University of Strathclyde

Department of Electronic and Electrical Engineering

Learning to Trade Power

by

Richard W. Lincoln

A thesis presented in fulfilment of the
requirements for the degree of

Doctor of Philosophy

2010

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Acknowledgements

The research reported in the present thesis was funded by the United Kingdom Engineering and Physical Research Council under grant GR/T28836/01.

Abstract

Connectionist reinforcement learning methods approximating value-functions offer few convergence guarantees, even in simple systems. Reinforcement learning has been applied previously to agent-based simulation of energy markets using only discrete action and sensor domains. If learning algorithms are to deliver on their potential for application in operational settings, modelling continuous domains is necessary. The contribution of this thesis is to show that policy-gradient reinforcement learning algorithms can be applied to continuous representations of energy trading problems and that their superior use of sensor data results in improved performance over previously applied value-function methods. From this it follows that algorithms which search directly in the policy space will be better suited to decision support applications and automated energy trade.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis describes the application of value-function and policy gradient reinforcement learning algorithms to the electric energy trade problem. This chapter introduces the problem and explains the motivation for the research. The goals of the research are stated along with the principle contributions. Finally, a reading guide is provided that provides an overview of the remaining chapters.

1.1 Motivation

Two trends characterise modern power systems Engineering: Increased liberalisation of the industry through competitive energy trade and increased presence of renewable energy generation on the network. As the number and variety of electricity sources becomes greater, the necessity for automated trade of their power increases. Control algorithms may draw sensor input from data networks and other sources and use some relevant measure of performance, such as profitability, to learn from trading decisions.

Methods including learning classifier systems, genetic algorithms and reinforcement learning have been successfully used to research the charateristics of energy markets in the past(Weidlich & Veit, 2008). This an alternative to the traditional closed-form equilibrium approaches to game theory research in which behavior emerges from the interactions of many separable, self-serving agents. Typically in these studies, an agent is associated with a portfolio of generating units and/or dispatchable loads. Interaction with the environment involves submission of offers to sell or bids to buy[1] a quantity of power at a specified price in a particular time period. The learning algorithms typically use revenue or earnings as a reward signal and adjust the policy used to select offer/bid price and quantity values.

Research into energy trade using reinforcement learning typically involves discretization the action domain, often into incremental markups on marginal cost (Weidlich & Veit, 2008). Also, either sensor domains are discretized or state information is disregarded altogether. Despite these simplifications, authors have been drawn many practical conclusions from this approach[ref].

In the field of robotic control, environment state, action and observation spaces are often continuous or mixed. Traditional reinforcement learning algorithms such as Sarsa and Q-learning can be applied to systems with continuous domains by using connectionist systems for value function approximation(Barto, Sutton, & Anderson, 1983). However, feedback between policy updates and value function changes can result in oscillations or divergence in these algorithms even when applied to simple systems(Peters & Schaal, 2008). In response to this, policy-gradient methods, pioneered by Williams(Williams, 1992), which search directly in the policy space have been developed and applied in many real-life settings(Sutton, Mcallester, Singh, & Mansour, 2000; Peters & Schaal, 2006; Moody & Saffell, 2001; Peshkin & Savova, 2002).

1.2 Problem Statement

Engineers must strive for complexity in their work. Rarely will a simple solution will perform a function to a higher degree than a more complex one. Certainly, where a function is either performed or not performed, prefer the simpler one, but most often problems can be solved to varying degrees.

The broad aim of the research presented in this thesis is to prove that the above conjecture applies to reinforcement learning algorithms for power trade. Previous research in this field (See Chapter 3 below) has used very simple algorithms in relation to those from the latest advances in artificial intelligence (See Sections 2.2.3 and 2.2.3 below). The goal is to prove that policy gradient methods, using artificial neural networks for policy function approximation, are better suited to learning the complex dynamics of a power system.

[1] Beware that certain authors may use the term “bid” to refer to an offer to sell when discussing single-sided auctions.

1.3 Research Contributions

This paper compares policy-gradient reinforcement learning algorithms REINFORCE and ENAC with value-function methods Roth-Erev, Q(λ) and Sarsa in their relative ability to trade electricity competitively. Power systems are modelled as balanced three-phase AC networks in the steady-state. Offers/bids for active and reactive power from agent participants are cleared using AC optimal power flow with an auction interface that returns single period revenue and earnings values. Through individual and multi-player experiments the methods are compared in their ability to learn quickly, compete in large systems and exploit characteristics of the power system. It is shown that, in electricity trade, policy-gradient methods:

- converge successfully on an optimal policy,
- are slower to converge than value-function algorithms,
- can learn more complicated characteristics of the power system than value-function algorithms,
- scale better when applied to larger systems and reactive power markets.

Section 2.1 of this paper presents the power system model, the optimal power flow formulation and the auction interface from MATPOWER. Reinforcement learning methods Sarsa, Q(λ), Roth-Erev, REINFORCE and ENAC are defined in Section 2.2. Section 4 introduces the three experiments used to compare the aforementioned methods. Numerical results from these experiments are reported in Section 5 and an interpretation and critical analysis of them is given in Section 5.4. Finally, a review of related research is presented in Section 3 and Section 7 provides a conclusion.

1.4 Outline

This thesis is focussed on the application of standard and advanced reinforcement learning algorithms to a particular problem domain. The reader will require a certain degree of prior knowledge, or must be willing to read much of the referenced material, to fully understand the methodology taken. The intended audience is engineering and economics researchers interested in the application of reinforcement learning algorithms to the problem of trading energy in electric power systems.

Chapter 2

Background

This chapter provides an introduction to optimal power flow and reinforcement learning. Different formulations of the optimal power flow problem are explained along with the principles behind interior-point methods, common used to find their solutions. Reinforcement learning is also a generic term and Section 2.2 introduces computational approaches to learning to maximise long term reward. Definitions are provided for the valued-based and direct search algorithms compared in subsequent chapters.

2.1 Optimal Power Flow

Optimal power flow is a term used to describe a broad class of problems in which an objective function is optimised while constraints on the optimisation variables, which represent physical characterisitcs of the electric power system, are satisfied. Optimisation techniques typically solve problems of the form

$$\min_{x} f(x) \tag{2.1}$$

subject to

$$\begin{aligned} g(x) &= 0 && (2.2)\\ h(x) &\leq 0 && (2.3)\\ x_{min} \leq x &\leq x_{max} && (2.4) \end{aligned}$$

2.1.1 Interior-Point Methods

2.2 Reinforcement Learning

This section describes agent's policies that represent a store of experience and the learning algorithms that its modify parameters. Together these components form models of individual behavior which are used to determine the actions to be performed in the agent's environment and to learn from received rewards.

For a comprehensive introduction to reinforcement learning with evaluations of algorithm designs through mathematical analysis and computational experiments the intersted reader is directed to the seminal work by Barto and Sutton

2.2.1 Introduction

The problem of learning how best to interact with an environment so as to maximise some long-term reward is one that arises in many aspect of life. Reinforcement learning is a term that is typically applied to understanding, automating and solving this problem through computational approaches. Unlike with the majority of Machine Learning techinques, the algorithms are not instructed as to which actions to take, but must learn to maximise the long-term reward through trial-and-error.

Reinforcement learning starts with an interactive, goal-seeking individual and an associated environment. The individuals require the ability to sense aspects of their environment, perform actions that influence the state of their environment and be assigned rewards as a response to their chosen action. An agent is said to follow a particular *policy* when mapping the perceived state of its environment to an action choice.

Value-based methods attempt to find the optimal policy by approximating a *value-function* which returns the total reward an agent can expect to accumulate, given an initial state and following the current policy thereafter.

Policy-gradient methods are an alternative to this which represent a policy using a learned function approximator with its own parameters The function approximator is updated according to the gradient of expected reward with respect to these parameters.

2.2.2 Value Function Methods

Basic Roth-Erev

The Roth-Erev reinforcement learning algorithm uses a stateless policy to select actions from a discrete domain(Roth et al., 1995; Erev & Roth, 1998). The dataset stored by each agent, j, contains an array of length K, where K is the number of feasible actions k. Each value in the array represents the propensity for selection of the associated action in all states of the environment. Following interaction t in which agent j performed on the environment action k', for arbitrary positive t, a reward, $r_{jk'}(t)$, is calculated. The propensity for agent j to select action k for interaction $t+1$ is

$$q_{jk}(t+1) = \begin{cases} (1-\phi)q_{ik}(t) + r_{jk'}(t)(1-\epsilon), & k = k' \\ (1-\phi)q_{ik}(t) + r_{jk'}(t)(\frac{\epsilon}{K-1}), & k \neq k' \end{cases} \tag{2.5}$$

where ϕ and ϵ denote *recency* and *experimentation* parameters, respectively. The recency (forgetting) parameter degrades the propensity for all actions and prevents the value from going unbounded. It is intended to represent the tendency for players to forget older action choices and to prioritise more recent experience. The experimentation parameter prevents the probability of choosing an action from going to zero and thus encourages exploration of the action space.

Erev and Roth proposed that actions be selected according to a discrete probability distribution function where action k is selected for interaction $t+1$ with probability:

$$p_{jk}(t+1) = \frac{q_{jk}(t+1)}{\sum_{l=0}^{K} q_{jl}(t+1)} \tag{2.6}$$

Since $\sum_{l=0}^{K} q_{jl}(t+1)$ increases with t, a reward $r_{jk}(t)$ for performing action k will have a greater effect on the probability $p_{jk}(t+1)$ during early interactions while t is small. This is intended to represent Psychology's *Power Law of Practice* in which it is qualitatively stated that, with practice, learning occurs at a decaying expoential rate and that a learning curve will eventually flatten out.

This algorithm may alternatively use a form of the *softmax* method (Sutton & Barto, 1998) using the Gibbs, or Boltzmann, distribution to select action k for the

$t + 1$th interaction with probability

$$p_{jk}(t+1) = \frac{e^{q_{jk}(t+1)/\tau}}{\sum_{l=0}^{K} e^{q_{jl}(t+1)/\tau}} \tag{2.7}$$

where τ is the *temperature* parameter. This parameter may be decreased in value over the course of an experiment since high values give all actions similar probability and encourage exploration of the action space, while low values promote exploitation of past experience.

Variant Roth-Erev

Two shortcomings of the basic Roth-Erev algorithm (§2.2.2) have been identified and a variant formulation proposed(Nicolaisen, Petrov, & Tesfatsion, 2002). The problems are that the values by which propensities are updated can be zero or very small for certain combinations of the experimentation parameter ϵ and the total number of feasible actions K. Also, all propensity values are decreased by the same amount when the reward, $r_{jk'}(t)$ is zero. Under the variant algorithm the propensity of agent j to select action k for interaction $t + 1$ becomes:

$$q_{jk}(t+1) = \begin{cases} (1-\phi)q_{ik}(t) + r_{jk'}(t)(1-\epsilon), & k = k' \\ (1-\phi)q_{ik}(t) + q_{jk}(t)(\frac{\epsilon}{K-1}), & k \neq k' \end{cases} \tag{2.8}$$

As with the basic Roth-Erev algorithm, the propensity for the action that the reward is associated with is adjusted by the experimentation parameter. All other action propensities are adjusted by a small proportion of their current value.

SARSA

The SARSA algorithm is an on-policy Temporal Difference control method, similar to Q-learning. The action-value update for agent j is defined by

$$Q_j(s_{jt}, a_{jt}) + \alpha[r_{jt+1} + \gamma Q_j(s_{jt+1}, a_{jt+1}) - Q_j(s_{jt}, a_{jt})]. \tag{2.9}$$

While the Q-learning algorithm updates action-values using a greedy policy, which is a different policy to that being followed, SARSA uses the discounted future reward of the next state-action observation following the original policy.

Q-Learning

The formulation of the Q-learning algorithm used is that of the original off-policy Temporal Difference algorithm developed by Watkins(Watkins, 1989). The action-value function, $Q(s, a)$, returns values from a $M \times N$ matrix where M and N are arbitrary positive numbers equal to the total number of feasible states and actions, respectively. Each value represents the *quality* of taking a particular action, a, in state s. Actions are selected using either the ϵ-greedy or softmax (See section 2.2.2) methods. The ϵ-greedy method either selects the action (or one of the actions) with the highest estimated value or it selects an action at random, uniformly, independently of the estimated values with, typically small, probability ϵ.

Agent j will observe a reward, r_{jt}, and a new state, s_{jt+1}, after taking action a_{jt} at step t when in state s_{jt}. The state-action value, $Q_j(s_{jt}, a_{jt})$, is updated according to the maximum value of available actions in state s_{t+1} and becomes

$$Q_j(s_{jt}, a_{jt}) + \alpha[r_{jt+1} + \gamma \max_a Q_j(s_{jt+1}, a_{jt}) - Q_j(s_{jt}, a_{jt})] \tag{2.10}$$

where α and γ are the learning rate, $0 \leq \alpha \leq 1$, and discount factor, $0 \leq \gamma \leq 1$, respectively. The learning rate determines the extent to which new rewards will override the effect of older rewards. The discount factor allows the balance between maximising immediate rewards and future rewards to be set.

Q(λ)

With the Q-learning formulation, described in equation 2.10, only the quality associated with the previous state, s_{jt}, is updated. However, the preceding states can also, in general, be said to be associated with the reward r_{jt+1}. Eligibility traces are a mechanism for facilitating this effect and in algorithms such as Q(λ), the λ refers to it. The eligibility trace for a state $e(s)$ represents how eligible the state s is to receive credit or blame for the error. The term “trace” refers to fact that only recently visited states become eligible. The eligibility value for the current state is increased, while for all other states it is attenuated by a factor λ.

The off-policy nature of Q-learning requires special care to be taken when implementing eligibility traces. While the algorithm may learn a greedy policy, in which the action with the maximum value would always be taken, typically a policy with some degree of exploration will be followed when choosing actions. If an exploratory (pseudo-random) step is taken the preceding states can no longer be considered eligible for credit or blame. Setting λ to 0 for non-greedy actions removes much of the benefit of using eligibility traces if exploratory actions are frequent. A solution to this has been developed, but requires a very complex implementation (Peng & Williams, 1996). A naïve approach can be taken, where the effect of exploratory actions is ignored, but the results of this are unexplored.

2.2.3 Policy Gradient Methods

REINFORCE

The previously defined learning methods typically rely upon discretisation of the sensor and action spaces so the associated values may be stored in tables. The memory requirements for this restrict the application of these methods to only small environments. Many environments, particularly from real applications, exhibit continuous sensor and/or action spaces and require generalisation techniques to be employed to provide a more compact policy representation.

REINFORCE is an associative reinforcement learning algorithm that determines a policy by modifying the parameters of a policy function approximator, rather than approximating a value function (Williams, 1992). Commonly, feedforward artificial neural networks are used to represent the policy, where the input is a representation of the state and the output is action selection probabilities. In learning, a *policy gradient* approach is taken where the weights of the network are adjusted in the direction of the gradient of expected reinforcement.

Defining the network, let $\mathbf{x}^i$ denote the vector of inputs to the ith unit and y_i denote output of the unit. In the input layer of the network the elements x_j of $\mathbf{x}^i$ are normalised sensor values from the environment and in the output layer, or in any hidden layers, they are outputs from the j unit in the preceding layer. Let $\mathbf{w}^i$ denote the vector of the weights, w_{ij}, on the connections to the ith unit. The output of the ith unit is dependant on the vector of inputs, $\mathbf{x}^i$, and the associated weights, $\mathbf{w}^i$.

For each interaction of the agent with the environment, each parameter w_{ij} is incremented by

$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})\frac{\partial \ln \rho_i}{\partial w_{ij}} \tag{2.11}$$

where α_{ij} is the *learning factor*, b_{ij} is the *reinforcement baseline* and ρ_i is the performance of the policy (e.g., the average reward per interaction).

ENAC

ToDo: Episodic Natural Actor Critic(Peters & Schaal, 2008).

Chapter 3

Related Work

3.1 Agent-Based Simulation

Relative to the traditional closed-form equilibrium approaches, agent-based simulation of (electricity) markets is a new field of research. For comprehensive reviews and surveys of the many different techniques that have been applied in recent years the interested reader is directed to (Weidlich & Veit, 2008; Tesfatsion & Judd, 2006; Visudhiphan, 2003). This section will focus on reviewing literature from the field in which reinforcement learning techniques were applied in combination with explicit power system models. A short review is also provided of some more general applications of reinforcement learning with connectionist systems and policy-gradient methods.

3.1.1 Learning Method Comparisons

3.1.2 Heuristic Approaches

3.1.3 Simulations Applying Genetic Algorithms

3.1.4 Learning Classifier Systems

3.1.5 Simulations Applying Q-learning

Krause et al. have published agent-based energy market research in which Q-learning methods were applied while considering physical system properties. In a comparison between Nash equilibrium analysis and agent-based simulation, the suitability of bottom-up modelling for the assessment of market evolution was assessed(Krause et al., 2004). This is built upon in subsequent publications which evaluate the influence on market power and social welfare of three congestion management schemes and which analyse strategic behavior in combined gas and electricity markets(Krause & Andersson, 2006; Kienzle, Krause, Egli, Geidl, & Andersson, 2007). Power Transmission Distribution Factors (PTDF) are used in place of explicit power flow equations in determining line flows. The action domain of generating agents is limited to 0%, 5% and 10% markups on true marginal costs. The implementation of the Q-learning method used does not differentiate between environment states when selecting actions. This is a modification to the traditional formulation that still results in convincing conclusions, as with the popular Roth-Erev method.

There are similar applications of Q-learning in which states *are* defined, but none model the AC transmission system. A common approach is to use categorised market price from the previous period as state information(Bakirtzis & Tellidou, 2006; Xiong, Okuma, & Fujita, 2004).

3.1.6 Simulations Applying Roth-Erev

The AMES (Agent-based Modeling of Electricity Systems) power market test bed is a software package that models core features of the Wholesale Power Market Platform (WPMP) – a market design proposed by the US Federal Energy Regulatory Commission (FERC) in April 2003 for common adoption in regions of the US(Sun & Tesfatsion, 2007a). The design features:

- a centralised structure managed by an independent market operator,
- parallel day-ahead and real-time markets and
- locational marginal pricing.

Learning agents may represent load serving entities or generating companies and learn using implementations of the Roth-Erev method (See sections 2.2.2 and 2.2.2, above) built on the Repast agent simulation toolkit(Gieseler, 2005). The permissive license under which the source code for these algorithms has been released allowed direct translation of them for use in this study. Agents learn from the solutions of hourly bid/offer based DC-OPF problems formulated as quadratic programs and solved using QuadProgJ(Sun & Tesfatsion, 2007b).

The ability of generator agents to learn particular supply offers has been demonstrated along with the plotting and data handling capabilities of AMES using a 5-bus network model(Li & Tesfatsion, 2009). The same network has been used to investigate strategic capacity withholding in FERC wholesale power market design(Hongyan & Tesfatsion, 2009). Generator agents report linear marginal cost functions to the market operator and supply functions are formed through linear interpolation between the prices at minimum and maximum production limits. Load serving agents submit combinations of fixed demand bids and price-sensitive bid functions, the ratio between which is varied between 0.0 and 1.0 to test physical and economic capacity withholding potential. Comparing results from a benchmark case (in which true production costs are reported, but higher than marginal cost functions may be reported) and cases in which reported production limits may be less than the true values, the authors find, that with sufficient capacity reserve, no evidence to suggest potential for inducing higher net earnings through capacity withholding in the WPMP.

3.2 Closed-Form Equilibrium

3.3 Policy-Gradient Reinforcement Learning

3.4 Open Source Power Engineering Software

Chapter 4

Methodology

The present chapter concerns the approach taken in comparing methods for learning to trade power. Transmission system constraints are accounted for and the power system model used is defined. Generator costs are added to this model and used to form an optimal power flow problem. An auction interface to the optimal power flow is used to provide a representation of a realistic electricity market. This market is then combined with a multi-agent system to create a simultion platform for competitive energy trade. Finally, three experiments are defined to test different aspects of the methods learning abilities.

4.1 Electricity Market Model

Computation of the generator dispatch points is executed using parts of the of the optimal power flow formulation from MATPOWER (R. Zimmerman, Murillo-Sánchez, & Thomas, 2009). In order that the optimal power flow routine could be coupled with agents from the machine learning library PyBrain, the MATLAB$^{\text{TM}}$ source code from MATPOWER was translated to the Python programming language. With the permission of the MATPOWER developers the resulting package has been released under the terms of version 2 of the GNU General Public License as a project named PYLON (Lincoln, Galloway, & Burt, 2009). Sparse matrix objects from the convex optimisation library CVXOPT were used to allow the implementation to scale well to solving for very large systems.

This section describes parts of the optimal power flow formulation, unit-decommitment algorithm and auction interface from MATPOWER that were used to represent a centralised electricity market. Notable components of the full optimal power flow forumlation (available in (R. D. Zimmerman & Murillo-Sánchez, 2007)) that have been ignored are shunt capacitors and inductors, generator P-Q capability curves and dispatchable loads. The power system model is described by defining the bus, branch and generator objects. The power flow equations associated with a network of these components are subsequently defined. The constrained cost variable approach to modelling generator cost functions from (Wang, Murillo-Sanchez, Zimmerman, & Thomas, 2007) is introduced, from which the optimal power flow formulation follows.

The experiments described in Section 4 require an optimal power flow problem to be solved at each step. To accelerate the simulation process for certain experiments the option to use a linearised DC formulation is used, the formulation of which is provided also. The tradeoffs between using DC models over AC have been examined in (Overbye, Cheng, & Sun, 2004) and found reasonable for locational marginal price calculations.

Since the optimal power flow formulations do not facilitate shutting down expensive generators, the unit-decommitment algorithm from MATPOWER is defined. Finally, to provide an interface to agent participants that resembles that of real electricity market, MATPOWER's auction wrapper for the optimal power flow routine is described.

4.1.1 Power System Model

The power system is assumed to be a three-phase AC system operating in the steady-state and under balanced conditions in which it may be represented by a single phase network of busbars connected by branch objects.

Branches

Each branch is modelled as a medium length transmission line in series with a transformer at the *from* end. A nominal-π model with total series admittance $y_s = 1/(r_s + jx_s)$ and total shunt capacitance b_c is used to represent the transmission line. The transformer is assumed to be ideal and both phase-shifting and tap-changing, with the ratio between primary winding voltage v_f and secondary winding voltage $N = \tau e^{j\theta_{ph}}$ where τ is the tap ratio and θ_{ph} is the phase shift angle.

From Kirchhoff's current law the current in the series impedance is

$$i_s = \frac{b_c}{2} v_t - i_t \tag{4.1}$$

and from Kirchhoff's voltage law the voltage across the secondary winding of the transformer is

$$\frac{v_f}{N} = v_t + \frac{i_s}{y_s} \tag{4.2}$$

Substituting i_s from (4.1), gives

$$\frac{v_f}{N} = v_t - \frac{i_t}{y_s} + v_t \frac{b_c}{2y_s} \tag{4.3}$$

and rearranging in terms if i_t, gives

$$i_t = v_s \left(\frac{-y_s}{\tau e^{\theta_{ph}}} \right) + v_r \left(y_s + \frac{b_c}{2} \right) \tag{4.4}$$

The current through the secondary winding of the transformer is

$$N^* i_f = i_s + \frac{b_c}{2} \frac{v_f}{N} \tag{4.5}$$

Substituting i_s from (4.1), gives

$$N^* i_f = \frac{b_c}{2} v_t - i_t + \frac{b_c}{2} \frac{v_f}{N} \tag{4.6}$$

Substituting $\frac{v_f}{N}$ from (4.3) and rearranging, gives

$$i_s = v_s \left(\frac{1}{\tau^2} \left(y_s + \frac{b_c}{2} \right) \right) + v_r \left(\frac{y_s}{\tau e^{-j\theta}} \right) \tag{4.7}$$

Generators

Each generator i is modelled as an apparent power source $s_g^i = p_g^i + j q_g^i$ at a specific bus k, where p_g^i is the active power injection and q_g^i the reactive power injection, each expressed in per-unit to the system base MVA. Upper and lower limits on p_g^i are specified by p_{max}^i and p_{min}^i, respectively, where $p_{max}^i > p_{min}^i \geq 0$. Similarly, upper and lower limits on q_g^i are specified by q_{max}^i and q_{min}^i, respectively, where $q_{max}^i > q_{min}^i$.

Buses and Loads

At each bus k, constant active power demand is specified by p_d^k and reactive power demand by q_d^k. Upper and lower limits on the voltage magnitude at the bus are defined by $v_m^{k,max}$ and $v_m^{k,min}$, respectively. For one bus with an associated generator, designated the *reference* bus, the voltage angle is θ_k^{ref} and typically valued zero. Dispatchable loads are modelled as generators with negative p_g^i, where $p_{min}^i < p_{max}^i = 0$.

4.1.2 AC Power Flow Equations

For a network of n_b buses, n_l branches and n_g generators, let Cg be the $n_b \times n_g$ bus-generator connection matrix such that the $(i,j)^{th}$ element of C_g is 1 if generator j is connected to bus i. The $n_b \times 1$ vector of complex power injections from generators at all buses is

$$S_{g,bus} = C_g \cdot S_g \tag{4.8}$$

where $S_g = P_g + jQ_g$ is the $n_g \times 1$ vector with the i^{th} element equal to s_g^i.

Combining (4.7) and (4.4), the *from* and *to* end complex current injections for branch l are

$$\begin{bmatrix} i_f^l \\ i_t^l \end{bmatrix} = \begin{bmatrix} y_{ff}^l & y_{ft}^l \\ y_{tf}^l & y_{tt}^l \end{bmatrix} \begin{bmatrix} v_f^l \\ v_t^l \end{bmatrix} \tag{4.9}$$

where

$$\begin{aligned} y_{ff}^l &= \frac{1}{\tau^2}\left(y_s + \frac{b_c}{2}\right) && (4.10) \\ y_{ft}^l &= \frac{y_s}{\tau e^{-j\theta_{ph}}} && (4.11) \\ y_{tf}^l &= \frac{-y_s}{\tau e^{j\theta_{ph}}} && (4.12) \\ y_{tt}^l &= y_s + \frac{b_c}{2} && (4.13) \end{aligned}$$

Let Y_{ff}, Y_{ft}, Y_{tf} and Y_{tt} be $n_l \times 1$ vectors where the l-th element of each corresponds to y_{ff}^l, y_{ft}^l, y_{tf}^l and y_{tt}^l, respectively. Furthermore, let C_f and C_t be the $n_l \times n_b$ branch-bus connection matrices, where $C_{f_{i,j}} = 1$ and $C_{t_{i,k}} = 1$ if branch i connects *from* bus j *to* bus k. The $n_l \times n_b$ branch admittance matrices are

$$Y_f = \mathbf{diag}(Y_{ff})C_f + \mathbf{diag}(Y_{ft})C_t \tag{4.14}$$

$$Y_t = \mathbf{diag}(Y_{tf})C_f + \mathbf{diag}(Y_{tt})C_t \tag{4.15}$$

and relate the complex bus voltages V to the branch *from* and *to* end current vectors

$$I_f = Y_f V \tag{4.16}$$

$$I_t = Y_t V \tag{4.17}$$

The $n_b \times n_b$ bus admittance matrix is

$$Y_{bus} = C_f^{\mathsf{T}} Y_f + C_t^{\mathsf{T}} \tag{4.18}$$

and it relates the complex bus voltages to the nodal current injections

$$I_{bus} = Y_{bus} V \tag{4.19}$$

The complex power losses from all branches are expressed as a non-linear function of V

$$\begin{aligned} S_{bus}(V) &= \mathbf{diag}(V) I_{bus}^* \\ &= \mathbf{diag}(V) Y_{bus}^* V^* \end{aligned} \tag{4.20}$$

The complex power injections at the *from* and *to* ends of all branches are also expressed as a non-linear functions of V

$$\begin{aligned} S_f(V) &= \mathbf{diag}(C_f V) I_f^* \\ &= \mathbf{diag}(C_f V) Y_f^* V^* \end{aligned} \tag{4.21}$$

$$\begin{aligned} S_t(V) &= \mathbf{diag}(C_t V) I_t^* \\ &= \mathbf{diag}(C_t V) Y_t^* V^* \end{aligned} \tag{4.22}$$

4.1.3 DC Power Flow Equations

The same power system model is used in the formulation of the linearised DC power flow equations, but the following additional assumptions are made:

- The resistance r_s and shunt capacitance b_c of all branch can be considered negligible.

$$y_s \approx \frac{1}{jx_s}, b_c \approx 0 \tag{4.23}$$

- Bus voltage magnitudes $v_{m,i}$, are all approximately 1 per-unit.

$$v_i \approx 1e^{j\theta_i} \tag{4.24}$$

- The voltage angle difference between bus i and bus j is small enough that

$$\sin\theta_{ij} \approx \theta_{ij} \tag{4.25}$$

Applying the assumption that branches are lossless from (4.23), the quadrants of the branch admittance matrix, (4.10), (4.11), (4.12) and (4.13), approximate to

$$y_{ff}^l = \frac{1}{jx_s\tau^2} \tag{4.26}$$

$$y_{ft}^l = \frac{-1}{jx_s\tau e^{-j\theta_{ph}}} \tag{4.27}$$

$$y_{tf}^l = \frac{-1}{jx_s\tau e^{j\theta_{ph}}} \tag{4.28}$$

$$y_{tt}^l = \frac{1}{jx_s} \tag{4.29}$$

Applying the uniform bus voltage magnitude assumption from 4.24 to (4.9), the branch *from* end current approximates to

$$i_f \approx \frac{e^{j\theta_f}}{jx_s\tau^2} - \frac{e^{j\theta_t}}{jx_s\tau e^{-j\theta_{ph}}} \tag{4.30}$$

$$= \frac{1}{jx_s\tau}(\frac{1}{\tau}e^{j\theta_f} - e^{j(\theta_t+\theta_{ph})}) \tag{4.31}$$

The branch *from* end complex power flow $s_f = v_f i_f^*$ therefore approximates to

$$s_f \approx e^{j\theta_f} \cdot \frac{j}{x_s\tau}(\frac{1}{\tau}e^{-j\theta_f} - e^{j(\theta_t+\theta_{ph})}) \tag{4.32}$$

$$= \frac{1}{x_s\tau}\left[\sin(\theta_f - \theta_t - \theta_{ph}) + j\left(\frac{1}{\tau} - \cos(\theta_f - \theta_t - \theta_{ph})\right)\right] \tag{4.33}$$

Applying the voltage angle difference assumption from 4.25 yields the approximation

$$p_f \approx \frac{1}{x_s \tau}(\theta_f - \theta_t - \theta_{ph}) \tag{4.34}$$

Let B_{ff} and $P_{f,ph}$ be the $n_l \times 1$ vectors where $B_{ff_i} = \frac{1}{x_s^i \tau^i}$ and $P_{f,ph_i} = \frac{-\theta_{ph}^i}{x_s^i \tau^i}$. If the system B matrices are

$$\begin{aligned} B_f &= \mathbf{diag}(Bff)(C_f - C_t) & (4.35) \\ B_{bus} &= (C_f - C_t)^{\mathsf{T}} B_f & (4.36) \end{aligned}$$

then the real power bus injections are

$$P_{bus}(\Theta) = B_{bus}\Theta + P_{bus,ph} \tag{4.37}$$

where Θ is the $n_b \times 1$ vector of bus voltage angles and

$$P_{bus,ph} = (C_f - C_t)^{\mathsf{T}} + P_{f,ph} \tag{4.38}$$

The active power flows at the branch *from* ends are

$$P_f(\Theta) = B_f\Theta + P_{f,ph} \tag{4.39}$$

and $P_t = -P_f$ since branches are assumed to be lossless.

4.1.4 AC OPF Formulation

Generator active and, optionally, reactive power output costs are defined by convex n-segment piecewise linear cost functions.

$$c^{(i)}(x) = m_i p + c_i \tag{4.40}$$

for $p_i \leq p \leq p_{i+1}$, $i = 1, 2, \ldots n$ where $m_{i+1} \geq m_i$ and $p_{i+1} > p_i$. Since these costs are non-differentiable the constrained cost variable approach from (Wang et al., 2007) is used to make the optimisation problem smooth. For each generator i a helper cost variable y_i added to the objective function. Inequality constraints

$$y_i \geq m_{i,j}(p - p_j) + c_j, \quad j = 1 \ldots n \tag{4.41}$$

require y_i to lie on the epigraph of $c^{(i)}(x)$. The objective of the optimal power flow problem is to minimise the sum of the cost variables for all generators.

$$\min_{\theta,V_m,P_g,Q_g,y} \sum_{i=1}^{n_g} y_i \tag{4.42}$$

Equality constraints enforce the balance between generator complex power injections S_g and the sum of apparent power demand S_d and the branch losses expressed in (4.20).

$$S_{bus}(V) + S_d - S_g = 0 \tag{4.43}$$

Branch complex power flow limits S_{max} are enforced by the inequality constraints

$$|S_f(V)| - S_{max} \leq 0 \tag{4.44}$$

$$|S_f(V)| - S_{max} \leq 0 \tag{4.45}$$

The reference bus voltage angle θ_i is fixed by the equality constraint

$$\theta_i^{ref} \leq \theta_i \leq \theta_i^{ref}, \quad i \in \mathcal{I}_{ref} \tag{4.46}$$

Upper and lower limits on the optimisation variables V_m, P_g and Q_g are enforced by the inequality constraints

$$v_m^{i,min} \leq v_m^i \leq v_m^{i,max}, \qquad i = 1 \ldots n_b \tag{4.47}$$

$$p_g^{i,min} \leq p_g^i \leq p_q^{i,max}, \qquad i = 1 \ldots n_g \tag{4.48}$$

$$q_g^{i,min} \leq q_g^i \leq q_q^{i,max}, \qquad i = 1 \ldots n_g \tag{4.49}$$

4.1.5 DC OPF Formulation

Piecewise linear cost functions are also used to define generator active power costs in the DC optimal power flow formulation. Since the power flow equations are linearised, following assumptions (4.23), (4.24) and (4.25), the optimal power flow problem simplifies to a linear program. The voltage magnitude variables V_m and generator reactive power set-point variable Q_g are eliminated following assumption (4.25) since branch reactive power flows depend on bus voltage angle differences.

The objective function reduces to

$$\min_{\theta,P_g,y} \sum_{i=1}^{n_g} y_i \tag{4.50}$$

Combining the nodal real power injections, expressed as a function of Θ, from (4.37) with active power generation P_g and active demand P_d, the power balance constraint is

$$B_{bus}\Theta + P_{bus,ph} + P_d - C_g P_g = 0 \tag{4.51}$$

Limits on branch active power flows $B_f\Theta$ and $B_t\Theta$ are enforced by inequality constraints

$$B_f\Theta + P_{f,ph} - F_{max} \;\leq\; 0 \tag{4.52}$$

$$-B_f\Theta + P_{f,ph} - F_{max} \;\leq\; 0 \tag{4.53}$$

The reference bus voltage angle equality constraint from (4.46) and the p_g limit constraint from (4.48) are applied also.

4.1.6 OPF Solution

4.1.7 Unit Decommitment

In the OPF formulation above (See section 2.1) the solver must attempt to dispatch generators within their minimum and maximum power limits. Expensive generators can not be completely shutdown even if doing so would result in a lower total system cost. To achieve this an implementation of the *unit decommitment* algorithm (See Algorithm 1, below) from MATPOWER was used(R. D. Zimmerman & Murillo-Sánchez, 2007, p. 20). The algorithm finds the least cost dispatch by solving repeated OPF problems with different combinations of generating units at their minimum active power limit deactivated. The lowest cost solution is returned when no further improvement can be made and no candidate generators remain.

Algorithm 1 Unit decommitment

```
initialise $N \leftarrow 0$
solve initial OPF
$L_{tot} \leftarrow$ total load capacity
while total min gen. capacity $> L_{tot}$ do
    $N \leftarrow N + 1$
end while
repeat
    for c in candidates do
        solve OPF
    end for
until done = True
```

4.1.8 Auction Interface

Solving the optimisation problem defined above (See section 2.1) is intended to represent the function of a pool market operator. To present agents participating in this market with a simpler interface, more representative of a pool market an implementation of the "smart market" auction clearing mechanism from MATPOWER was used(R. D. Zimmerman & Murillo-Sánchez, 2007, p. 31). Using this interface the OPF problem is formulated from a list of offers to sell and bids to buy power.

An offer/bid specifies a quantity of power in MW and a price for that power in $/MWh, to be traded over a particular period of time. The market accepts sets of offers and bids and uses the solution of the unit decommitment algorithm to return sets of *cleared* offer and bids. The cleared offers/bids can then be used to produce dispatch orders from which values of revenue and earnings/losses may be determined.

The market features the ability to set maximum offer price limits and minimum bids price limits. The process of clearing the market begins by withholding offers/bids outwith these limits, along with those specifying non-positive quantities. Valid offers/bids for each generator are then sorted into non-decreasing/increasing order and used to form new piecewise-linear cost functions and adjust the generator's active power limits.

The dispatch points and nodal prices from solving a unit decommitment OPF with the newly configured generators as input are used in the auction clearing mechanism to determine the proportion of each offer/bid block that should be cleared and

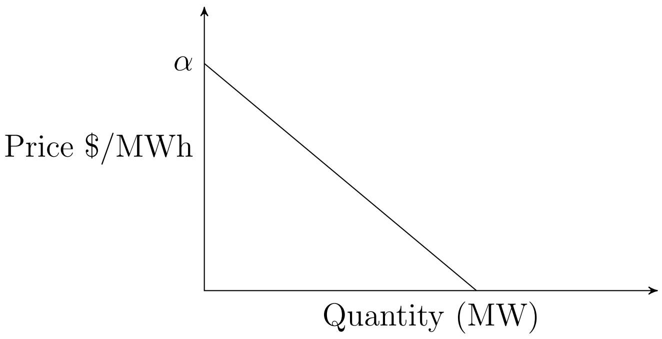

Figure 4.1: Acceptable price range

the associated price for each.

Different pricing options arise from the fact that a gap in which any price is acceptable to all participants may exist between the last accepted offer price and the last accepted bid price (See Figure X). This allows, for example, the prevention of bids setting the price, even when they are marginal, by selecting the *last accepted offer* auction type.

4.2 Multi-Agent System

This section describes the implementation of agents and the coordination of their interactions in multi-agent systems. A generic market environment, with which agents interact regardless of the learning method employed, is defined along with tasks that associate a purpose with an environment. The design of connectionist systems and tables, used to represent agent policies, are given and the process by which they are modified by the agent's learning algorithm is explained. Finally, the collection of agents and tasks into a multi-agent system and the sequence of interactions is illustrated.

4.2.1 Agent, Task & Environment

Environment

Each generator/dispatchable load in the power system model (See Section 4.1.1, above) is associated with an agent[1] via the agent's environment. Each environ-

[1] Management of a portfolio of generators is also supported by the architecture used, but this feature has not been exploited.

ment maintains an association with a singular market instance for submission of offers/bids. Two main operations are supported by an agent's environment.

For a power system with n_b buses, n_l and n_g generators, the `getSensors` method returns a $n_s \times 1$ vector of sensor values s_e^i for generator i where $n_s = 2n_b + 2n_l + 3n_g$. s_g^i represents the visible state of the environment for the agent associated with generator i. s_e^i is composed of sensor values for all buses, branches and generators.

$$s_{e,l}^i = \begin{bmatrix} P_f \\ Q_f \\ P_t \\ Q_t \\ \mu_{S_f} \\ \mu_{S_t} \end{bmatrix}, \quad s_{e,b}^i = \begin{bmatrix} V_m \\ V_a \\ \lambda_P \\ \lambda_Q \\ \mu_{v_{min}} \\ \mu_{v_{max}} \end{bmatrix}, \quad s_{e,g}^i = \begin{bmatrix} P_g \\ \mu_{p_{min}} \\ \mu_{p_{max}} \\ \mu_{q_{min}} \\ \mu_{q_{max}} \end{bmatrix} \quad s_e^i = \begin{bmatrix} s_{e,b}^i \\ s_{e,b}^i \\ s_{e,g}^i \end{bmatrix} \tag{4.54}$$

Not all values are used by the agent and the filtration is done according to the agent's task.

The `performAction` method takes $n_a \times 1$ vector of action values a_e if $s_{bid} = 0$, otherwise a $2n_a \times 1$ vector. If $s_{bid} = 0$, the i-th element of a_e is the offered/bid price in \$/MWh, where $i = 1, 2, \ldots n_{in}$. If $s_{bid} = 1$, the j-th element of a_e is the offered/bid price in \$/MWh, where $j = 1, 3, 5, \ldots n_{in} - 1$ and the k-th element of a_e is the offered/bid quantity in MW where $j = 2, 4, 6, \ldots n_{in}$. The action vector is separated into offers/bids and submitted to the market. If $s_{bid} = 0$, then $qty = p_{max}/n_{in}$.

Task

An agent does not interact directly with it's environment, but is associated with a particular task. A task associates a purpose with an environment and defines what constitutes a reward. Regardless of the learning method employed, the goal of an agent participant is to make a financial profit and the rewards are thus defined as the sum of earnings from the previous period t as calcualted by the market. Sensor data from the environment is filtered according to the task being performed. Agents using the value-function methods under test have a tabular representation of their policy with one row per environment state. Thus, observations consist of a single integer value s_v, where $s_v \leq n_s$ and $s_v \in \mathbb{Z}^+$. Agents using the policy-gradient methods under test have policy functions represented by connectionist systems that use an input vector w_i of arbitrary length where the i-th element $\in \mathbb{R}$. Before input to the connectionist policy function approximator, sensor values are scaled to be between -1 and 1. Outputs from the policy are denormalised using action limits before the action is performed on the environment.

Agent

Agent i is defined as an entity capable of producing an action a_i based on previous observations of its environment s_i, where a_i and s_i are vectors of arbitrary length. As illustrated in Figure X, each agent is associated with a *module*, a *learner* and a *dataset*. The module represents the agent's policy for action selection and returns an action vector a_m when activated with observation s_t. The value-function methods under test use modules which represent a $N \times M$ table, where N is the total number of states and M is the total number of actions.

$$\begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,m} \\ v_{2,1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ v_{n,1} & \cdots & \cdots & v_{n,m} \end{bmatrix} \tag{4.55}$$

Whereas for the policy gradient methods, the module is a connectionist network of other modules as illustrated in Figure X. The learner can use any reinforcement learning algorithm and modifies the values/parameters of the policy module to increase expected future reward. The dataset stores state-action-reward tuples for each interaction between the agent and its environment. The stored history is used by value-function learners when computing updates to the policy values. Policy gradient learners search directly in the space of the policy network parameters.

Value-function learners have an association with an explorer module which returns an explorative action a_e when activated with the current state s_t and action a_m from the policy module. For example, the ϵ-greedy explorer has a randomness parameter ϵ and a decay parameter d. When the ϵ-greedy explorer is activated, a random number x_r is drawn where $0 \leq x_r \leq 1$. If $x_r < \epsilon$ then a random vector of the same length as a_e is returned, otherwise $a_e = a_m$.

4.2.2 Simulation Event Sequence

In each simulation of a system consisting of one or more task-agent pairs a sequence of interactions is coordinated, as illustrated in Figure X.

At the beginning of each step/period the market is initialised and all offers/bids removed. From each task-agent tuple (T, A) an observation s_t is retrieved from T and integrated into agent A. When an action is requested from A its module is activated with s_t and the action a_e is returned. a_e is performed on the environment of A via its associated task T. Recall, this process involves the submission of offer/bids to the market. Once all actions have been performed the offer/bids are cleared using the auction mechanism. Each task T is requested to return a reinforcement reward r_t. All cleared offers/bids associated with the generator in the environment of T are retrieved from the market and r_t is computed from the difference between revenue and cost values.

$$r_t = \text{revenue} - (c_{fixed} + c_{variable}) \tag{4.56}$$

The reward r_t is given to agent A and the value is stored under a new sample is the dataset, along with the last observation s_t and the last action performed a_e. Each agent is instructed to learn from its actions using r_t, at which point the values/parameters of the module of A are updated according to the algorithm of the learner.

This constitutes one step of the simulation and the process is repeated until the specified number of steps are complete. Unless agents are reset, the complete history of states, actions and received rewards is stored in the dataset of each agent.

4.3 Experiment I: Basic Learning

To the best of the author's knowledge, this thesis presents the first case of policy gradient reinforcement learning methods being applied to electricity trading problems. It must first be proven that these methods are capable of learning a basic power trading policy. This section describes the method used to compare methods in their ability to do so.

4.3.1 Aims & Objectives

The purpose of this first experiment is to compare the relative abilities of value-function and policy gradient methods in learning a basic policy for trading power. The objective of the exercise is to examine:

- Speed of convergence to an optimal policy,
- Magnitude and variance of profit and,
- Sensitivity to algorithm parameter changes.

4.3.2 Method of Simulation

Each learning method is tested individually using a range of parameter configurations. A power system model with one bus, one generator k and one dispatchable load l, as illustrated in Figure X is used. In this context, the market clearing process is equivalent to creating offer and bids stacks and finding the point of intersection. A passive agent is associated with the dispatchable load. This agent bids for $-p_{g,l}^{min}$ at marginal cost each period regardless of environment state or reward signal. A dispathcable load is used instead of a constant load to allow a price to be set. Generator k is given sufficient capacity to supply the demand of the dispatchable load, $p_{g,k}^{max} > -p_{g,l}^{min}$, and the marginal of the k is half that of the load l. The generator and dispatchable load attributes are given in Table X. A price cap for the market is set to twice the marginal cost of the l at full capacity, $p_{g,l}^{min}$. The DC optimal power flow formulation (See Section 2.1, above) is used to clear the market and reactive power trade is omitted. The Python code used to conduct the simulations is provided in Listing X.

4.4 Experiment II: Competitive Trade

4.5 Experiment III: Constraint Exploitation

Chapter 5

Results

5.1 Experiment I: Basic Learning

5.2 Experiment II: Competitive Trade

5.3 Experiment III: Constraint Exploitation

5.4 Discussion

5.5 Critical Analysis

Chapter 6

Further work

6.1 AC Optimal Power Flow

6.2 Decentralised Trade

6.3 Standarisation

6.4 Blackbox optimisation

Chapter 7

Summary conclusions

Bibliography

Bakirtzis, A., & Tellidou, A. (2006, November). Agent-based simulation of power markets under uniform and pay-as-bid pricing rules using reinforcement learning. In *Power systems conference and exposition, 2006. psce '06. 2006 ieee pes* (p. 1168-1173). 12

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-13*, 834-846. 2

Erev, I., & Roth, A. E. (1998). Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *The American Economic Review*, *88*(4), 848–881. 6

Gieseler, C. (2005). *A Java reinforcement learning module for the Repast toolkit: Facilitating study and implementation with reinforcement learning in social science multi-agent simulations.* Unpublished master's thesis, Department of Computer Science, Iowa State University. 12

Hongyan, L., & Tesfatsion, L. (2009, March). Capacity withholding in restructured wholesale power markets: An agent-based test bed study. In *Power systems conference and exposition, 2009* (p. 1-11). 13

Kienzle, F., Krause, T., Egli, K., Geidl, M., & Andersson, G. (2007, September). Analysis of strategic behaviour in combined electricity and gas markets using agent-based computational economics. In *1st european workshop on energy market modelling using agent-based computational economics* (p. 121-141). Karlsruhe, Germany. 12

Krause, T., & Andersson, G. (2006). Evaluating congestion management schemes in liberalized electricity markets using an agent-based simulator. In *Power engineering society general meeting, 2006. ieee.* 12

Krause, T., Andersson, G., Ernst, D., Beck, E., Cherkaoui, R., & Germond, A.

(2004). Nash Equilibria and Reinforcement Learning for Active Decision Maker Modelling in Power Markets. In *Proceedings of 6th IAEE European Conference 2004, modelling in energy economics and policy.* 12

Li, H., & Tesfatsion, L. (2009, July). The ames wholesale power market test bed: A computational laboratory for research, teaching, and training. In *Ieee proceedings, power and energy society general meeting.* Alberta, Canada. 13

Lincoln, R., Galloway, S., & Burt, G. (2009, May). Open source, agent-based energy market simulation with Python. In *Energy market, 2009. eem 2009. 6th international conference on the european* (p. 1-5). 14

Moody, J., & Saffell, M. (2001, July). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, *12*(4), 875-889. 2

Nicolaisen, J., Petrov, V., & Tesfatsion, L. (2002, August). Market power and efficiency in a computational electricity market with discriminatory double-auction pricing. *Evolutionary Computation, IEEE Transactions on*, *5*(5), 504–523. Available from `http://dx.doi.org/10.1109/4235.956714` 7

Overbye, T., Cheng, X., & Sun, Y. (2004, Jan.). A comparison of the ac and dc power flow models for lmp calculations. In *System sciences, 2004. proceedings of the 37th annual hawaii international conference on* (p. 9-). 15

Peng, J., & Williams, R. J. (1996). Incremental multi-step q-learning. In *Machine learning* (pp. 226–232). Morgan Kaufmann. 9

Peshkin, L., & Savova, V. (2002). Reinforcement learning for adaptive routing. In *Neural networks, 2002. ijcnn '02. proceedings of the 2002 international joint conference on* (Vol. 2, p. 1825-1830). 2

Peters, J., & Schaal, S. (2006, October). Policy gradient methods for robotics. In *Intelligent robots and systems, 2006 ieee/rsj international conference on* (p. 2219-2225). 2

Peters, J., & Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, *71*(7-9), 1180–1190. 2, 10

Roth, A. E., Erev, I., Fudenberg, D., Kagel, J., Emilie, J., & Xing, R. X. (1995). Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. *Games and Economic Behavior*, *8*(1), 164-212. 6

Sun, J., & Tesfatsion, L. (2007a). Dynamic testing of wholesale power market designs: An open-source agent-based framework. *Computational Economics*, *30*(3), 291–327. 12

Sun, J., & Tesfatsion, L. (2007b, June). Open-source software for power industry research, teaching, and training: A dc-opf illustration. In *Power engineering society general meeting, 2007. ieee* (p. 1-6). 12

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction (adaptive computation and machine learning)*. Mit Press. Gebundene Ausgabe. 6

Sutton, R. S., Mcallester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *In advances in neural information processing systems 12* (Vol. 12, pp. 1057–1063). 2

Tesfatsion, L., & Judd, K. L. (2006). *Handbook of computational economics, volume 2: Agent-based computational economics (handbook of computational economics)*. Amsterdam, The Netherlands: North-Holland Publishing Co. 11

Visudhiphan, P. (2003). *An agent-based approach to modeling electricity spot markets*. Unpublished doctoral dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. 11

Wang, H., Murillo-Sanchez, C., Zimmerman, R., & Thomas, R. (2007, Aug.). On computational issues of market-based optimal power flow. *Power Systems, IEEE Transactions on*, *22*(3), 1185-1193. 15, 20

Watkins, C. (1989). *Learning from delayed rewards*. Unpublished doctoral dissertation, University of Cambridge, England. 8

Weidlich, A., & Veit, D. (2008, July). A critical survey of agent-based wholesale electricity market models. *Energy Economics*, *30*(4), 1728–1759. Available from `http://dx.doi.org/10.1016/j.eneco.2008.01.003` 1, 2, 11

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine learning* (pp. 229–256). 2, 9

Xiong, G., Okuma, S., & Fujita, H. (2004). Multi-agent based experiments on uniform price and pay-as-bid electricity auction markets. In *Electric utility deregulation, restructuring and power technologies, 2004. (drpt 2004). proceedings of the 2004 ieee international conference on* (Vol. 1, pp. 72–76). 12

Zimmerman, R., Murillo-Sánchez, C., & Thomas, R. J. (2009, July). Matpower's extensible optimal power flow architecture. In *Ieee pes general meeting*. Calgary, Alberta, Canada. 14

Zimmerman, R. D., & Murillo-Sánchez, C. E. (2007, September). Matpower: A matlab$^{\text{TM}}$power system simulation package (Version 3.2 ed.) [Computer software manual]. School of Electrical Engineering, Cornell University, Ithaca, NY

14853. 15, 22, 23