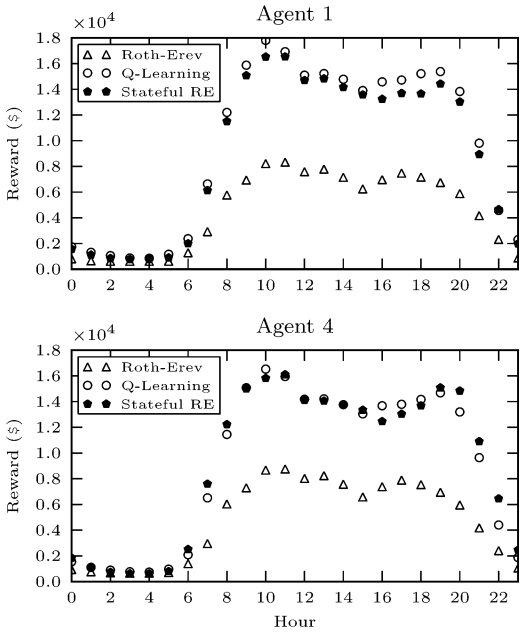

Fig. 2. Modified Roth-Erev compared with Stateful Roth-Erev.

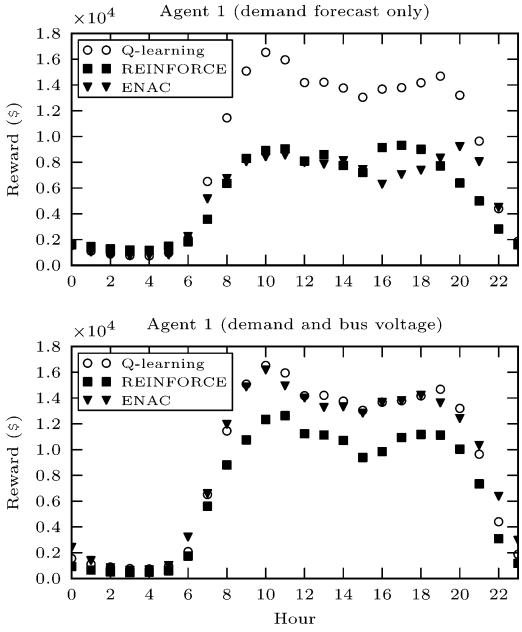

Fig. 3. Average rewards for agent 1 under two state configurations.

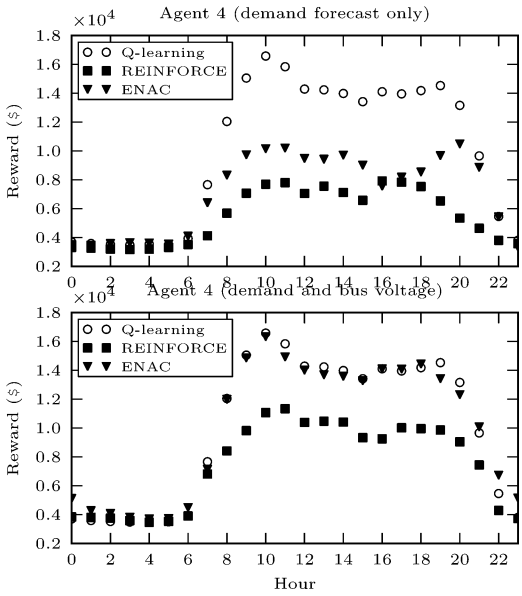

Fig. 4. Average rewards for agent 4 under two state configurations.

where $d = 0.995$ is a decay parameter and $\sigma_n = -0.5$ specifies the value that is converged to asymtotically. Constant learning rates are used in each simulation with $\gamma = 0.01$ for REINFORCE and $\gamma = 0.005$ for ENAC. Hourly trade for a year of 364 days is simulated for each algorithm.

VI. Results

To compare algorithms, the average reward for each hour of the day over one simulated year is calculated for agents 1 and 4 and plotted. Results for only agents 1 and 4 are given as agents 1 and 2 have identical portfolios and agent 3's portfolio consists mostly of Hydro plant with zero cost. When the marginal cost of a generator is zero, regardless of the percentage markup chosen, the offer price is always zero. This is a limitation of marking up prices based on a percentage of marginal cost, rather than by a fixed price value and results in passive market behavior from agent 3 under all algorithms.

Fig. 2 compares the Modified Roth-Erev method [23] with the Stateful Roth-Erev method. The plots show average rewards for agents 1 and 4 when using Q-learning and the two Roth-Erev variants.

Figs. 3 and 4 compare policy gradient methods when submitting one offer per generator under two different state vector configurations. Fig. 3 concerns agent 1 and shows the average reward received for a state vector consisting solely of the demand forecast and for a combined demand forecast and bus voltage profile state vector. Fig. 4 shows average rewards for agent 4 under the same configurations. The discrete environment used by Q-learning does not use the voltage profile data to define its state, but results using purely the system demand are shown in all of the plots in Figs. 3 and 4 for comparison.

Fig. 5 shows average rewards for agents 1 and 4 using Q-learning and ENAC with *two* offers required per generator. The continuous environment used by with the ENAC method is presented with the same enhanced state vector that produced the results in Figs. 3 and 4 under the one offer per generator case.

VII. Discussion

Agents with a discrete environment have 216 possible actions to choose from in each state when required to submit one offer per generator. Fig. 2 shows that, using Q-learning, the agents are able to learn an effective policy that yields increased profits using two different portfolios. The importance of utilizing environment state data in a dynamic electricity setting is illustrated by the differences in average reward received by the modified Roth-Erev method and the Stateful Roth-Erev method. The optimal action for an agent depends upon the current system load and the stateless Roth-Erev formulation is unable to interpret

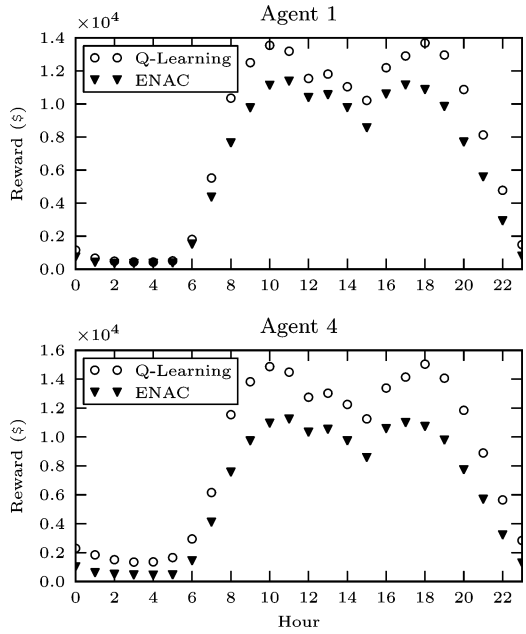

Fig. 5. Average rewards for two offers per generator.

this. The Stateful Roth-Erev method can be seen to achieve approximately the same performance as Q-learning.

Including bus voltage constraint data in the state for a discrete environment would result in a state space of impractical size, but including it in a continuous environment was straightforward. Figs. 3 and 4 show that ENAC achieves greater profits when presented with a combined demand forecast and bus voltage state vector. REINFORCE performs less well than ENAC, but also shows improvement over the pure demand forecast case. ENAC achieves equivalent, but not greater performance than Q-learning in all periods of the trading day when using the voltage data. It is not able to use the additional state information to achieve any advantage over Q-learning, but it does learn a profitable policy.

Changing the number of offers that are required to be submitted for each generator from 1 to 2 increases the number of discrete action possibilities in each state to 46 656. Fig. 5 shows that Q-learning is still able to achieve a similar level of reward as under the one offer case. The profitability for both methods is degraded, but ENAC receives significantly lower average reward when required to produce a larger action vector and is not able to use the increased flexibility in its offer structure to any advantage.

VIII. Conclusion

Policy gradient methods are found to be a valid option for modeling the strategies of electricity market participants. However, in this paper they have been outperformed by a traditional action-value function algorithm in all of the simulations. No evidence has been found to suggest that policy gradient methods can exploit complex constraints in a power system model. However, they have been shown to improve in performance when operating with a richer state vector that includes bus voltage level and voltage constraint information. Some limitations of the standard Roth-Erev method in a dynamic environment have been found and an alternative configuration that rectifies the issues has been demonstrated. Q-learning was able to produce an effective policy in all simulations, including one involving a relatively large action space that saw degraded performance from a policy gradient method.

AC optimal power flow adds enormously to simulation times when analyzing an entire year of hourly trading interactions. The addition of bus voltage data to the state vector improved the performance of the policy gradient methods, but it has not been shown if the same could not be achieved by perhaps using bus voltage angles from a DC optimal power flow.

No study of parameter sensitivity is performed, and alternative function approximation and back-propagation techniques and configurations could also be investigated in the future. Given the performance of the Q-learning method in this paper, further work might also involve extended versions of this method, such as Neuro-Fitted Q-Iteration [41] and GQ(λ) [42], that have been developed for use in continuous multivariate environments.

Acknowledgment

The authors would like to thank the researchers from Cornell University, especially Dr. R. Zimmerman, for their work on the optimal power flow formulations in MATPOWER and for giving the authors permission to translate them into Python. Similarly, the authors would like to thank the researchers from Dalle Molle Institute for Artificial Intelligence (IDSIA) and the Technical University of Munich involved in developing the reinforcement learning algorithms and artificial neural networks that form PyBrain.

References

[1] A. M. L. P. Kaelbling and M. Littman, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[2] G. Tesauro, "TD-Gammon, a self-teaching backgammon program, achieves master-level play," *Neural Computat.*, vol. 6, no. 2, pp. 215–219, 1994.

[3] J. N. Tsitsiklis and B. V. Roy, "Feature-based methods for large scale dynamic programming," in *Proc. Machine Learning*, 1994, pp. 59–94.

[4] G. Gordon, "Stable function approximation in dynamic programming," in *Proc. 12th Int. Conf. Machine Learning*, , 1995, pp. 261–268, Morgan Kaufmann.

[5] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Proc. 12th Int. Conf. Machine Learning*, , 1995, pp. 30–37, Morgan Kaufmann.

[6] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Advances in Neural Information Processing Systems*, 2000, vol. 12, pp. 1057–1063.

[7] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Proc. 2006 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Oct. 2006, pp. 2219–2225.

[8] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 4, pp. 875–889, Jul. 2001.

[9] J. Moody, L. Wu, Y. Liao, and M. Saffell, "Performance functions and reinforcement learning for trading systems and protfolios," *J. Forecast.*, vol. 17, pp. 441–470, 1998.

[10] L. Peshkin and V. Savova, "Reinforcement learning for adaptive routing," in *Proc. 2002 Int. Joint Conf. Neural Networks (IJCNN 2002)*, 2002, vol. 2, pp. 1825–1830.

[11] A. E. Roth, I. Erev, D. Fudenberg, J. Kagel, J. Emilie, and R. X. Xing, "Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term," *Games Econ. Behav.*, vol. 8, no. 1, pp. 164–212, 1995.

[12] Application of Probability Methods Subcommittee, "IEEE reliability test system," *IEEE Trans. Power App. Syst.*, vol. PAS-98, no. 6, pp. 2047–2054, Nov. 1979.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, May 1998.
[14] C. Watkins, "Learning From delayed rewards," Ph.D. dissertation, Univ. Cambridge, Cambridge, U.K., 1989.
[15] G. A. Rummery and M. Niranjan, Online Q-Learning Using Connectionist Systems Cambridge University Engineering Department, Tech. Rep. No. CUED/F-INFENG/TR 166, 1994.
[16] R. L. Rivest and C. E. Leiserson, *Introduction to Algorithms*. New York: McGraw-Hill, 1990.
[17] , L. Fausett, Ed., *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Upper Saddle River, NJ: Prentice-Hall, 1994.
[18] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Proc. Advances in Neural Information Processing Systems*, 1996, vol. 8, pp. 1038–1044.
[19] H. Benbrahim, "Biped dynamic walking using reinforcement learning," Ph.D. dissertation, Univ. New Hampshire, Durham, NH, 1996.
[20] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Proc. Machine Learning*, 1992, pp. 229–256.
[21] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7–9, pp. 1180–1190, 2008.
[22] J. Peters, "Policy gradient methods," *Scholarpedia* vol. 5, no. 11, p. 3698, 2010. [Online]. Available: http://www.scholarpedia.org/article/Policy_gradient_methods.
[23] J. Nicolaisen, V. Petrov, and L. Tesfatsion, "Market power and efficiency in a computational electricity market with discriminatory double-auction pricing," *IEEE Trans. Evol. Computat.*, vol. 5, no. 5, pp. 504–523, Aug. 2002.
[24] P. Visudhiphan and M. Ilic, "Dynamic games-based modeling of electricity markets," in *Proc. IEEE Power Eng. Soc. 1999 Winter Meeting*, Feb. 1999, vol. 1, pp. 274–281.
[25] J. Bower and D. Bunn, "Experimental analysis of the efficiency of uniform-price versus discriminatory auctions in the England and Wales electricity market," *J. Econ. Dynam. Control*, vol. 25, no. 3–4, pp. 561–592, Mar. 2001.
[26] D. Ernst, A. Minoia, and M. Ilic, "Market dynamics driven by the decision-making of both power producers and transmission owners," in *Proc. IEEE Power Eng. Soc. General Meeting, 2004*, Jun. 2004, pp. 255–260.
[27] G. Conzelmann, G. Boyd, V. Koritarov, and T. Veselka, "Multi-agent power market simulation using EMCAS," in *Proc. IEEE Power Eng. Soc. General Meeting*, Jun. 2005, vol. 3, pp. 2829–2834.
[28] J. Bower, D. W. Bunn, and C. Wattendrup, "A model-based analysis of strategic consolidation in the german electricity industry," *Energy Pol.*, vol. 29, no. 12, pp. 987–1005, 2001.
[29] D. W. Bunn and F. S. Oliveira, "Evaluating individual market power in electricity markets via agent-based simulation," *Ann. Oper. Res.*, pp. 57–77, 2003.
[30] T. Krause, E. V. Beck, R. Cherkaoui, A. Germond, G. Andersson, and D. Ernst, "A comparison of Nash equilibria analysis and agent-based modelling for power markets," *Int. J. Elect. Power Energy Syst.*, vol. 28, no. 9, pp. 599–607, 2006.
[31] T. Krause and G. Andersson, "Evaluating congestion management schemes in liberalized electricity markets using an agent-based simulator," in *Proc. IEEE Power Eng. Soc. General Meeting, 2006*, 2006.
[32] F. Kienzle, T. Krause, K. Egli, M. Geidl, and G. Andersson, "Analysis of strategic behaviour in combined electricity and gas markets using agent-based computational economics," in *Proc. 1st Eur. Workshop Energy Market Modelling Using Agent-Based Computational Economics*, Karlsruhe, Germany, Sep. 2007, pp. 121–141.
[33] J. Wang, V. Koritarov, and J.-H. Kim, "An agent-based approach to modeling interactions between emission market and electricity market," in *Proc. IEEE Power & Energy Soc. General Meeting, 2009 (PES 2009)*, Jul. 2009, pp. 1–8.
[34] M. A. Rastegar, E. Guerci, and S. Cincotti, "Agent-based model of the Italian wholesale electricity market," in *Proc. 6th Int. Conf. Eur. Energy Market, 2009*, May 2009, pp. 1–7.
[35] A. R. Micola and D. W. Bunn, "Crossholdings, concentration and information in capacity-constrained sealed bid-offer auctions," *J. Econ. Behav. Organiz.*, vol. 66, no. 3–4, pp. 748–766, 2008.
[36] A. Weidlich and D. Veit, "Bidding in interrelated day-ahead electricity markets—Insights from an agent-based simulation model," in *Proc. 29th IAEE Int. Conf.*, Jul. 2006.
[37] D. Veit, A. Weidlich, J. Yao, and S. Oren, "Simulating the dynamics in two-settlement electricity markets via an agent-based approach," *Int. J. Manage. Sci. Eng. Manage.*, vol. 1, no. 2, pp. 83–97, 2006.
[38] D. Vengerov, "A gradient-based reinforcement learning approach to dynamic pricing in partially-observable environments," *Future Gen. Comput. Syst.*, vol. 24, no. 7, pp. 687–693, 2008.
[39] R. Zimmerman, C. Murillo-Sanchez, and R. Thomas, "MATPOWER: Steady-state operations, planning and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011.
[40] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstie, and J. Schmidhuber, "PyBrain," *J. Mach. Learn. Res.*, vol. 11, pp. 743–746, 2010.
[41] M. Riedmiller, "Neural fitted Q iteration—First experiences with a data efficient neural reinforcement learning method," in *Proc. 16th Eur. Conf. Machine Learning*, 2005, pp. 317–328, Springer.
[42] H. R. Maei and R. S. Sutton, "$GQ(\lambda)$: A general gradient algorithm for temporal-difference prediction learning with eligibility traces," in *Proc. 3rd Conf. Artificial General Intelligence*, Lugano, Switzerland, 2010.

Richard Lincoln received the M.Eng. degree in electrical and mechanical engineering from The University of Edinburgh, Edinburgh, U.K., in 2005 and the Ph.D. degree in electrical and electronic engineering from The University of Strathclyde, Glasgow, U.K., in 2011.

His research interests include automated energy trade and open source software for power system simulation and visualization.

Stuart Galloway is currently a senior lecturer in the Institute for Energy and Environment at the University of Strathclyde, Glasgow, U.K. He was initially appointed as a Rolls-Royce Senior Research Fellow focusing on novel distributed generation control and electricity market trading problems. Prior to this, he undertook doctoral research in applied mathematics at the University of Edinburgh. His current research interests include the application of optimization techniques to power engineering problems, the modeling of novel electrical power systems, and market simulation.

Bruce Stephen (M'09) received the B.Sc. degree from Glasgow University, Glasgow, U.K., and the M.Sc. and Ph.D. degrees from the University of Strathclyde, Glasgow.

He currently holds the post of Senior Research Fellow within the Institute for Energy and Environment at the University of Strathclyde. His research interests include distributed information systems, machine learning applications in power system and animal welfare condition monitoring and asset management.

Graeme Burt (M'95) received the B.Eng. degree in electrical and electronic engineering in 1988 and the Ph.D. degree in 1992 following research into fault diagnostic techniques for power networks, both from the University of Strathclyde, Glasgow, U.K.

He is a Reader at the University of Strathclyde. His research interests include power system modeling and intelligent system applications in power engineering.