2008

A comparative study of Roth-Erev and modied Roth-Erev reinforcement learning algorithms for uniform-price double auctions

Mridul Pentapalli
Iowa State University

A comparative study of Roth-Erev and modified Roth-Erev reinforcement learning algorithms for uniform-price double auctions

by

Mridul Pentapalli

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Iowa State University

Ames, Iowa

2008

DEDICATION

I would like to dedicate this thesis to my parents, my teachers and most of all, Usha.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank Dr. Leigh Tesfatsion, without whom this work would never have been possible. Her dedication and enthusiasm are an inspiration to me. I would also like to thank my committee members, Dr. Giora Slutzki and Dr. Vasant Honavar for their help and support.

I would like to thank all my professors at ISU. A huge thanks to my friends Vikram, Srikanth and Narasimha for lending me their computers for running simulations, to Sreedhar and Staci for their help in proof-reading, and to Pradipta Chatterjee who put the idea in my head that started my MS adventure. Thanks to Milan Thakar for helping me with my recommendations, and to Tom Barbalet for recommendations, books and orcs.

Thanks to Richard Rockhold and everyone at Mediafour for their support.

A special thanks to Linda Dutton and Deb Vance for always being there when needed.

ABSTRACT

This work focuses on multi-agent learning in market contexts. It reports findings from a comparative study of three reinforcement learning algorithms currently in use for a variety of market applications. Two double-auction market testbeds are developed and used to carry out benchmark comparisons involving intensive parameter sweeps with heat map visualization of parameter sensitivities. A primary concern is the degree to which each tested algorithm permits learning agents to converge to the choice of a best action measured in terms of accumulated profits. Some findings from a mathematical analysis of the algorithms' properties are also reported.

The three reinforcement learning algorithms studied in this work are: the Roth-Erev algorithm proposed by Erev and Roth [1], the Modified Roth-Erev (MRE) reinforcement learning algorithm proposed by Nicolaisen et al [2] and the Variant Roth-Erev (VRE) learning algorithm proposed by Sun and Tesfatsion [3].

CHAPTER 1. OVERVIEW

The current literature about multi-agent reinforcement learning (RL) is relatively scarce because of the inherent complexity involved in understanding a dynamic system with many agents learning and interacting simultaneously. Mathematical analysis too is not available in many areas. The inherent difficulty of analytical study of learning algorithms in a multi-agent setting makes computational experiments a much better choice of understanding their behavior.

This work is a step in that direction. We consider a very well known reinforcement learning algorithm developed by Roth and Erev [1] on the basis of human-subject experiments, together with two variations of this algorithm currently being used for the study of wholesale power markets. The behavior of these algorithms in a multi-agent context has not been completely understood and by systematically and exhaustively testing these algorithms over their entire parameter space, we determine the performance of these algorithms and we attempt to explain their behavior. The algorithms that we have considered for this work are widely used in wholesale power market simulations and we hope that this work allows others to understand the parameter settings in order to achieve the best performance for the learning agents.

A wholesale power market is in essence a single commodity double auction and we compare the performance of the RL algorithms in two testbeds based on the clearing house double auction. Double auctions are a good choice as testbeds, since they have been studied widely, especially from the perspective of learning agents and the overall market efficiency [4]. The first testbed is a simple double auction model that simulates a single-commodity clearing house double auction. For the second testbed, we use the computational platform ("AMES") currently being used to study the performance of restructured wholesale power markets [2].

Using the two testbeds, we define five increasingly complex experimental test cases, starting from a very simple setup involving a single learning agent to a highly complex setup with five competing reinforcement learning agents. To improve the visualization and to make the results readily available to the readers, we have introduced the concept of using "heat maps" to display the information generated by the experiments.

This work is immediately relevant to the Agent-based Computational Economics (ACE) group in the Economics department at Iowa State University [1] and the PowerACE group based in Germany[2]. The ACE group can utilize the results from this work in specifying the parameter values in their future work.

This work attempts to lay down a foundation for analysis of computational experiments to understand the behavior of RL algorithms. Presently, researchers are studying algorithms that incorporate evolutionary techniques and reinforcement learning. One of the candidate RL algorithms that can be considered is NEAT+Q proposed by Whiteson and Stone [5]. This sophisticated evolutionary-learning algorithm based on NEAT [6] attempts to combine an artificial neural network based genetic algorithm with reinforcement learning to merge the best features of both types of learning in a single algorithm. However, a deeper understanding of RL performance under alternative parameter specifications would seem to be an important prerequisite. We intend to investigate the possibility of comparing NEAT+Q and other RL algorithms in future studies.

The tested RL algorithms are listed in chapter 2. In chapter 3, the five test cases based on the testbeds are described. Chapter 4 shows the results from the experiments. In chapter 5, some simple mathematical analysis is done on the algorithms. The appendices provide greater information on the software used in this work, and the architecture and implementation of the testbeds used.

[1]The ACE homepage is available at *http://www.econ.iastate.edu/tesfatsi/ace.htm*

[2]The PowerACE group homepage is available at *http://www.powerace.de/engl/index.html*

1.1 History

The Roth-Erev Algorithm was proposed by Alvin Roth and Ido Erev [1]. The authors conducted a series of human experiments and proposed this algorithm to replicate the behavior of their test subjects. This algorithm is considered as one of the prime algorithms to model human behavior and is one of the important algorithms developed in this manner. An interesting difference between the Roth-Erev algorithm and the specifications of reinforcement learning given in Sutton and Barto [7] is that there is no explicit value function in the Roth-Erev algorithm.

The MRE RL algorithm was proposed by Nicolaisen, Petrov and Tesfatsion [2] in response to some of the perceived drawbacks of the Roth-Erev algorithm. The authors state that the original Roth-Erev algorithm fails to learn if the action taken has zero payoff. In the context of a double-auction electricity market, the authors show that market efficiency is substantially higher when the learning traders use the MRE RL algorithm instead of the default Roth-Erev RL algorithm.

The VRE RL algorithm modifies the MRE RL algorithm to ensure that action choice probabilities are properly calculated as nonnegative values even when rewards (profit earnings) can take on negative values. In the MRE RL algorithm, the probabilities are defined to be proportional to the propensity values and in the case of the VRE RL algorithm, the probabilities are calculated based on the exponents of the propensities for each action. This algorithm is currently used as the default learning algorithm in the AMES market framework [3], [8] and [9].

CHAPTER 2. TESTED LEARNING ALGORITHMS

2.1 Roth-Erev Reinforcement Learning Algorithm

The Roth-Erev RL algorithm specifies an equal initial propensity value $q(0)$ for each of the N actions of the learning agent at time $t = 0$. The action propensities are then updated based on the action selected at each subsequent time $t = 1, 2, \ldots$.

If the action a_k is selected at time t, then the propensities for the actions at time $t+1$ are defined as:

$$\begin{aligned} q_j(t+1) &= (1-r) \times q_j(t) + \pi_k(t) \times (1-e) \text{ if } j = k && (2.1) \\ q_j(t+1) &= (1-r) \times q_j(t) + \pi_{\mathbf{k}}(\mathbf{t}) \times \frac{e}{N-1} \text{ if } j \neq k && (2.2) \end{aligned}$$

The *choice probability* of each action at time t, $p_j(t)$ i.e. the probability of choosing the action a_j at time t, is defined as:

$$p_j(t) = \frac{q_j(t)}{\sum\limits_{i=0}^{N-1} q_i(t)} \tag{2.3}$$

where

- $q_j(t)$ is the propensity of action j at time t
- $q_j(0)$ is the propensity of action j at time $t = 0$, also known as initial propensity.
- $\pi_k(t)$ is the reward obtained for action k at time t
- r is the recency parameter
- e is the experimentation parameter

- N is the total number of actions
- a_k is the k^{th} action among the set of N possible actions for the learning agent.

2.2 Modified Roth-Erev Reinforcement Learning Algorithm

The MRE RL algorithm initially assigns an equal initial propensity value $q(0)$ to each of the N actions of the learning agent, similar to the Roth-Erev RL algorithm. However, the action propensities at time t are calculated in a slightly different way. If the action a_k is chosen at time t, the action propensities at time $t + 1$ are calculated to be

$$\begin{aligned} q_j(t+1) &= (1-r) \times q_j(t) + \pi_k(t) \times (1-e) \text{ if } j = k && (2.4) \\ q_j(t+1) &= (1-r) \times q_j(t) + \mathbf{q_j(t)} \times \frac{e}{N-1} \text{ if } j \neq k && (2.5) \end{aligned}$$

The *choice probability* of each action at time t, $p_j(t)$ is given by

$$p_j(t) = \frac{q_j(t)}{\sum\limits_{i=0}^{N-1} q_i(t)} \qquad (2.6)$$

2.3 Variant Roth-Erev Reinforcement Learning Algorithm

The VRE RL algorithm is a variant of the MRE RL algorithm. The setup of the initial propensities and the propensity update functions are unchanged from the MRE RL algorithm. Therefore, the action propensities at time $t + 1$ based on the action a_k chosen at time t are given by:

$$\begin{aligned} q_j(t+1) &= (1-r) \times q_j(t) + \pi_k(t) \times (1-e) \text{ if } j = k && (2.7) \\ q_j(t+1) &= (1-r) \times q_j(t) + \mathbf{q_j(t)} \times \frac{e}{N-1} \text{ if } j \neq k && (2.8) \end{aligned}$$

The *choice probability* of each action at time t, $p_j(t)$ is calculated using a Gibbs-Boltzmann distribution given by:

$$p_j(t) = \frac{e^{\frac{q_j(t)}{T}}}{\sum_{i=0}^{N-1} e^{\frac{q_i(t)}{T}}} \tag{2.9}$$

where

- T is the Boltzmann cooling parameter

Conceptually, the parameters used in these three RL algorithms can be further motivated as follows:

- Experimentation e: The two components of on-line learning 'exploration' and 'exploitation' have to be balanced correctly in order to obtain maximum payoffs. Over-exploration and over-exploitation lead to non-optimal solutions [7], [10]. The experimentation parameter controls the degree of exploration. By increasing the experimentation parameter value, the propensity of the actions not chosen at time t is increased by some factor ('spillover'). This increases the probability of these actions to be chosen at time $t + 1$. This also prevents 'premature fixation' on some action, where the agents fixates on a single action at a very early stage and fails to test any other action.

- Recency r: The recency parameter allows an effect called 'forgetting' [1]. Due to this effect, the learning agent ignores the past effect of learning by a factor given by the recency parameter. Increasing the recency parameter increases this 'forgetting' effect. In a dynamic system where the rewards for an action vary over time, the recency parameter should be large to allow a greater forgetting and negate the effect of actions taken over a longer period. This allows the agent to ignore the rewards obtained in past actions, because they might have become sub-optimal over time.

- Initial Propensity $q_j(0)$: The initial propensity for each action j indicates the learning agent's reward expectation from taking this action. Typically, if the initial propensity is too high as compared to the expected rewards from the agent's actions, then the agent will cycle through a lot of actions. If the initial propensity is too low, then there is a likelihood of early fixation in which the agent will fixate on some non-optimal action at a very early stage and reduce the probability of choosing the remaining actions to an infinitesimal value.

- Boltzmann cooling parameter: The Boltzmann cooling parameter is used to help control the degree to which differences in action propensities are emphasized in the transition from action choice propensities to action choice probabilities. If the variations in the action propensities are not too large, a judicious selection of the cooling parameter allows the differences between the propensities to be amplified in the probability calculations.

CHAPTER 3. FIVE TEST CASE STUDIES

3.1 SimpleModel-I: One Learning Agent with One Profitable Action

3.1.1 Model Design

The structural environment consists of a simple single-commodity double auction with a fixed number of buyers and sellers. There are a total of twelve agents, six sellers and six buyers. There is one seller with a reinforcement learning component and four possible actions. We denote this agent as the 'learning seller'. There are five fixed sellers that sell at a fixed ask price. All the buyers are passive buyers that buy at a fixed bid price. The market is run for one thousand *trading periods*. At the end of each trading period, based on the bid and ask prices of the traders, the market computes the supply-demand curve and the clearing price. In a given trading period, if the trader is infra-marginal, he gets a positive profit. The actions of the learning seller are fixed such that one of the actions makes the agent infra-marginal and the remaining actions make him extra-marginal. Due to this setup, there is only one action that generates a fixed positive reward and the rest of the actions generate zero rewards.

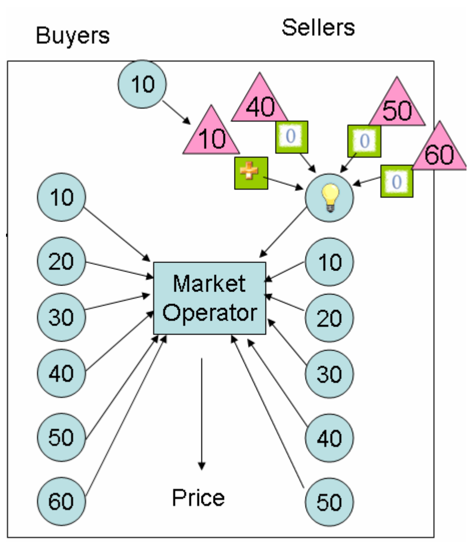


Figure 3.1 SimpleModel-I: One Learning Agent with One Profitable Action

3.1.2 Calculating the uniform clearing price

The demand curve is calculated by first randomly sorting the bid price/quantities given by the buyer agents. The price/quantities are then sorted in decreasing order of price.

Similarly the supply curve is calculated by first randomly sorting the ask price/quantities given by the seller agents. The price/quantities are then sorted in increasing order of price.

The intersection point between the two curves above is calculated. This is the uniform clearing price. All agents to the left of the uniform clearing price point are known as infra-marginal and obtain a profit given by the difference between the uniform price and the reservation value of the buyer or seller.

If there exist multiple intersection points with different prices, the average of the prices is considered as the uniform clearing price.

If there exist multiple intersection points with different quantities, the point with the largest quantity is considered as the intersection point, indicating that at the point where the ask and bid prices are the same, the transaction is considered to have taken place.

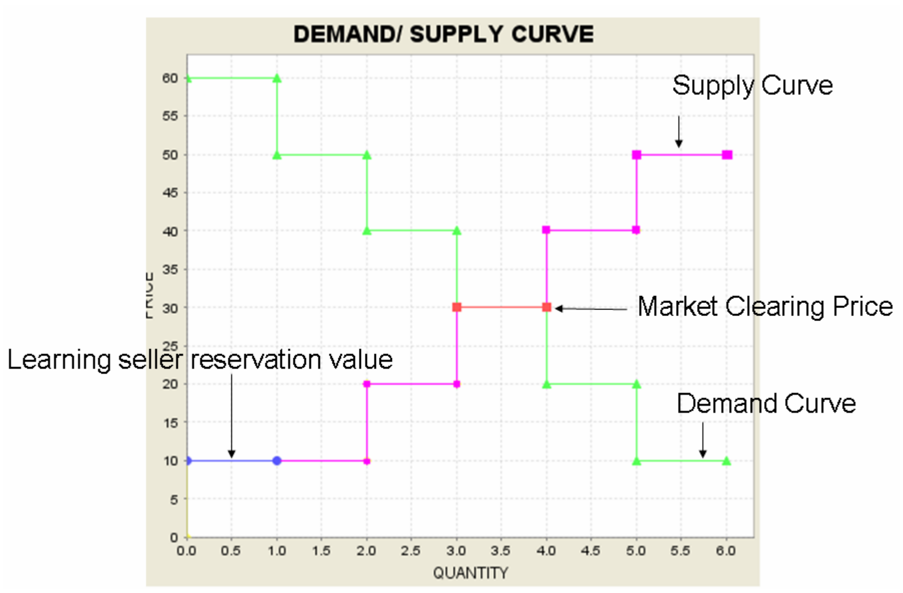


Figure 3.2 Demand/Supply Curve for SimpleModel-I with true reservation values

3.1.3 Experimental Design

The treatment factors that primarily affect how well the learning agent learns are the experimentation and the recency. Since the learning algorithm determines the actions based on the probabilities that are updated in every iteration, a certain number of simulations are run using unique random seeds. This ensures that the stochastic effects of the algorithm are minimized.

We vary the experimentation and recency parameters from 0.0 to 1.0 with an interval of 0.1 and with a 100 randomly generated random seeds. This gives rise to a total to 11 $\times$ 11 $\times$ 100 simulations. Each simulation was run for 1000 time periods. One time period corresonds to a single trading period. For the SimpleModel experiments, the simulations are run for two settings of initial propensity, viz. 1.0 and 1000.0.

Two sets of experiments are conducted based on the above parameter settings. In one set of experiments, the learning agent is a Roth-Erev RL agent and in the second set of experiments, the learning agent is a MRE RL agent.

3.2 SimpleModel-II: One Learning Agent with Two Profitable Actions

3.2.1 Model Design

The model setup is very similar to the one used in the above scenario. The only change is that the learning agent now has two actions that make the agent infra-marginal in the market and therefore both actions generate a fixed positive profit. Though both actions generate nearly equal profits, one of the actions always generates a greater profit than the other action.

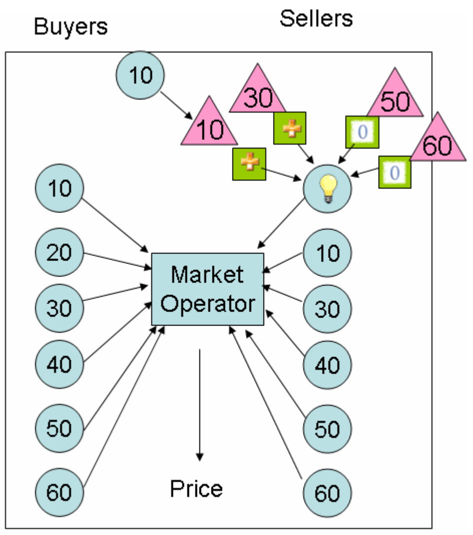


Figure 3.3 SimpleModel-II: One Learning Agent with Two Profitable Actions

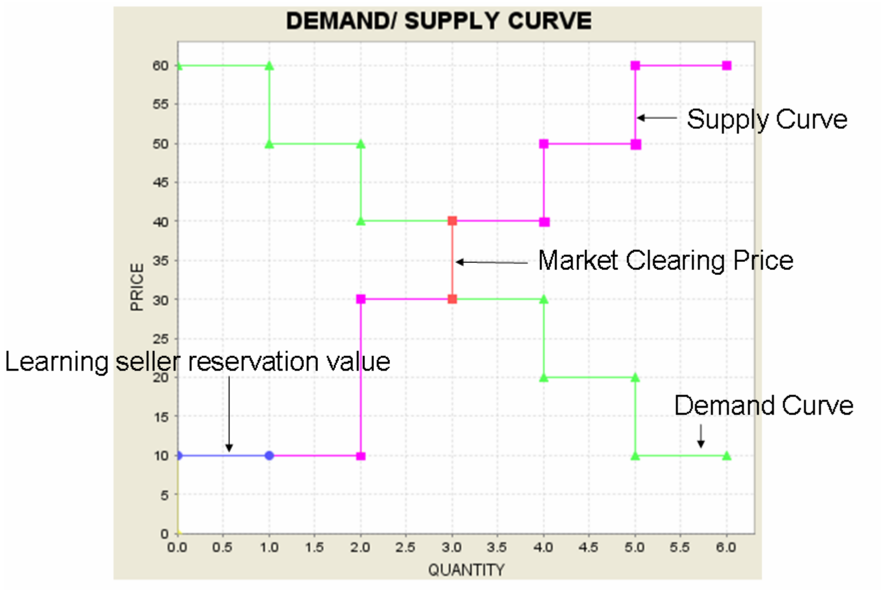


Figure 3.4 Demand Supply Curve for Simple Model-II with true reservation values

3.2.2 Experimental Design

We run the similar set of experiments by varying the experimentation and recency parameters from 0.0 to 1.0 with an interval of 0.1 and with a 100 random seeds. The random seeds are exactly the same as the random seeds used in the previous test case in order to make the enivornmenal conditions as similar as possible.

3.3 AMESModel-I: One Learning Agent

3.3.1 Model Design

The AMES market package is a wholesale power market platform simulation that consists of a transmission grid with a set of electricity traders consisting of generators and load supplying entities (LSEs) situated on the nodes of the transmission grid.

During each time period, based on the load schedule and the generator ask prices, the market operator, known as the independent service operator (ISO), calculates the commitments from each generator and therefore the generator profits.

The AMES package is explained in detail in the testbed implementation section D.2.1.

The AMES package comes with an example of a transmission grid containing five nodes. The experimental setup is based on this five node test case. This setup has five generators and five LSEs located on the transmission grid. All generators are fixed to use the same experimentation and recency parameter settings, except the learning generator G5. This generator uses the settings specified in the multi-run parameter file to sweep through the parameter space. The number of actions for the generators G1 - G4 is set to one, while the number of actions of the learning generator is set to forty.

For the AMESModel experiments, a modification has been made to the original Roth-Erev algorithm to avoid the possibility of negative propensities and to be able to compare the Roth-Erev and the VRE RL algorithms on an equal footing. Therefore, for the AMESModel experiments using the Roth-Erev RL algorithm, the choice probability update function 2.3 has been replaced by the Gibbs-Boltzmann distribution function shown in the VRE RL algorithm 2.9.

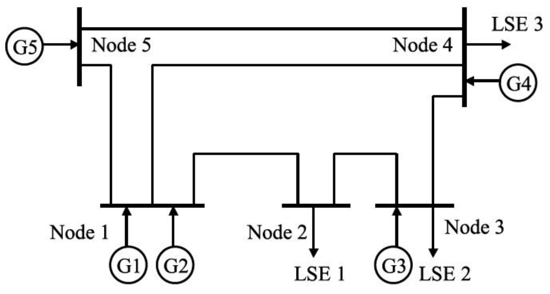


Figure 3.5 5-Node Test Case

3.3.2 Experimental Design

In the AMESModel-I experiments, for the learning generator G5, the recency and experimentation parameters were varied from 0.0 to 1.0 in increments of 0.1. All non-learning generators submit their true costs and capacities during each trading round. The initial propensity for all actions of the learning generator G5 were fixed at 140000 and the cooling parameter was fixed at 35000.

In the second experiment, all the parameters were the same as in the first experiment, except for the initial propensity being set to 6000 for the learning generator G5. The cooling parameter for the learning generator G5 was fixed at 1000.

Two sets of experiments are carried out with the above parameter settings. In one set of experiments, the learning agent is a Roth-Erev RL agent and in the second set of experiments, the learning agent is a VRE RL agent.

A total of $2\times2\times11\times11\times100$ simulations were carried out in the AMESModel-I experiments.

3.4 AMESModel-II: Two Learning Agents

3.4.1 Model Design

AMESModel-II is exactly the same as AMESModel-I with the following changes. Generator G4 is now a learning generator with forty actions and at each time period chooses its action using the VRE RL algorithm. For G4, the experimentation parameter was fixed at 0.97 and the recency parameter was fixed at 0.04 for all runs. In the first experiment, the initial propensity of generator G4 was fixed at 140000 and the Gibbs-Boltzmann cooling parameter was set to 35000.

In the second experiment, all the parameters were the same as in the first experiment, except for the initial propensity for generator G4 being set to 6000. The cooling parameter for generator G4 was also fixed at 1000.

3.4.2 Experimental Design

Same as AMESModel - I.

3.5 AMESModel-III: Five Learning Agents

3.5.1 Model Design

AMESModel-III has the same setup as AMESModel-I with the following change. Generators G1 - G4 are now learning agents with forty actions and choose their actions during each time period using the VRE RL algorithm. For generators G1 - G4, the experimentation parameter was fixed at 0.97, and the recency parameter was fixed at 0.4 for all runs. In the first experiment, the initial propensity for G1 - G4 was fixed at 140000 and the Gibbs-Boltzmann cooling parameter was set to 35000.

In the second experiment, all the parameters were the same as in the first experiment, except for the initial propensity for generators G1 - G4 being set to 6000. The cooling parameter for generators G1 - G4 was also fixed at 1000.

3.5.2 Experimental Setup

Same as AMESModel-I.

CHAPTER 4. RESULTS

The following four criteria are used to compare the performance of the three tested RL algorithms: Original Roth-Erev; Modified Roth-Erev (MRE); and Variant Roth-Erev (VRE).

1. *Average Total Profits*: For each simulation run, the total profits for the learning agent accrued over the run are calculated and reported as the Total Profit for the run. For each tested setting of the recency and experimentation parameters r and e, 100 simulation runs are conducted corresponding to 100 distinct random seeds. The Average Total Profits is the sum of the Total Profit obtained over the 100 runs divided by the number of runs.

   For the SimpleModel experiments, the Average Total Profits are calculated based on the profits of the learning seller only, and for the AMESModel experiments, the Average Total Profits are calculated based on the profits of the generator G5 only.

   For the SimpleModel experiments, the color scale varies from 0 to the maximum Average Total Profits that can be obtained by the learning seller agent.

2. *Convergent Action*: In a simulation run, at the final time period, for the learning seller, the action with choice probability greater than 0.999, if any, is denoted as the Convergent Action. A unique color is used denote each action in the generator's set of actions. For each simulation, if the Convergent Action exists, then using the unique color of that action, a dot is placed on the appropriate cell in the heat map. If no Convergent Action exists for a simulation run, then it is marked in black on the heat map.

   For the SimpleModel experiments, the Convergent Action is noted for the learning seller only, and for the AMESModel experiments, the Convergent Action is noted for the generator G5 only.

For the SimpleModel experiments, the color scale indicates the colors corresponding to the two actions, red for Action 0, and blue for Action 1. For the AMESModel experiments, the color scale indicates the color corresponding to each action in the learning agent's action domain and varies from 0 to 40.

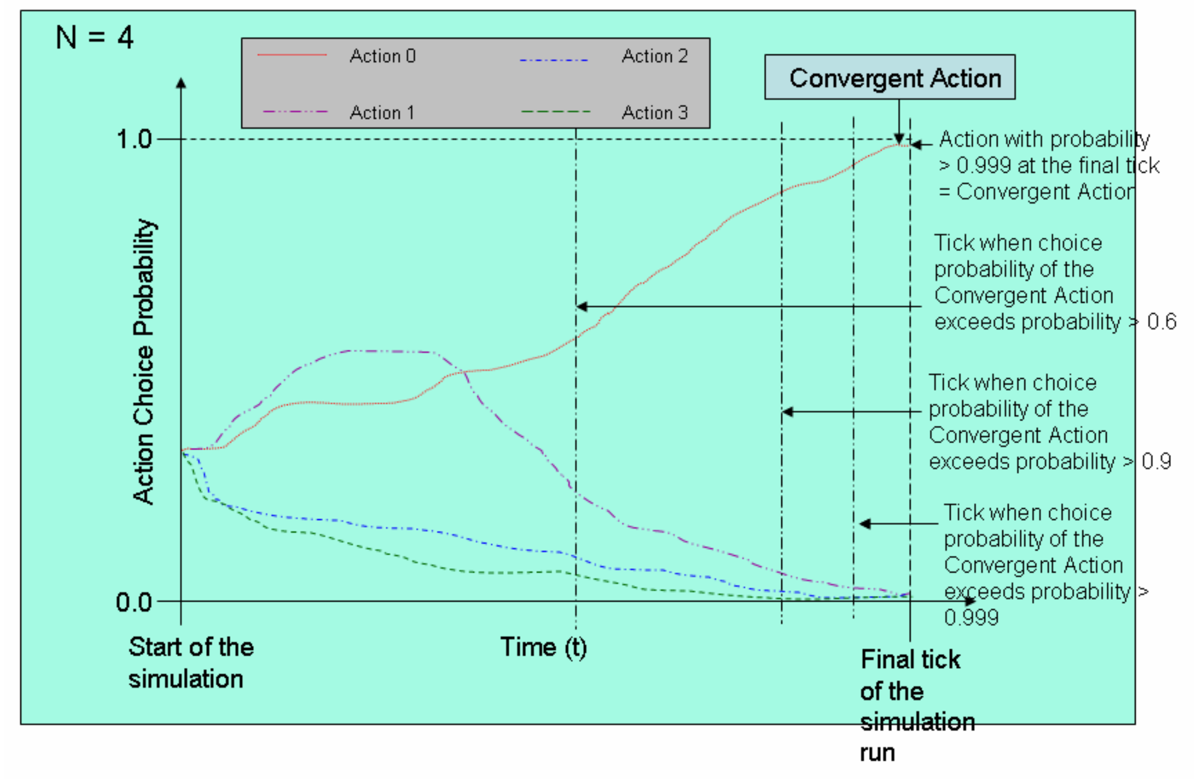

N = 4
Action 0
Action 1
Action 2
Action 3
Convergent Action
1.0
0.0
Action Choice Probability
Action with probability > 0.999 at the final tick = Convergent Action
Tick when choice probability of the Convergent Action exceeds probability > 0.6
Tick when choice probability of the Convergent Action exceeds probability > 0.9
Tick when choice probability of the Convergent Action exceeds probability > 0.999
Start of the simulation
Time (t)
Final tick of the simulation run

Figure 4.1 Sketch of a simulation run indicating the Convergent Action

3. *Average Time for Convergent Action to reach Threshold Probability*: The time periods at which the choice probability of the Convergent Action for the learning seller reaches the threshold probability of 0.6, 0.9 and 0.999 are noted. For the 100 simulations with the same experimentation and recency settings, the averages of such time periods are calculated. This average time is denoted as the Average Time for Convergent Action Reaching Threshold Probability. In the convergence graph, each cell is colored based on the average time at which the Convergent Action choice probability exceeds the threshold

probability. For a given experimentation and recency setting, if no simulation runs have a Convergent Action, then the cells are left blank.

For the SimpleModel experiments, the Average Time Periods are calculated based on the Convergent Action for the learning seller only, and for the AMESModel experiments, the Average Time Periods are calculated based on the Convergent Action for the generator G5 only.

The color scale indicates the time value starting from 0.

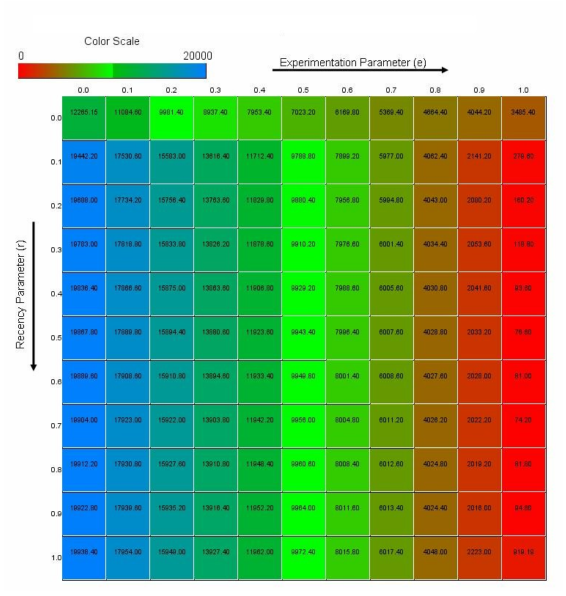
4. *Scaled Average Total Profits*: In the AMESModel experiments, the cells are colored based on the ratio of the Average Total Profits of the cell to the maximum Average Total Profits over the entire parameter space.

   The Average Total Profits are calculated based on the Total Profits of generator G5 only.

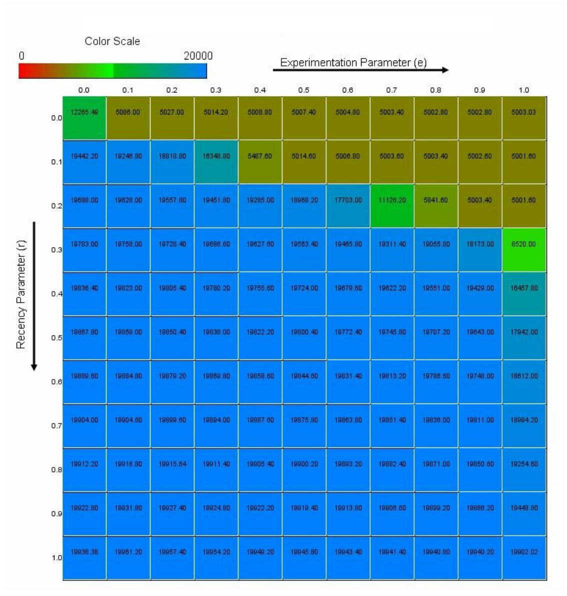
   The graph colors are based on a ratio and so the color scale indicates a value varying from 0.0 to 1.0.

4.1 SimpleModel-I: One Learning Agent with One Profitable Action

4.1.1 Experiment 1: High initial propensity



(a) Roth-Erev Algorithm



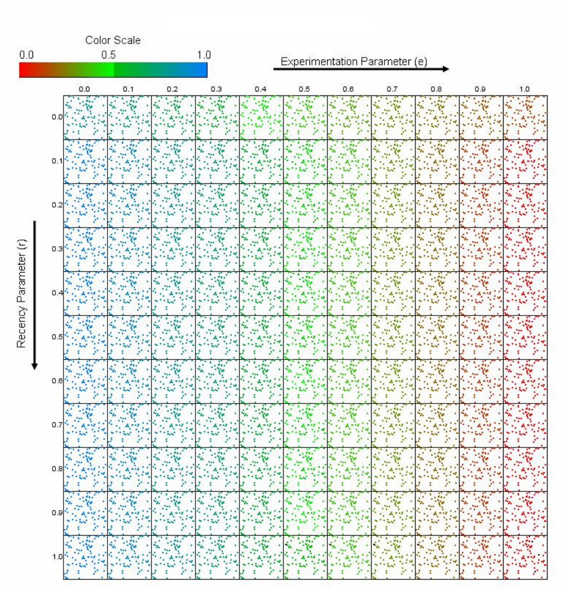
(b) Modified Roth-Erev Algorithm

Figure 4.2 Average Total Profits for the learning seller in SimpleModel-I: Experiment 1 (High Initial Propensity)

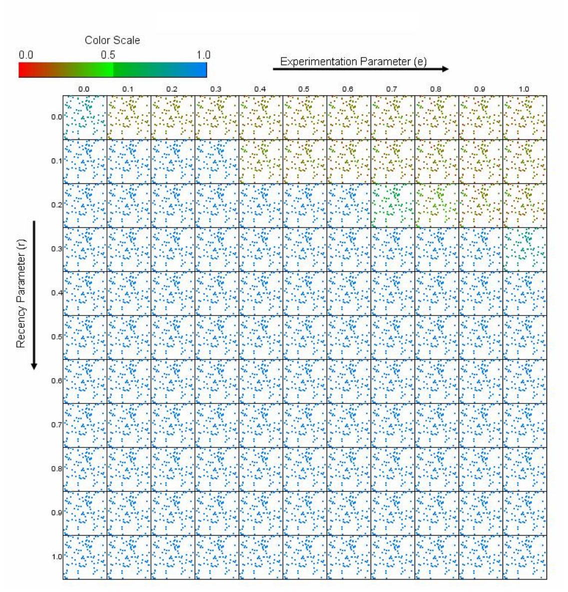
The graph for the profits of the learning seller on the Roth-Erev RL experiment shows a large gradation in the Average Total Profits. The mathematical analysis explains the results 5.4 and 5.5. The results are consistent with the results from [11].

In the MRE RL experiment, the green triangular zone at the top of the graph is because in that region, the recency r is such that $r < \frac{e}{N-1}$. A mathematical explanation of this region is given in 5.10. The learning seller using the MRE RL algorithm is able to obtain maximum profit over a much larger range of recency r and experimentation e.

The largest profits obtained in the Roth-Erev RL experiment are at (r, e) = (1.0, 0.0) and (0.9, 0.0). The largest profits obtained in the MRE RL experiment are at (r, e) = (1.0, 0.1) and (1.0, 0.2).



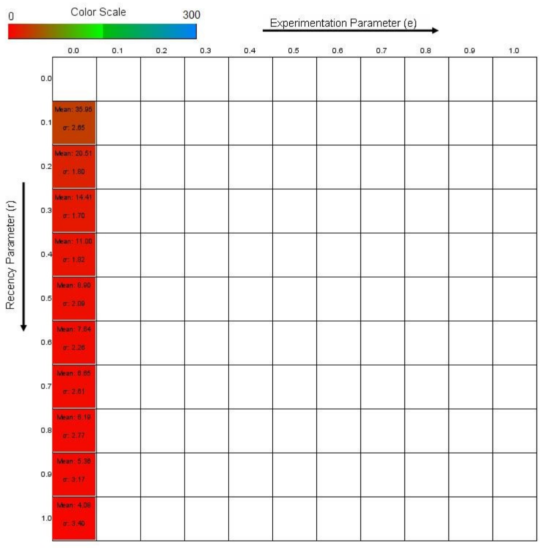
(a) Roth-Erev Algorithm



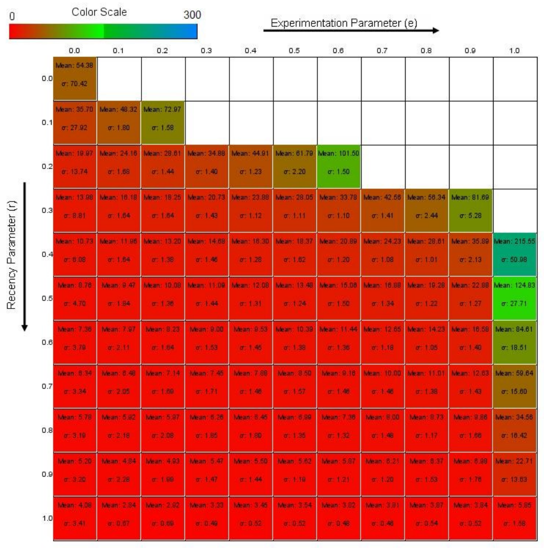
(b) Modified Roth-Erev Algorithm

Figure 4.3 Choice Probability of the profitable action at tick 1000 for the learning seller in SimpleModel-I: Experiment 1 (High Initial Propensity)

In the Roth-Erev RL experiment, the choice probability of the profitable action converges to a smaller probability value as the experimentation e value increases. This affects the profits, since only a single action generates a positive profit. There is a clear correlation between the choice probability of the profitable action and the profits obtained by the learning seller in both algorithms: when the profitable action appears to converge to a probability 1.0, the profits are maximum.



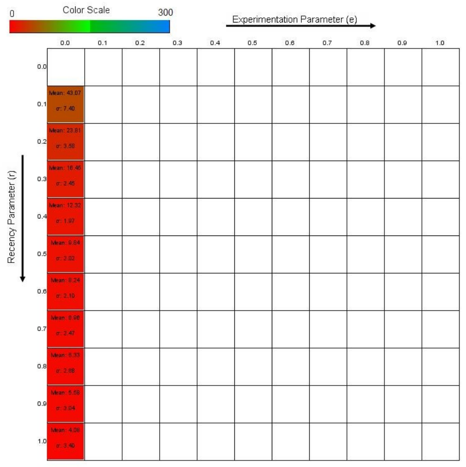
(a) Roth-Erev Algorithm



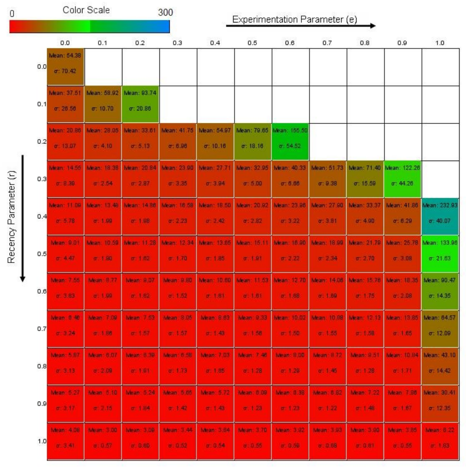
(b) Modified Roth-Erev Algorithm

Figure 4.4 Average Time period for the choice probability of the profitable action to reach 0.6 for the learning seller in SimpleModel-I: Experiment 1 (High Initial Propensity)

In the Roth-Erev RL experiment, the only convergence occurs when the recency is 0.0. This is a result of the convergence to a different value than 1.0 as shown in the mathematical analysis of SimpleModel-I (Refer to: 5.4 and 5.5). In the MRE RL experiment, the profitable action choice probability converges to 1.0 relatively fast over a large area. The MRE RL agent does not appear to converge only in the top right triangular region.

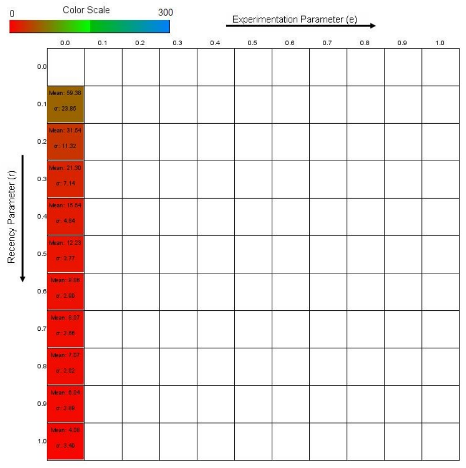


(a) Roth-Erev Algorithm

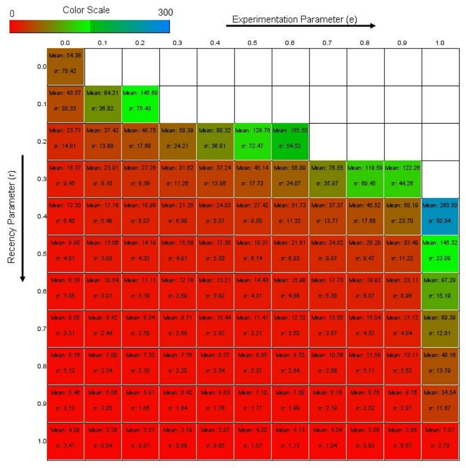


(b) Modified Roth-Erev Algorithm

Figure 4.5 Average Time period for the choice probability of the profitable action to reach 0.9 for the learning seller in SimpleModel-I: Experiment 1 (High Initial Propensity)



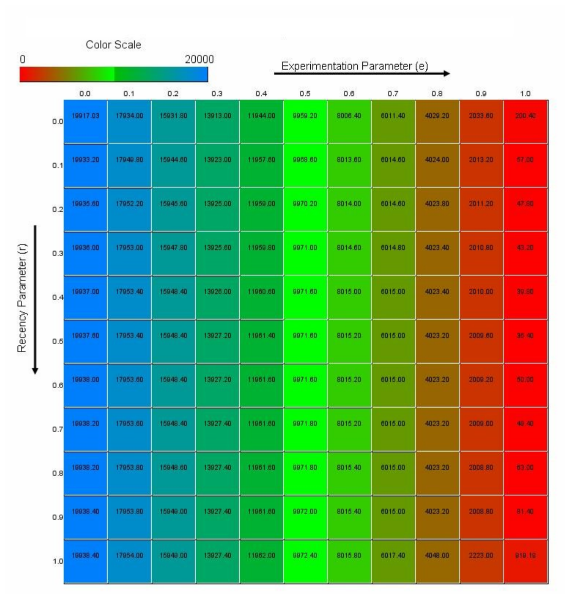
(a) Roth-Erev Algorithm



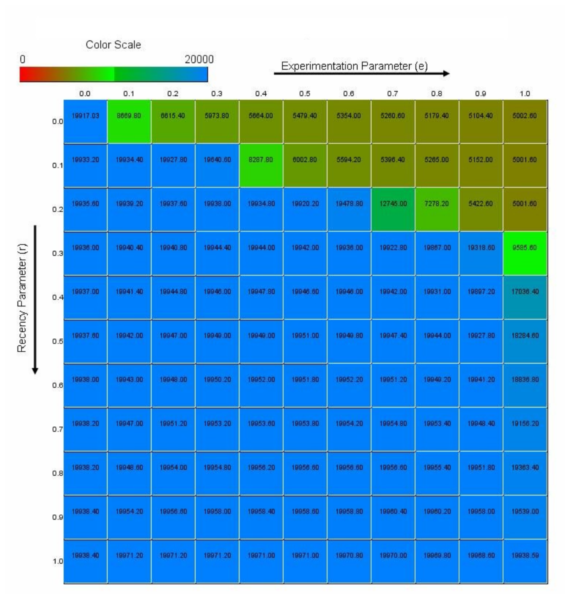
(b) Modified Roth-Erev Algorithm

Figure 4.6 Average Time period for the choice probability of the profitable action to reach 0.999 for the learning seller in SimpleModel-I: Experiment 1 (High Initial Propensity)

4.1.2 Experiment 2: Low initial propensity



(a) Roth-Erev Algorithm



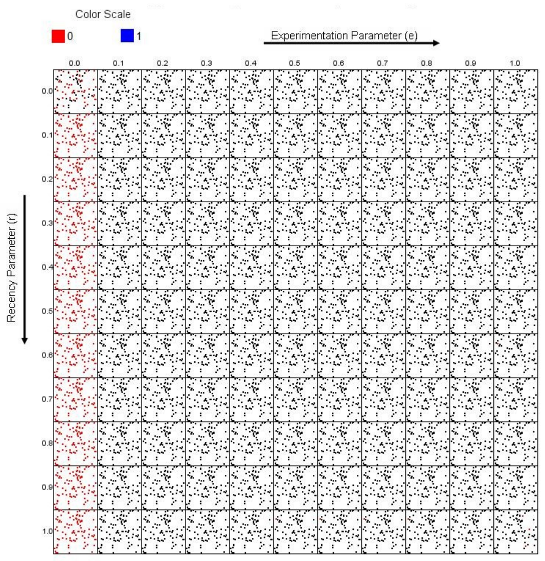
(b) Modified Roth-Erev Algorithm

Figure 4.7 Average Total Profits of the learning seller in SimpleModel-I: Experiment 2 (Low Initial Propensity)

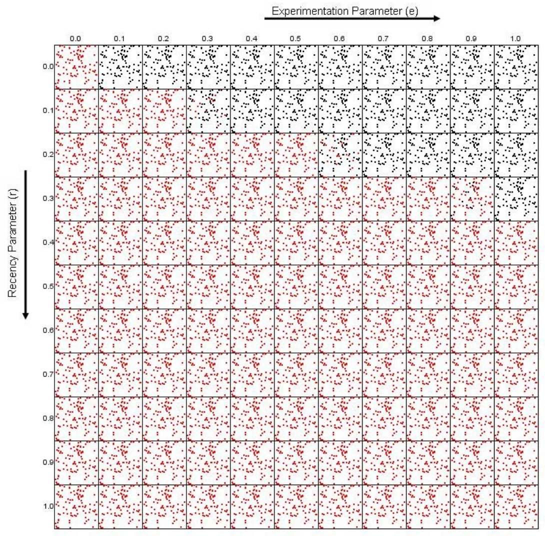
In the Roth-Erev RL experiment, the profits obtained by the learning agent do not seem to be impacted by substituting the high initial propensity of the learning agent with a low initial propensity. This can be explained by the convergence of the profitable action choice probability to a value less than 1.0 [1]. In the MRE RL experiment, the heat map is very similar to Experiment 1 (High Initial Propensity), indicating that the algorithm appears to be largely unaffected by the initial propensity.

The largest profits obtained for the learning seller in the RE RL experiment are at (r, e) = (1.0, 0.0) and (0.9, 0.0). The largest profits obtained for the learning seller in the MRE RL experiment are at (r, e) = (1.0, 0.1), (1.0, 0.2), (1.0, 0.3), (1.0, 0.4) and (1.0, 0.5).

[1]Refer to 5.4 and 5.5



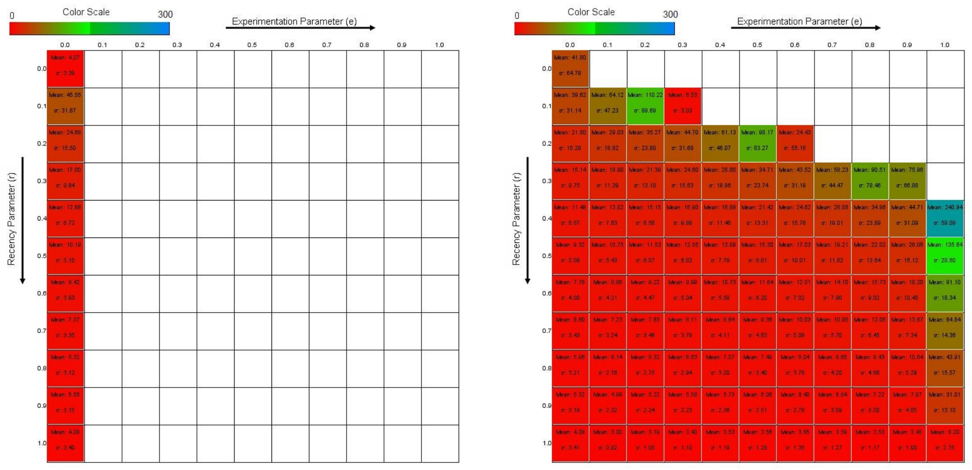
(a) Roth-Erev Algorithm



(b) Modified Roth-Erev Algorithm

Figure 4.8 Convergent Action for the learning seller in SimpleModel-I: Experiment 2 (Low Initial Propensity)

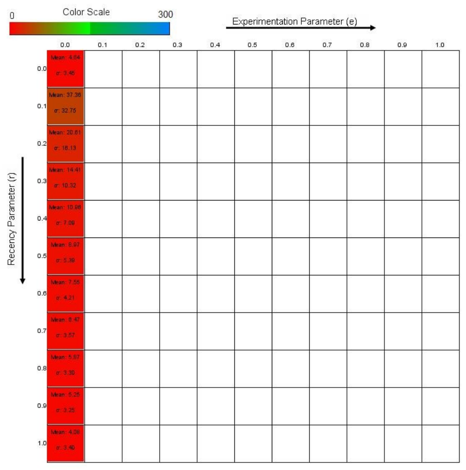
Once again, the profits are higher when the algorithm converges to the profitable action.



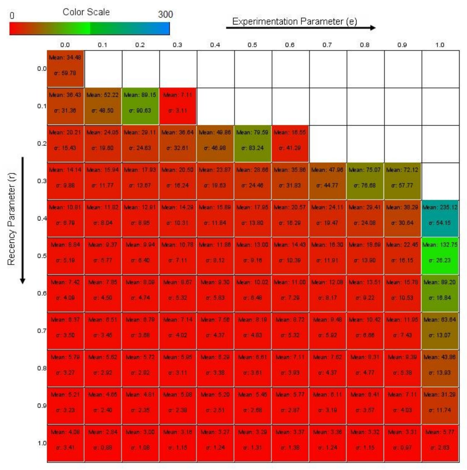
(a) Roth-Erev Algorithm (b) Modified Roth-Erev Algorithm

Figure 4.9 Average Time period for the choice probability of the profitable action to reach 0.6 for the learning seller in SimpleModel-I: Experiment 2 (Low Initial Propensity)

As can be seen from the graphs above, reducing the initial propensity to 1.0 has negigible effect on the Average Time Period for the Convergent Action reaching the Threshold Probabilities.

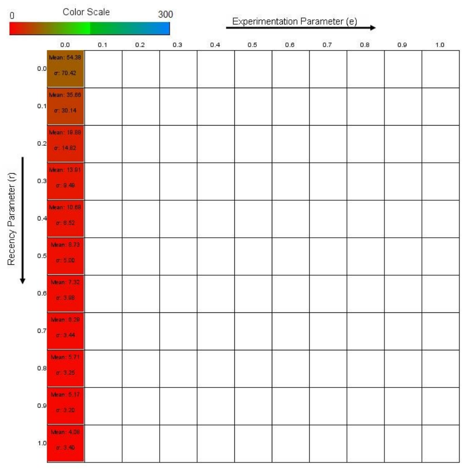


(a) Roth-Erev Algorithm

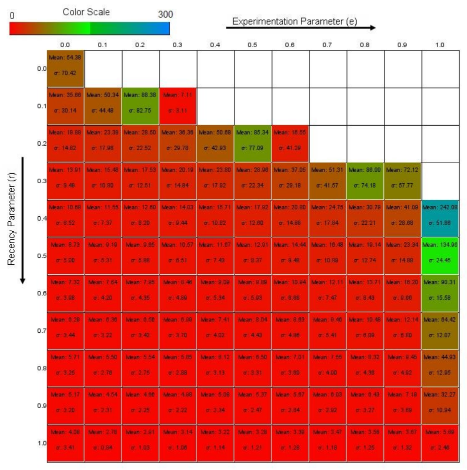


(b) Modified Roth-Erev Algorithm

Figure 4.10 Average Time period for the choice probability of the profitable action to reach 0.9 for the learning seller in SimpleModel-I: Experiment 2 (Low Initial Propensity)



(a) Roth-Erev Algorithm

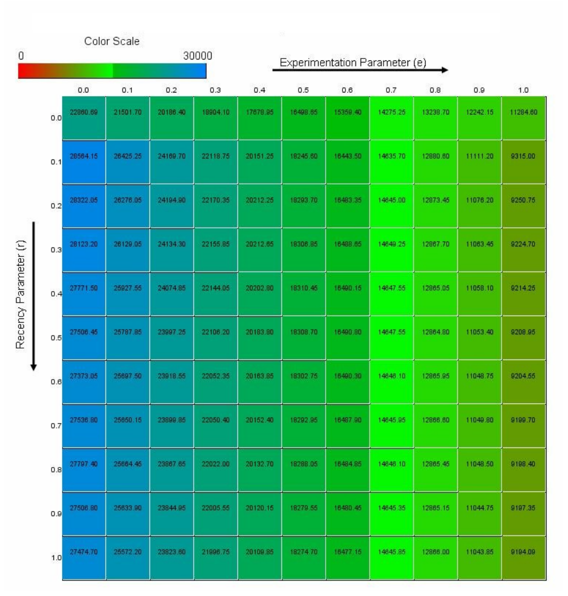


(b) Modified Roth-Erev Algorithm

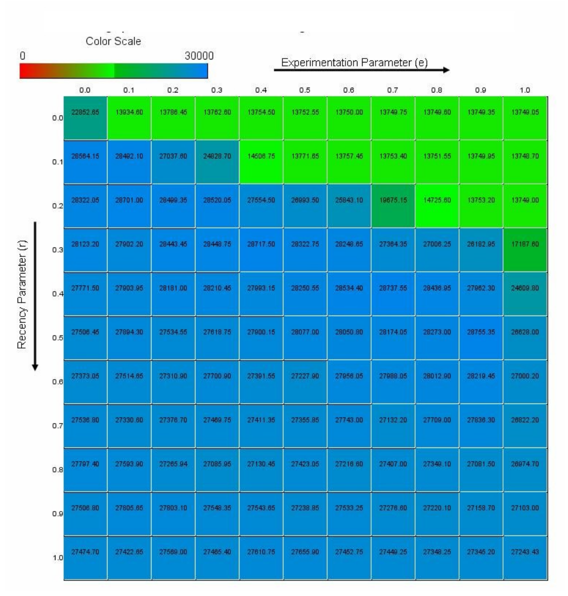
Figure 4.11 Average Time period for the choice probability of the profitable action to reach 0.999 for the learning seller in Simple-Model-I: Experiment 2 (Low Initial Propensity)

4.2 SimpleModel-II: One Learning Agent with Two Profitable Actions

4.2.1 Experiment 1: High initial propensity



(a) Roth-Erev Algorithm

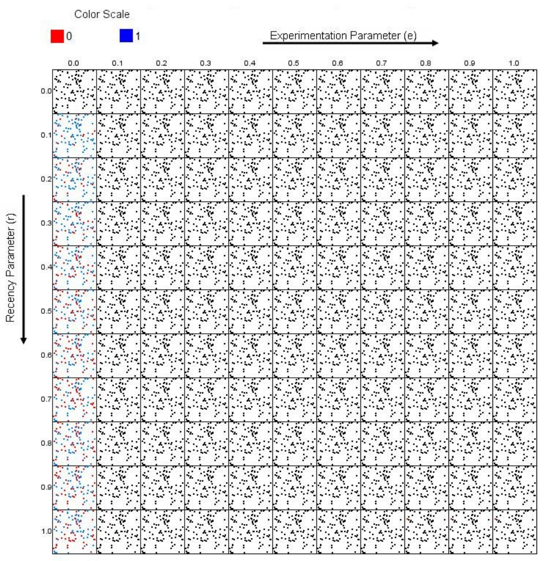


(b) Modified Roth-Erev Algorithm

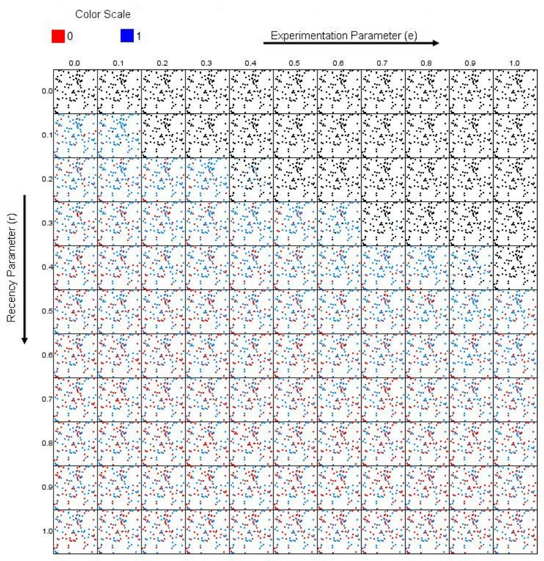
Figure 4.12 Average Total Profits of the learning seller in SimpleModel-II: Experiment 1 (High Initial Propensity)

It is likely that the increased profits for the learning seller shown in the two experiments are because there are two actions that generate positive profits. This directly increases the chance of selecting a rewarding action even in a pure random selection policy. The mathematical analysis done for SimpleModel-I can probably be used to explain the very similar gradations in the SimpleModel-II experiments. The upper-triangular region in the MRE RL experiment is due to the propensity limit 5.10. Once again, the MRE RL algorithm performs much better over a wide range of parameter settings.

The largest profits obtained in the RE RL experiment are at (r, e) = (0.1, 0.0) and (0.2, 0.0). The largest profits obtained in the MRE RL experiment are at (r, e) = (0.5, 0.9) and (0.4, 0.7).



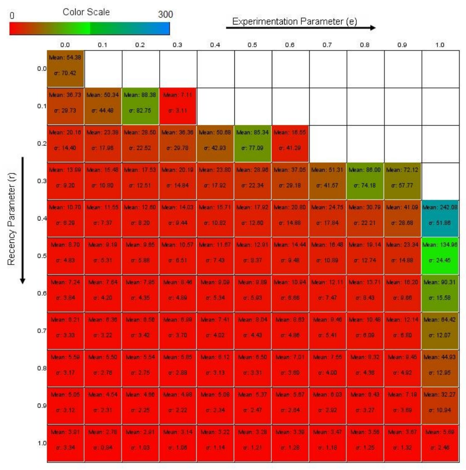
(a) Roth-Erev Algorithm



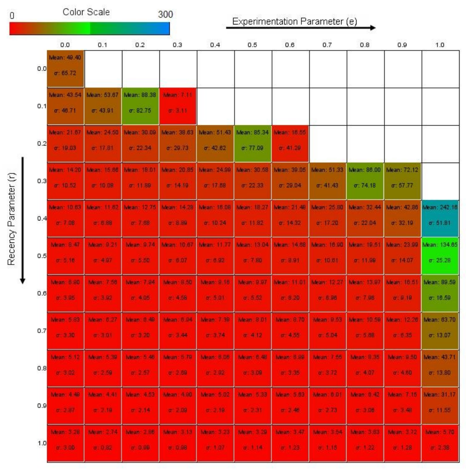
(b) Modified Roth-Erev Algorithm

Figure 4.13 Choice Probability of the Convergent Action at tick 1000 for the learning seller in SimpleModel-II: Experiment 1 (High Initial Propensity)

Interestingly enough, neither the Roth-Erev nor the MRE RL algorithms are able to consistently converge to the action with the largest profit. This is probably because of the relatively small difference between the rewards that are obtained in both actions for the learning seller. To accentuate the differences, a Gibbs-Boltzmann distribution can be used with an appropriate cooling parameter.



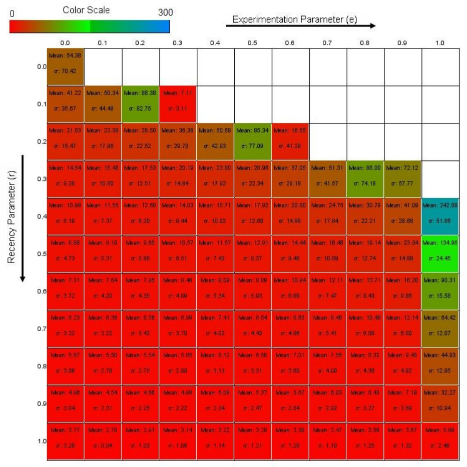
(a) Roth-Erev Algorithm



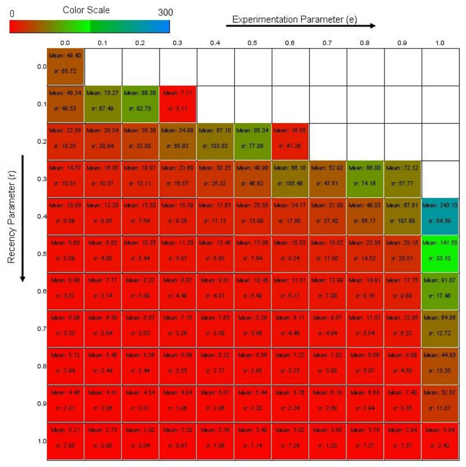
(b) Modified Roth-Erev Algorithm

Figure 4.14 Average Time Period for the Convergent Action choice probability reaching 0.6 for the learning seller in SimpleModel-II: Experiment 1 (High Initial Propensity)

The graphs show a gradient in the convergence. As the recency moves from 1.0 to 0.0 and experimentation parameters moves from 0.0 to 1.0, the time taken to converge to the Convergent Action increases. The increase rises as the recency parameter decreases and the experimentation parameter increases. This increase appears to be almost exponential and we conjecture that the convergence time will increase faster with smaller variations in the experimentation and recency parameters. Also, with the increase in the convergence time, the variance also increases sharply. The convergences are comparable in both the experiments.

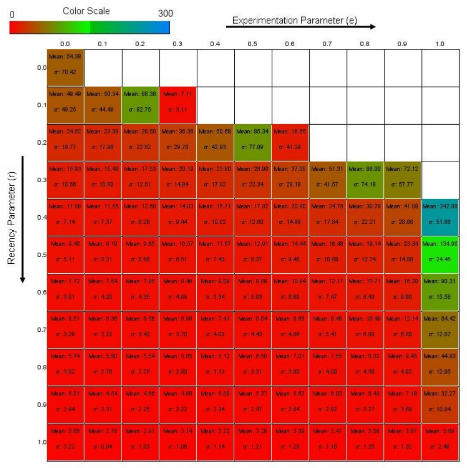


(a) Roth-Erev Algorithm

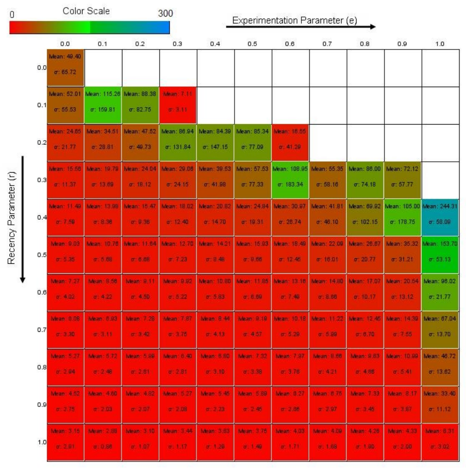


(b) Modified Roth-Erev Algorithm

Figure 4.15 Average Time Period for the Convergent Action choice probability reaching 0.9 for the learning seller in SimpleModel-II: Experiment 1 (High Initial Propensity)



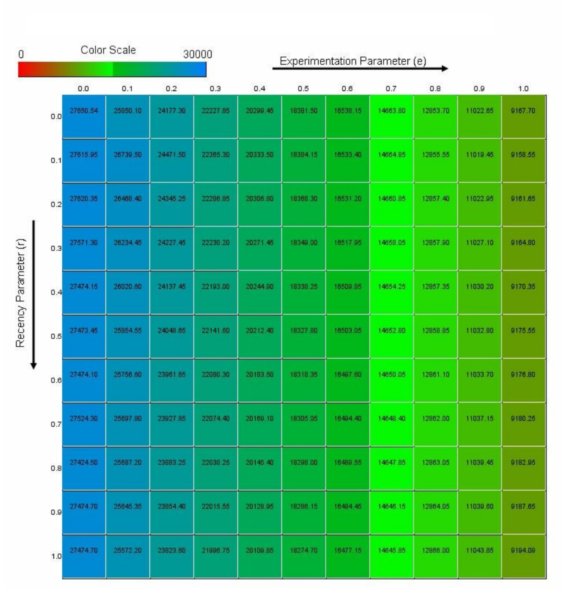
(a) Roth-Erev Algorithm



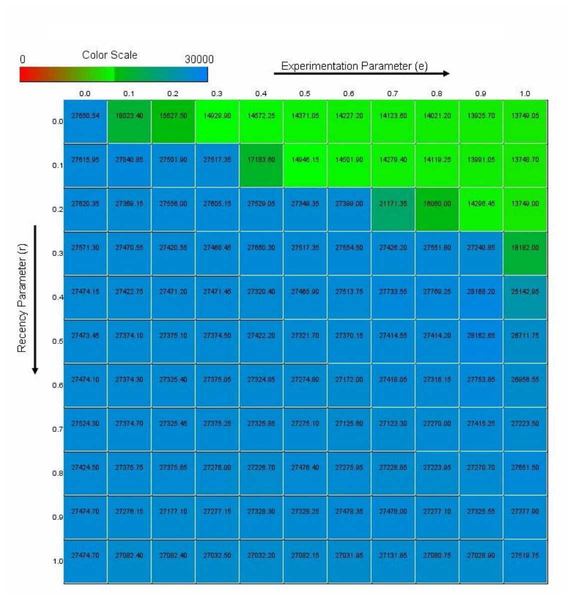
(b) Modified Roth-Erev Algorithm

Figure 4.16 Average Time Period for the Convergent Action choice probability reaching 0.999 for the learning seller in SimpleModel-II: Experiment 1 (High Initial Propensity)

4.2.2 Experiment 2: Low initial propensity



(a) Roth-Erev Algorithm

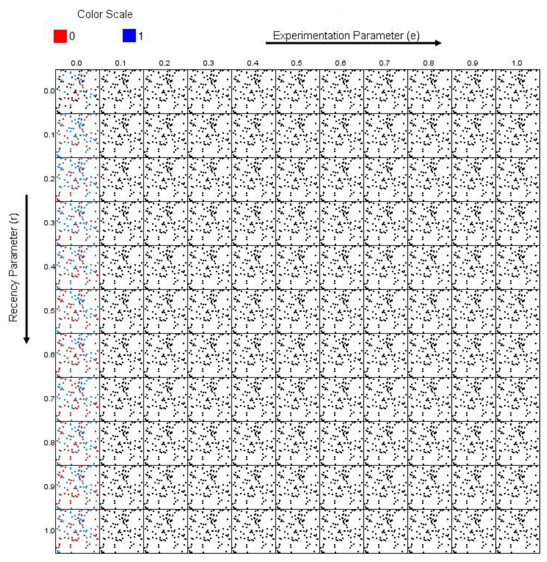


(b) Modified Roth-Erev Algorithm

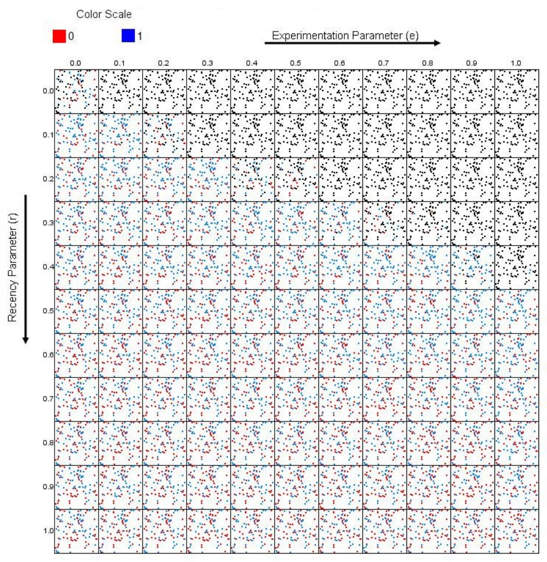
Figure 4.17 Average Total Profits of the learning seller in SimpleModel-II: Experiment 2 (Low Initial Propensity)

With low initial propensity, the profits and the convergences for the learning seller are very similar. The results can be explained in a similar manner as in Experiment 1 (High Initial Propensity).

The largest profits obtained for the learning seller in the RE RL experiment are at (r, e) = (0.0, 0.0) and (0.1, 0.0). The largest profits obtained in the VRE RL experiment are at (r, e) = (0.4, 0.9) and (0.5, 0.9).

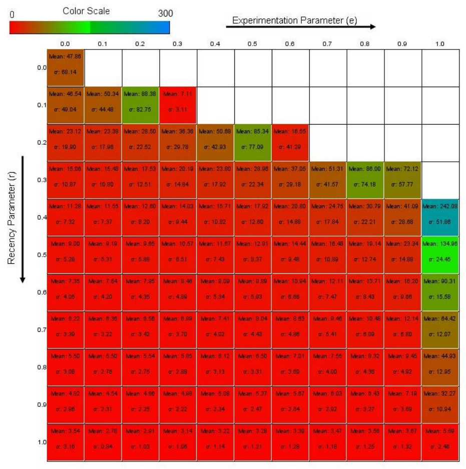


(a) Roth-Erev Algorithm

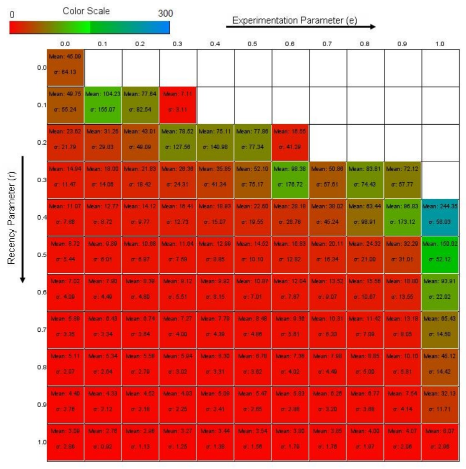


(b) Modified Roth-Erev Algorithm

Figure 4.18 Convergent Action for the learning seller in SimpleModel-II: Experiment 2 (Low Initial Propensity)

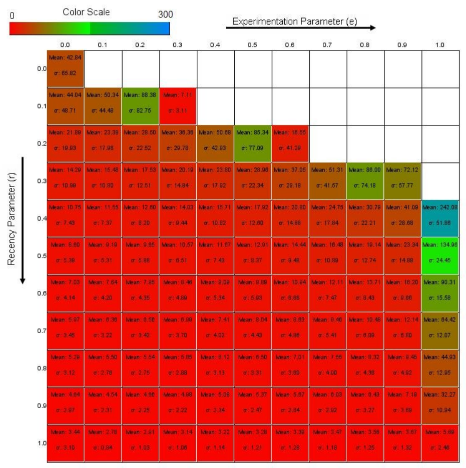


(a) Roth-Erev Algorithm

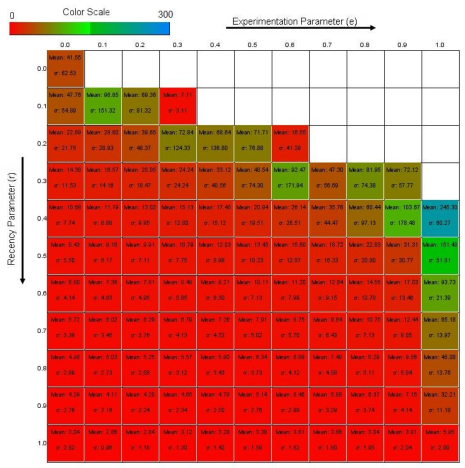


(b) Modified Roth-Erev Algorithm

Figure 4.19 Average Time Period for the Convergent Action choice probability reaching 0.6 for the learning seller in SimpleModel-II: Experiment 2 (Low Initial Propensity)

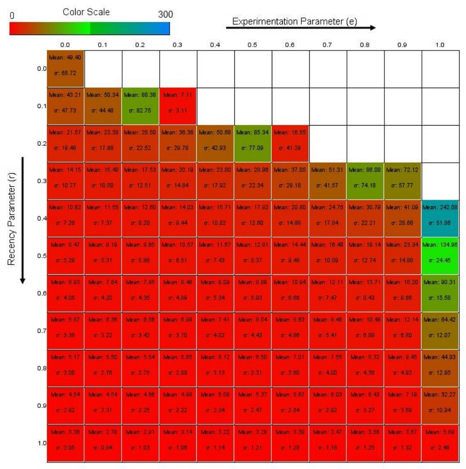


(a) Roth-Erev Algorithm

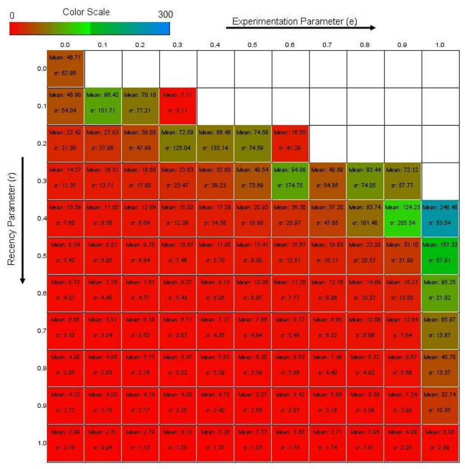


(b) Modified Roth-Erev Algorithm

Figure 4.20 Average Time Period for the Convergent Action choice probability reaching 0.9 for the learning seller in SimpleModel-II: Experiment 2 (Low Initial Propensity)



(a) Roth-Erev Algorithm

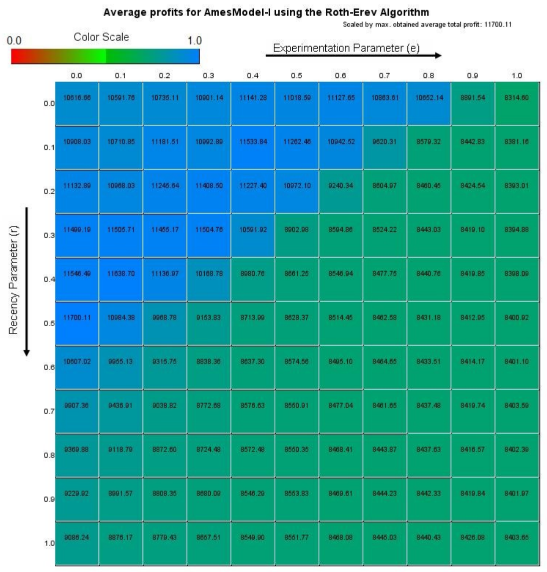


(b) Modified Roth-Erev Algorithm

Figure 4.21 Average Time Period for the Convergent Action choice probability reaching 0.999 for the learning seller in SimpleModel-II: Experiment 2 (Low Initial Propensity)

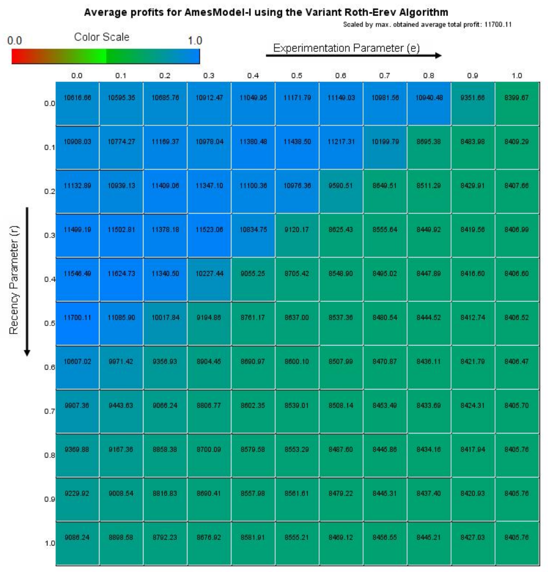
4.3 AMESModel-I: One Learning Agent

4.3.1 Experiment 1: High initial propensity



	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.0	10616.66	10591.76	10735.11	10901.14	11141.28	11018.59	11127.65	10863.61	10652.14	8891.54	8314.60
0.1	10908.03	10710.85	11181.51	10992.89	11533.84	11262.46	10942.52	9620.31	8579.32	8442.83	8381.16
0.2	11132.89	10968.03	11245.64	11408.50	11227.40	10972.10	9240.34	8604.97	8460.45	8424.54	8393.01
0.3	11499.19	11505.71	11455.17	11504.76	10591.92	8902.98	8594.86	8524.22	8443.03	8419.10	8394.88
0.4	11546.49	11638.70	11136.97	10168.78	8980.76	8661.25	8546.94	8477.75	8440.76	8419.85	8398.09
0.5	11700.11	10984.38	9968.78	9153.83	8713.99	8628.37	8514.45	8462.58	8431.18	8412.95	8400.92
0.6	10607.02	9955.13	9315.75	8838.36	8637.30	8574.56	8495.10	8464.65	8433.51	8414.17	8401.10
0.7	9907.36	9436.91	9038.82	8772.68	8576.63	8550.91	8477.04	8461.65	8437.48	8419.74	8403.59
0.8	9369.88	9118.79	8872.60	8724.48	8572.48	8550.35	8468.41	8443.87	8437.63	8416.57	8402.39
0.9	9229.92	8991.57	8808.35	8680.09	8546.29	8553.83	8469.61	8444.23	8442.33	8419.84	8401.97
1.0	9086.24	8876.17	8779.43	8657.51	8549.90	8551.77	8468.08	8446.03	8440.43	8426.08	8403.65

(a) Roth-Erev Algorithm



	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.0	10616.66	10595.35	10685.76	10912.47	11049.95	11171.79	11149.03	10981.56	10940.48	9351.66	8399.67
0.1	10908.03	10774.27	11169.37	10978.04	11380.48	11438.50	11217.31	10199.79	8695.38	8483.98	8409.29
0.2	11132.89	10939.13	11409.06	11347.10	11100.36	10976.36	9590.51	8649.51	8511.29	8429.91	8407.66
0.3	11499.19	11502.81	11378.18	11523.06	10834.75	9120.17	8625.43	8555.64	8449.92	8419.56	8406.99
0.4	11546.49	11624.73	11340.50	10227.44	9055.25	8705.42	8548.90	8495.02	8447.89	8416.60	8406.60
0.5	11700.11	11085.90	10017.84	9194.86	8761.17	8637.00	8537.36	8480.54	8444.52	8412.74	8406.52
0.6	10607.02	9971.42	9356.93	8904.45	8690.97	8600.10	8507.99	8470.87	8436.11	8421.79	8406.47
0.7	9907.36	9443.63	9066.24	8806.77	8602.35	8539.01	8508.14	8453.49	8433.69	8424.31	8405.70
0.8	9369.88	9167.36	8858.38	8700.09	8579.58	8553.29	8487.60	8445.86	8434.16	8417.94	8405.76
0.9	9229.92	9008.54	8816.83	8690.41	8557.98	8561.61	8479.22	8445.31	8437.40	8420.93	8405.76
1.0	9086.24	8898.58	8792.23	8676.92	8581.91	8555.21	8469.12	8456.55	8445.21	8427.03	8405.76

(b) Variant Roth-Erev Algorithm

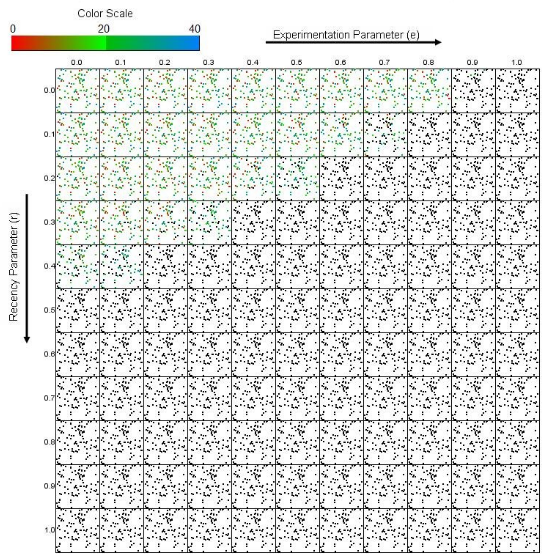
Figure 4.22 Scaled Average Total Profits for generator G5 in AMESModel-I: Experiment 1 (High Initial Propensity)

The profits shown in the cells of the heat map are scaled by a factor of 10^3.

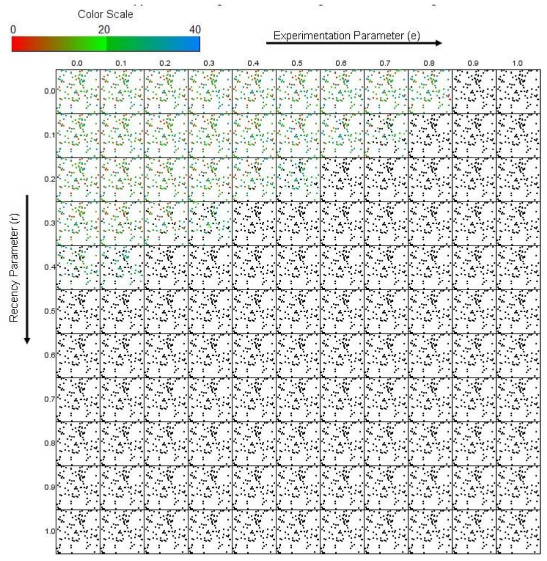
Similar to the results in SimpleModel-II and [11], the maximum rewards occur when the experimentation is close to 0. The results from the Roth-Erev and the VRE RL experiments are very similar because the setup of the transmission grid is such that generator G5 is always dispatched and always obtains a profit. Therefore, generator G5 never reaches a state where the 'zero updation' effect in the Roth-Erev RL algorithm arises.

The small region in the top left corner is the region where the convergence to a single action happens very fast. This does not tend to have the maximum rewards. The intermediate region has the maximum reward because the convergence to a single action happens based on the updating of the propensities that occurs due to experimentation. The lower region below that is the region where the propensity values shrink to such an extent that the probability of all actions tends to remain the same. This is the effect of the probabilities being updated based on the Gibbs-Boltzmann distribution and the large cooling parameter.

The different bands shown on the graph are a function of the recency and experimentation parameter settings. We observe that the bands appear to be on the periphery of the graph where the sum of the recency and experimentation appear to be constant. This can be explained by observing the update function in the RE and MRE RL equations 2.1 and 2.4. As the initial propensity and profits are similar in size, the effect of reducing the recency by 0.1 and at the same time adding 0.1 to the experimentation is approximately the same. This may explain the shape of the band for the maximum rewards.



(a) Roth-Erev Algorithm

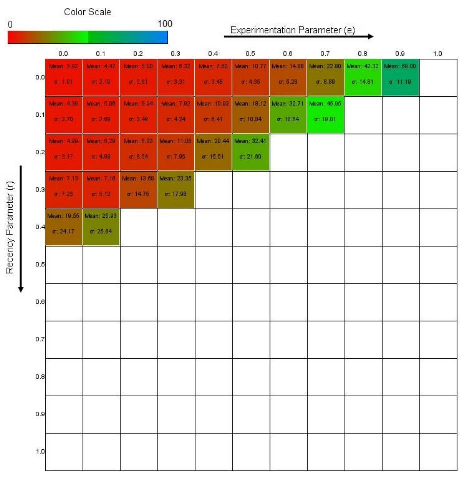


(b) Variant Roth-Erev Algorithm

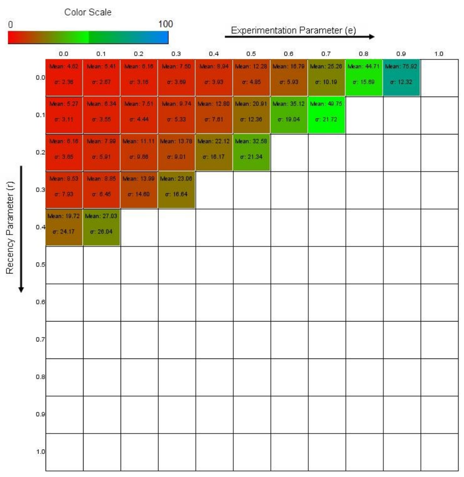
Figure 4.23 Convergent Action for generator G5 in AMESModel-I: Experiment 1 (High Initial Propensity)

The largest profits occur when the convergence is close to 20-25 ticks, indicating that a certain amount of experimentation is desirable even when there are no competing agents. In the top left corner, there is no clear single action to which the learning agent converges consistently. This is because all the actions in the action domain generate a positive profit and the algorithm can therefore converge to any one of them with some probability. In the region where the profits are the maximum, the simulation converges to a smaller subset of actions indicating that some actions are better than others and the learning agent is able to converge to this subset. In the remaining area of the graph, no single action attains a probability greater than 0.999 within the specified time period.

By viewing output XML files from the simulation runs corresponding to the remaining area in the graph, it is seen that the propensity values reduce much faster due to the large recency and experimentation parameters. The propensity values drop more rapidly because of the VRE RL algorithm spillover function 2.8, as compared to the Roth-Erev RL algorithm spillover function 2.2. However, in both cases, the cooling parameter remains very large in comparison to the propensity values. This attenuates the differences between the action propensities, and is reflected in the choice action probabilities. The learning agent therefore does not fixate on a single action.



(a) Roth-Erev Algorithm



(b) Variant Roth-Erev Algorithm

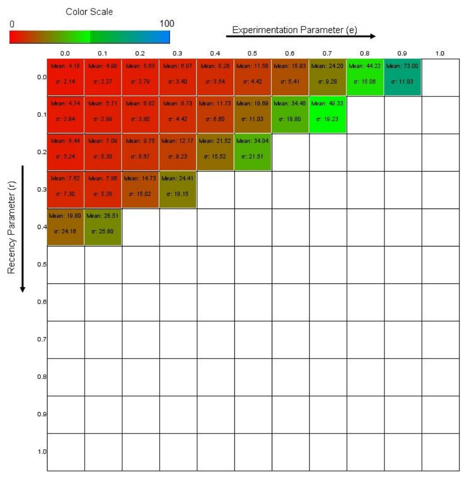
Figure 4.24 Average Time Period for the Convergent Action choice probability reaching 0.6 for generator G5 in AMESModel-I: Experiment 1 (High Initial Propensity)

In both the Roth-Erev and VRE RL experiments, with zero recency and zero experimentation, the spillover function does not update the propensities of any action. The update function simply adds the reward to the action propensity and calculates the profits. Since the probability calculation uses a Gibbs-Boltzmann distribution, the probability of the action chosen at the first tick increases much higher as compared to the other actions, thus tending to reach early convergence.

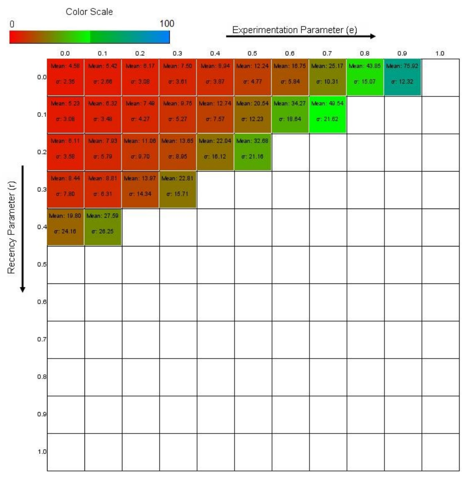
With increasing recency value and experimentation value, the propensity values reduce and the reward added to the propensity is reduced by a fraction based on the experimentation parameter. This makes the propensity of the remaining actions competitive with respect to the action chosen and this affects the convergence of the action.

As the recency and experimentation parameters are increased, the propensity values decrease even faster and because of the cooling factor, the probability of all actions tends to be approximately the same, making convergence to a single action nearly impossible.

The effect is more prominent because the initial propensity and the cooling parameter are larger compared to the actual profits obtained by the learning generator.

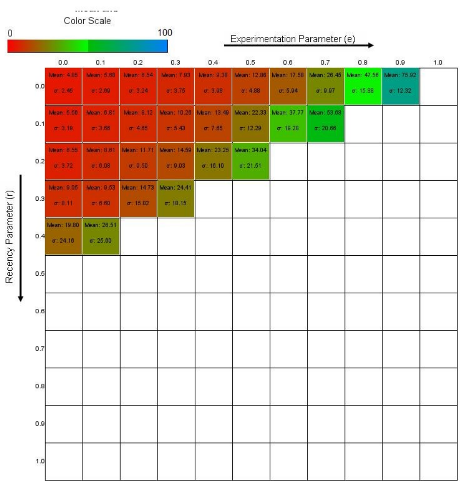


(a) Roth-Erev Algorithm

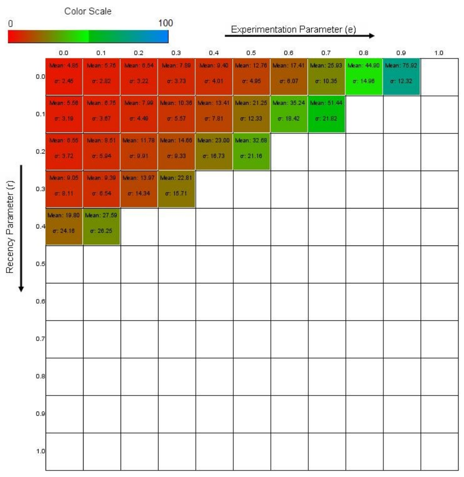


(b) Variant Roth-Erev Algorithm

Figure 4.25 Average Time Period for the Convergent Action choice probability reaching 0.9 for generator G5 in AMESModel-I: Experiment 1 (High Initial Propensity)



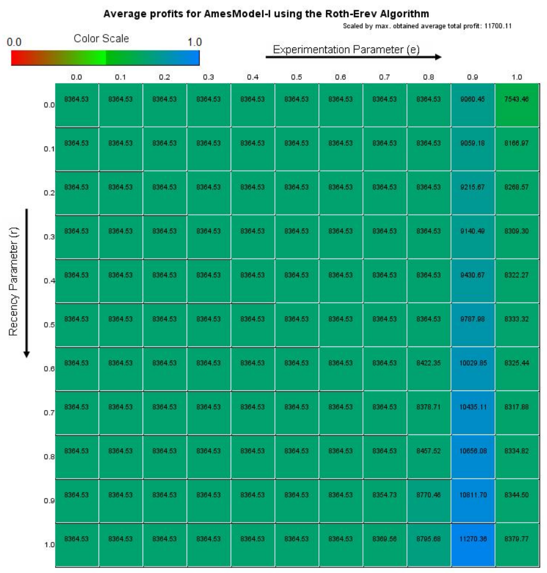
(a) Roth-Erev Algorithm



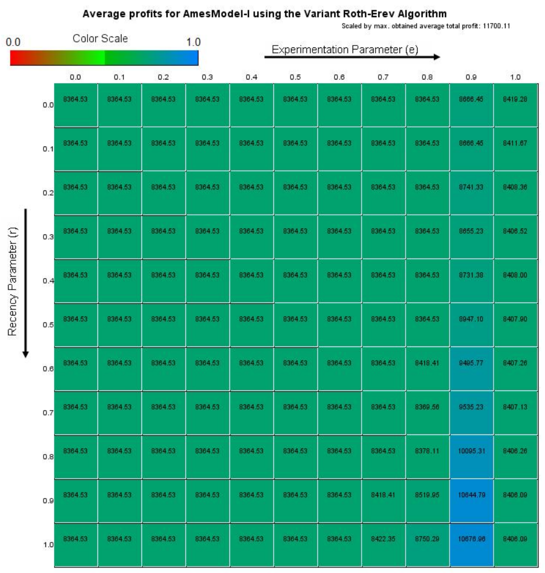
(b) Variant Roth-Erev Algorithm

Figure 4.26 Average Time Period for the Convergent Action choice probability reaching 0.999 for generator G5 in AMESModel-I: Experiment 1 (High Initial Propensity)

4.3.2 Experiment 2: Low initial propensity



(a) Roth-Erev Algorithm



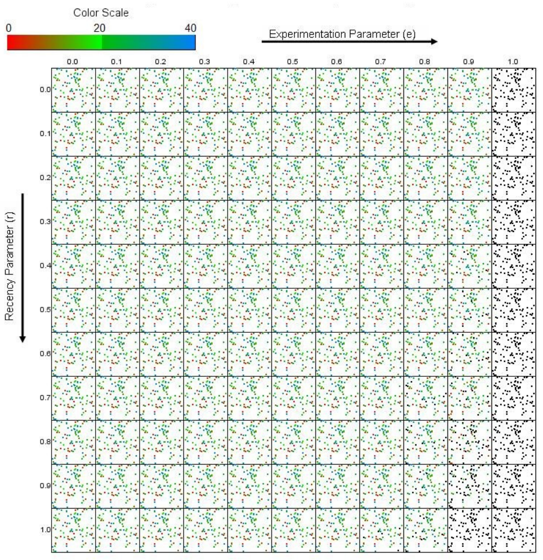
(b) Variant Roth-Erev Algorithm

Figure 4.27 Scaled Average Total Profits for generator G5 in AMESModel-I: Experiment 2 (Low Initial Propensity)

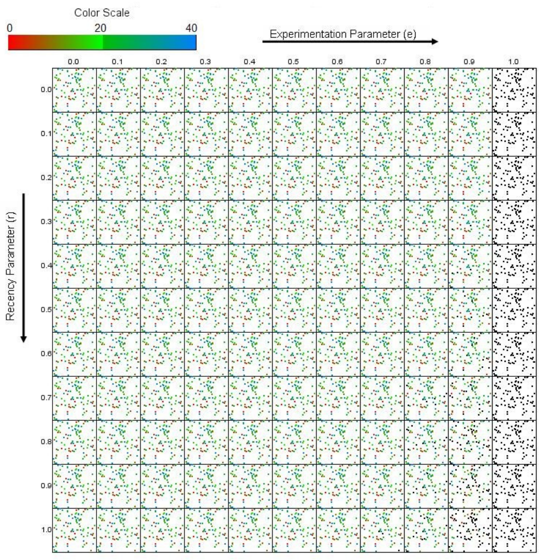
The profits shown in the cells of the heat map are scaled by a factor of 10^3.

The Average Total Profits for generator G5 across practically the entire parameter range are relatively constant. It is only with a large value of recency and experimentation parameters that the profits are higher. For almost the entirety of the parameter space, the recency and experimentation parameters are such that even an attenuated reward overwhelms the low initial propensity. The propensity of the action chosen at time $t = 0$ is approximately a 1000 times more than the propensity of the other actions at time $t = 1$. Due to this high action propensity, the probability of choosing that action in subsequent time steps immediately becomes greater than 0.999. This experiment shows the problem of 'early fixation'. Since the action that the agent fixates to is a random selection at time $t = 0$, the learner can be considered to be a 'zero-intelligence' learner.

For the small region where the recency and experimentation parameters are large, the experimentation parameter attenuates the reward in the update and spillover functions to such an extent that the remaining actions remain competitive compared to the action chosen. Hence, there is greater experimentation and a greater possibility of higher profits.



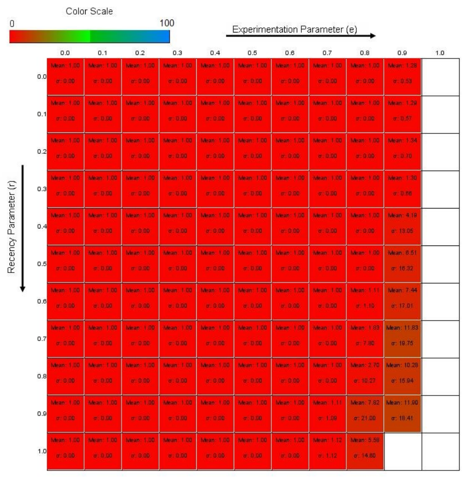
(a) Roth-Erev Algorithm



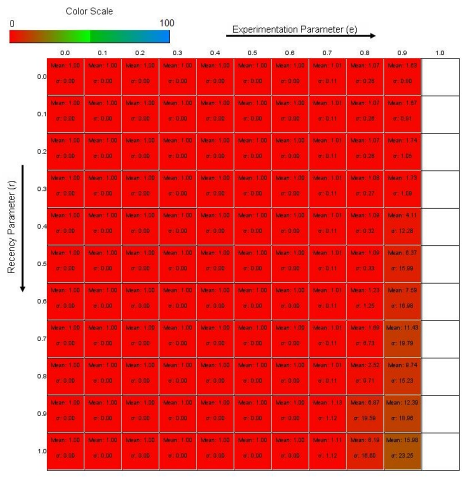
(b) Variant Roth-Erev Algorithm

Figure 4.28 Convergent Action for generator G5 in AMESModel-I: Experiment 2 (Low Initial Propensity)

The action to which the learning agent converges is essentially a random guess from the action domain, hence the Convergent Action in the simulation runs is distributed evenly across the action domain.



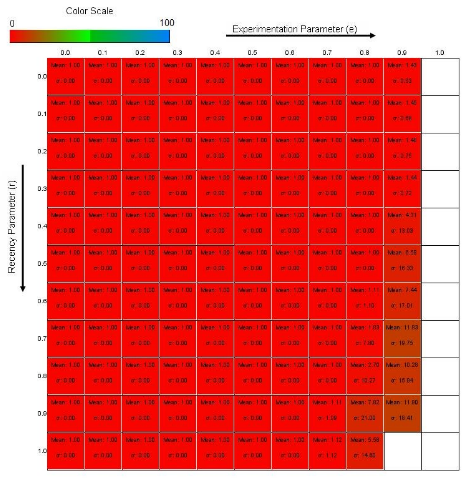
(a) Roth-Erev Algorithm



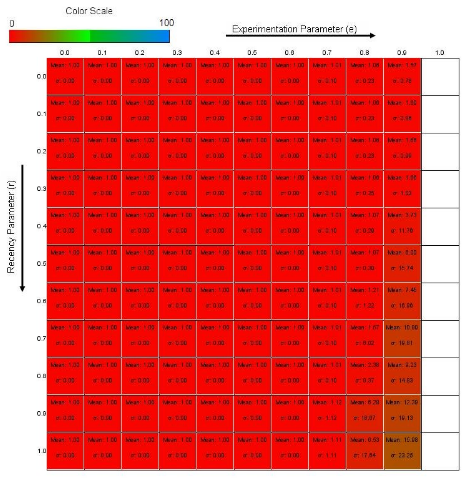
(b) Variant Roth-Erev Algorithm

Figure 4.29 Average Time Period for the Convergent Action choice probability reaching 0.6 for generator G5 in AMESModel-I: Experiment 2 (Low Initial Propensity)

The simulation convergences to a single action at the first tick. This is true for nearly the entire parameter space.

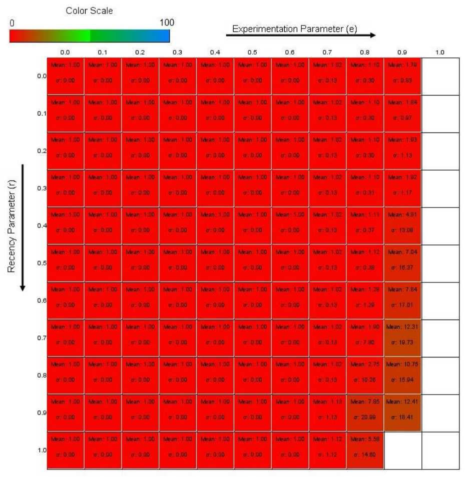


(a) Roth-Erev Algorithm

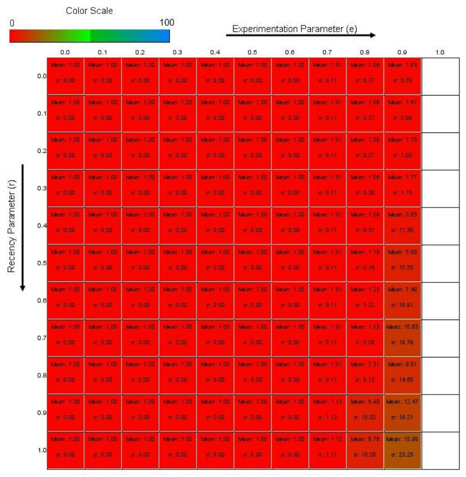


(b) Variant Roth-Erev Algorithm

Figure 4.30 Average Time Period for the Convergent Action choice probability reaching 0.9 for generator G5 in AMESModel-I: Experiment 2 (Low Initial Propensity)



(a) Roth-Erev Algorithm

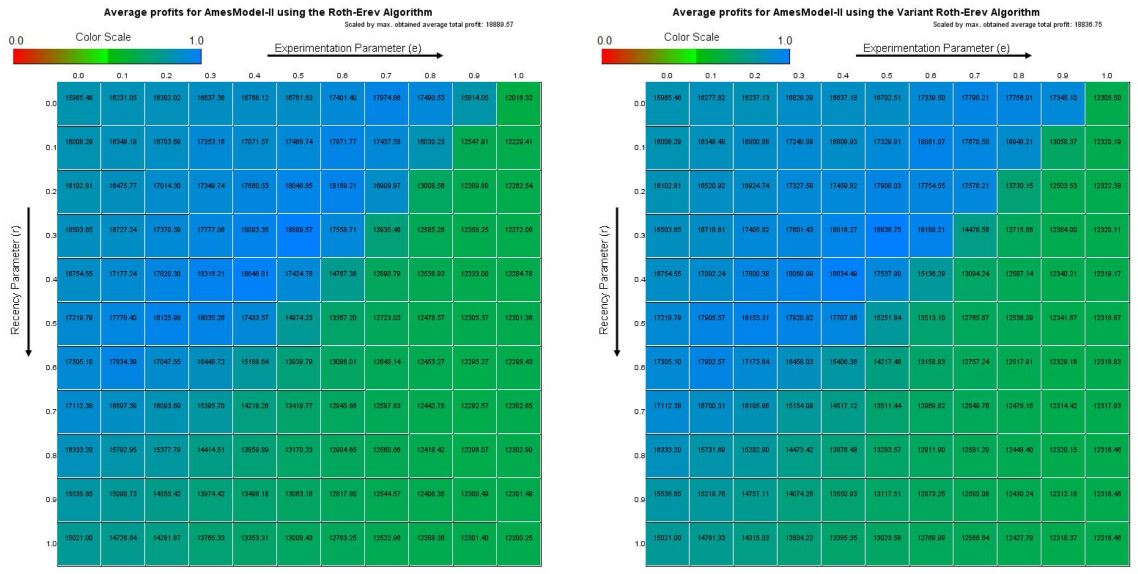


(b) Variant Roth-Erev Algorithm

Figure 4.31 Average Time Period for the Convergent Action choice probability reaching 0.999 for generator G5 in AMESModel-I: Experiment 2 (Low Initial Propensity)

4.4 AMESModel-II: Two Learning Agents

4.4.1 Experiment 1: High initial propensity



(a) Roth-Erev Algorithm

(b) Variant Roth-Erev Algorithm

Figure 4.32 Scaled Average Total Profits for generator G5 in AMESModel-II: Experiment 1 (High Initial Propensity)

The profits shown in the cells of the heat map are scaled by a factor of 10^3.

The pattern of the graph is very similar to that in AMESModel-I. The top right corner corresponds to the region where the convergence occurs too rapidly for learning. The region with the maximum rewards is where the rate of convergence is not too low and the propensities do not become too low with respect to the cooling parameter.

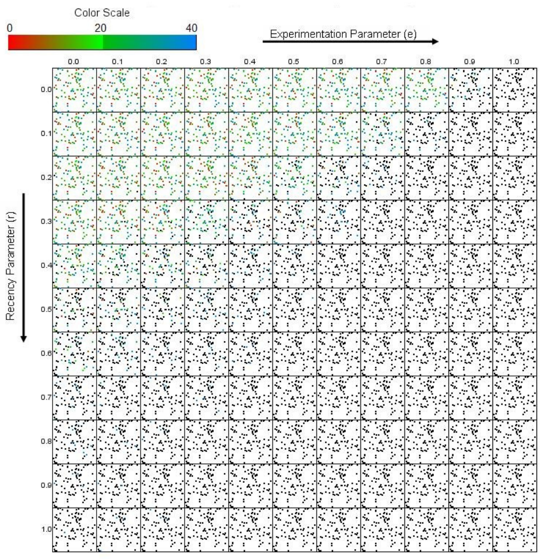
Since there are now two learning generators able to adapt, generator G5 begins to appear more competitive. Also, the locational marginal prices (LMPs) are higher now that there are two generators that are bidding higher than their true costs. [2]

The band for maximum profits for generator G5 has moved toward the bottom right corner. This could be because the rewards for generator G5 are higher as compared to AMESModel-I.

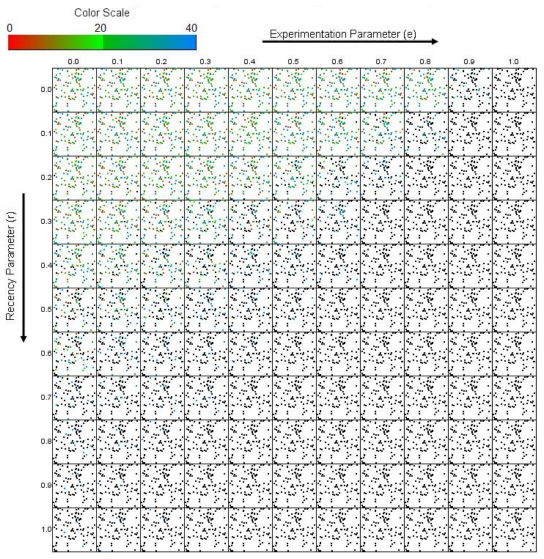
[2]Refer to appendix D.2.1 for a full description of the AMES Market package

A higher experimentation and recency settings allow the algorithm to settle into a subset of actions that provide larger profits for generator G5.

The Roth-Erev and VRE RL experiments are again comparable for the same reasons as in AMESModel-I.



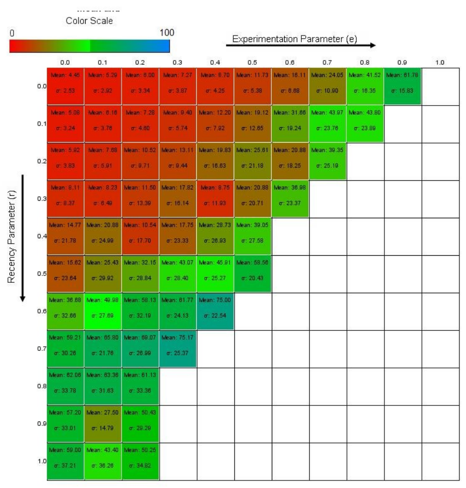
(a) Roth-Erev Algorithm



(b) Variant Roth-Erev Algorithm

Figure 4.33 Convergent Action for generator G5 in AMESModel-II: Experiment 1 (High Initial Propensity)

Once again, the patterns observed are very similar to that in AMESModel-I. In the top left corner, the generator G5 converges too quickly and so the simulations show a wide variation in the Convergent Actions. In the band with the maximum profits, the simulations appear to converge to a subset of the actions. In the remaining area of the graph, no single action reaches a probability greater than 0.999 in the specified time period. The explanation for this is similar to AMESModel-I.


(a) Roth-Erev Algorithm

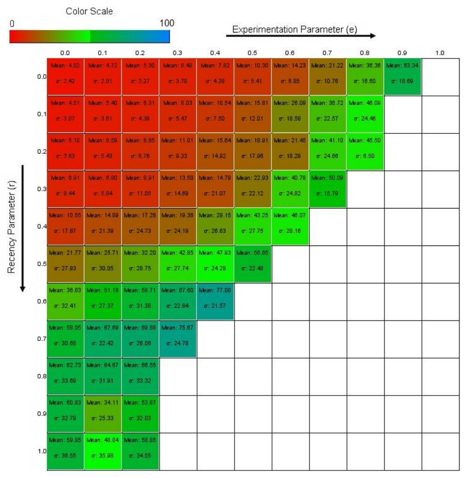
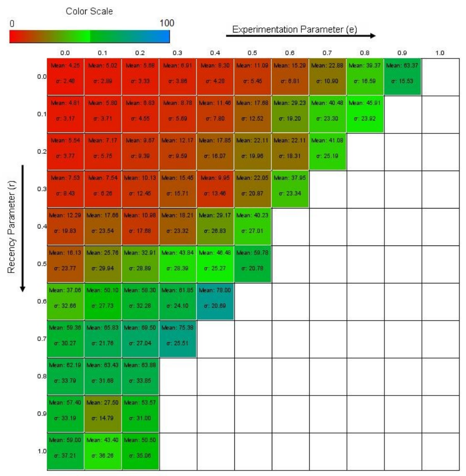
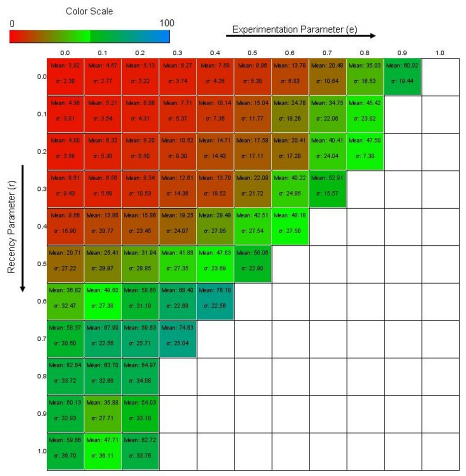

(b) Variant Roth-Erev Algorithm

Figure 4.34 Average Time Period for the Convergent Action choice probability reaching 0.6 for generator G5 in AMESModel-II: Experiment 1 (High Initial Propensity)

The convergence graph is very similar to that of AMESModel-I except that the band with the largest profit has moved closer to the bottom right corner. The area with maximum profits have a higher convergence time period as compared to AMESModel-I. This could be because the rewards are larger as compared to the initial propensity. The agent requires a larger experimentation and recency to be able to settle into an action that gives a larger payoff.

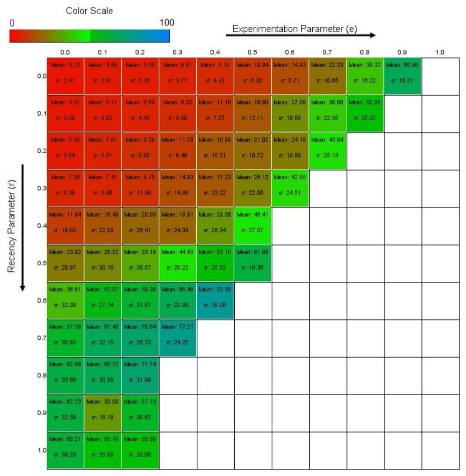


(a) Roth-Erev Algorithm

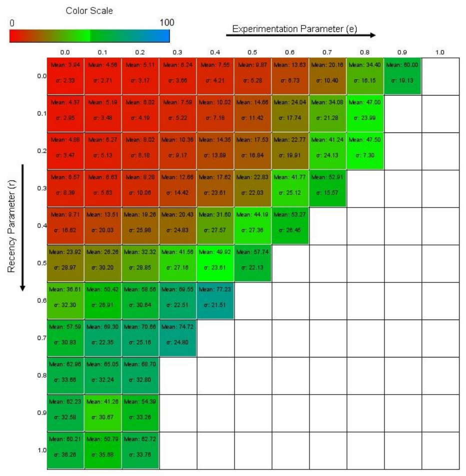


(b) Variant Roth-Erev Algorithm

Figure 4.35 Average Time Period for the Convergent Action choice probability reaching 0.9 for generator G5 in AMESModel-II: Experiment 1 (High Initial Propensity)



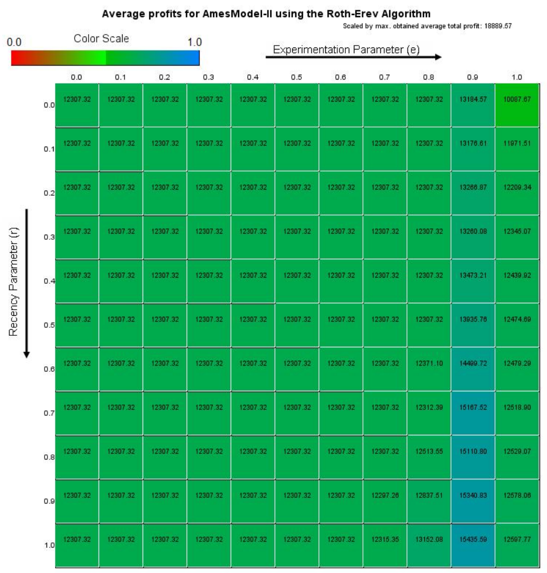
(a) Roth-Erev Algorithm



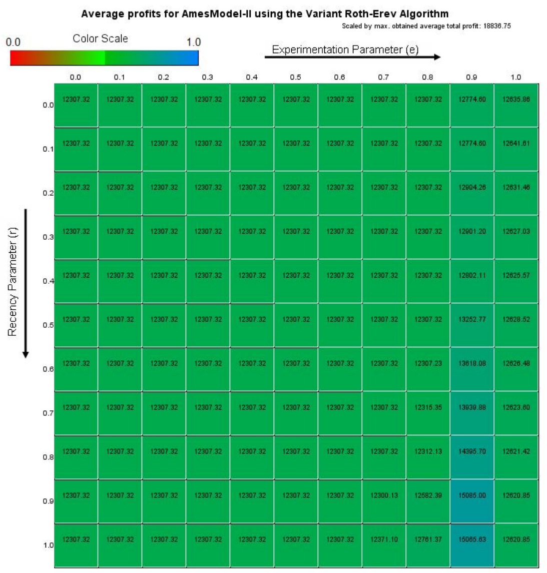
(b) Variant Roth-Erev Algorithm

Figure 4.36 Average Time Period for the Convergent Action choice probability reaching 0.999 for generator G5 in AMESModel-II: Experiment 1 (High Initial Propensity)

4.4.2 Experiment 2: Low initial propensity



(a) Roth-Erev Algorithm

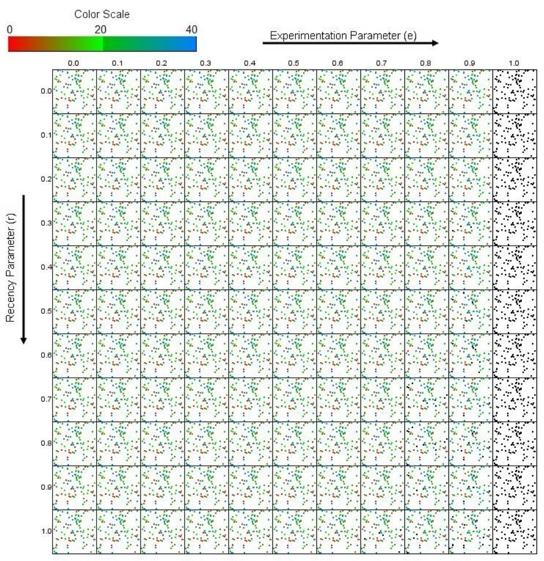


(b) Variant Roth-Erev Algorithm

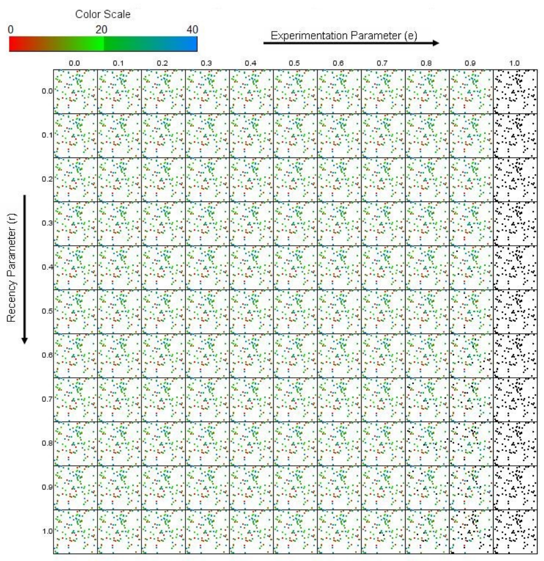
Figure 4.37 Scaled Average Total Profits for generator G5 in AMESModel-II: Experiment 2 (Low Initial Propensity)

The profits shown in the cells of the heat map are scaled by a factor of 10^3.

As above, this experiment corresponds to the 'zero-intelligence' learning agent. This is once again, an example of 'early fixation'.



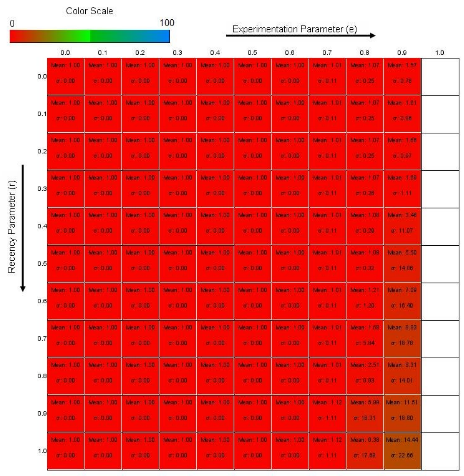
(a) Roth-Erev Algorithm



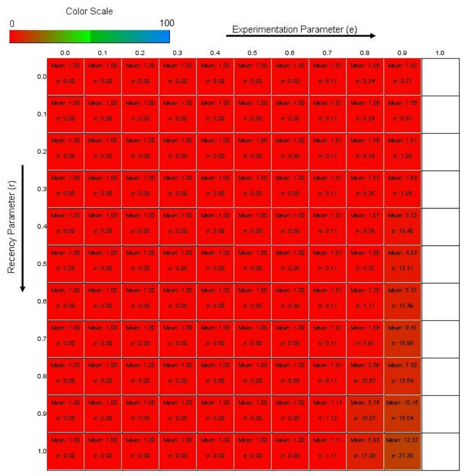
(b) Variant Roth-Erev Algorithm

Figure 4.38 Convergent Action for generator G5 in AMESModel-II: Experiment 2 (Low Initial Propensity)

As expected, the actions vary widely because a random action is chosen for a given random seed and that action converges to a very high probability at the first tick itself.



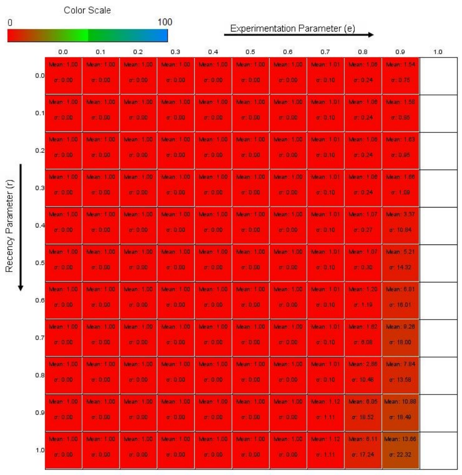
(a) Roth-Erev Algorithm



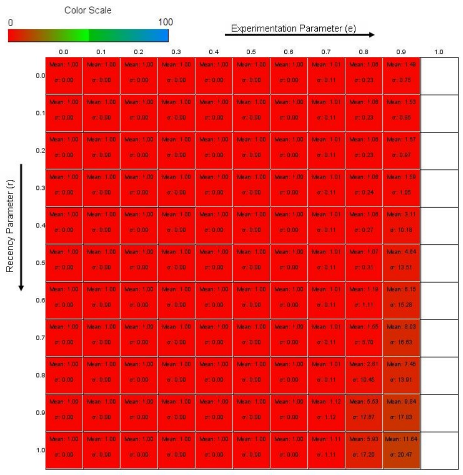
(b) Variant Roth-Erev Algorithm

Figure 4.39 Average Time Period for the Convergent Action choice probability reaching 0.6 for generator G5 in AMESModel-II: Experiment 2 (Low Initial Propensity)

The actions again converge too rapidly to be able to learn anything. The simulation converges to the random action selected on the first day of the simulation run. There is therefore no learning involved.

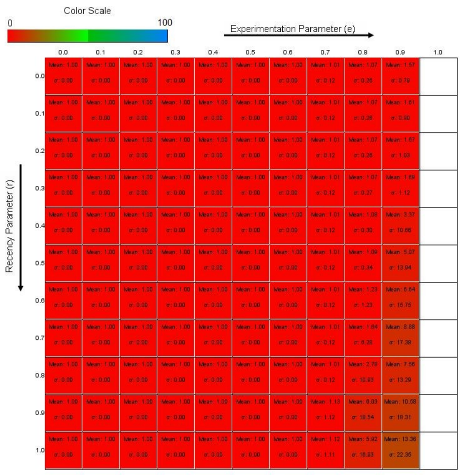


(a) Roth-Erev Algorithm

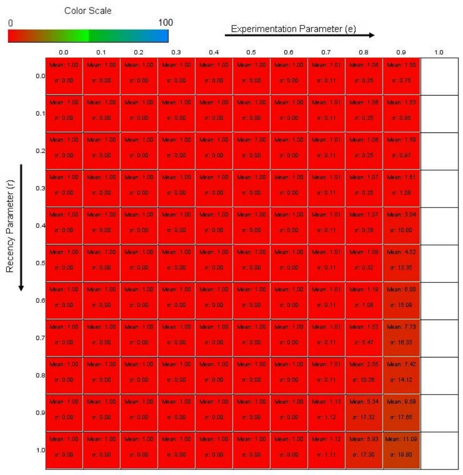


(b) Variant Roth-Erev Algorithm

Figure 4.40 Average Time Period for the Convergent Action choice probability reaching 0.9 for generator G5 in AMESModel-II: Experiment 2 (Low Initial Propensity)



(a) Roth-Erev Algorithm

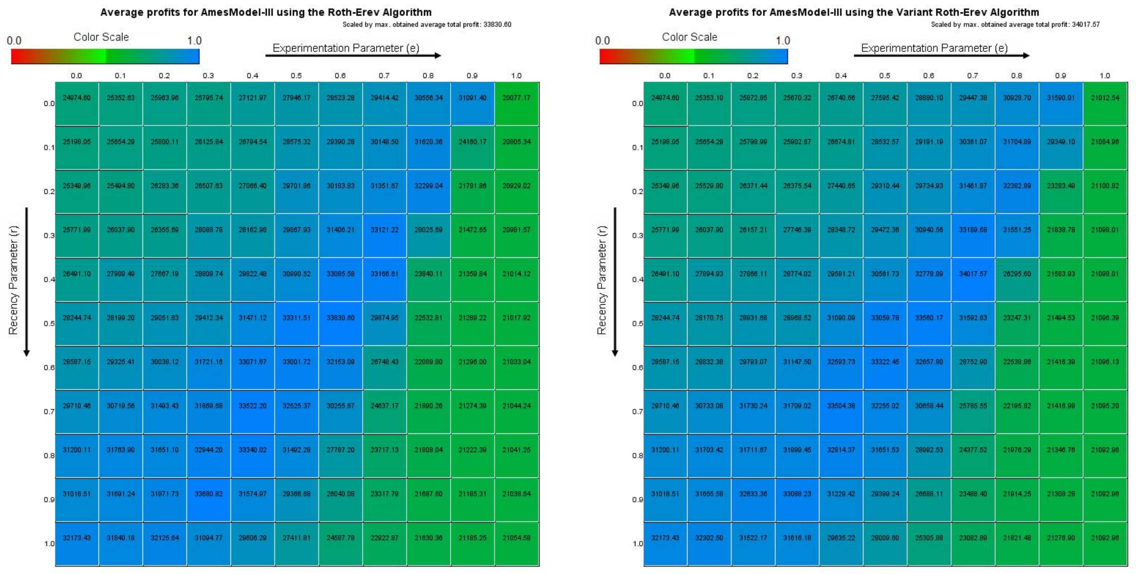


(b) Variant Roth-Erev Algorithm

Figure 4.41 Average Time Period for the Convergent Action choice probability reaching 0.999 for generator G5 in AMESModel-II: Experiment 2 (Low Initial Propensity)

4.5 AMESModel-III: Five Learning Agents

4.5.1 Experiment 1: High initial propensity

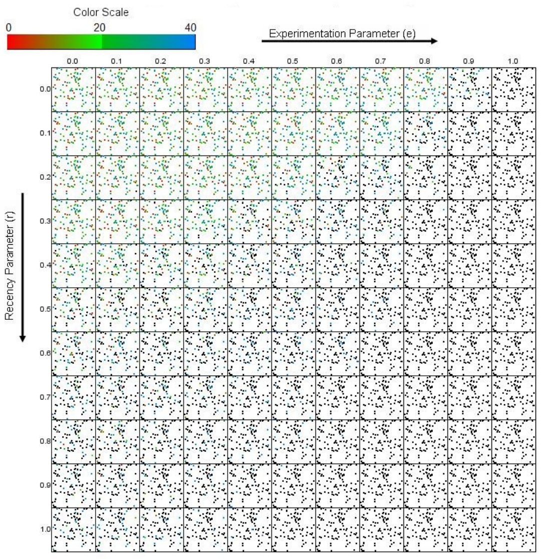


(a) Roth-Erev Algorithm (b) Variant Roth-Erev Algorithm

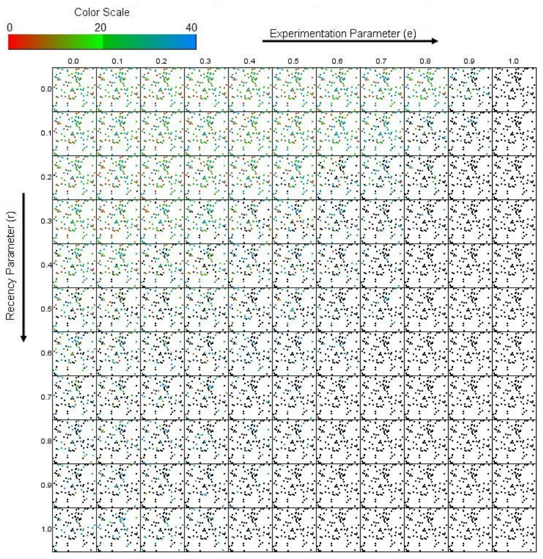
Figure 4.42 Scaled Average Total Profits for generator G5 in AMESModel-III: Experiment 1 (High Initial Propensity)

The profits shown in the cells of the heat map are scaled by a factor of 10^3.

The explanation of the graph is very similar to that in AMESModel-I. The graph has shifted further toward the bottom-right corner. This could be because the rewards are much higher and hence, action propensities in AMESModel-III will tend to be higher as compared to the action propensities in AMESModel-I for a given time period. We can also expect greater variation in the actions chosen in the early stages. This is similar to the results in [12], where higher initial propensity encourages greater experimentation.



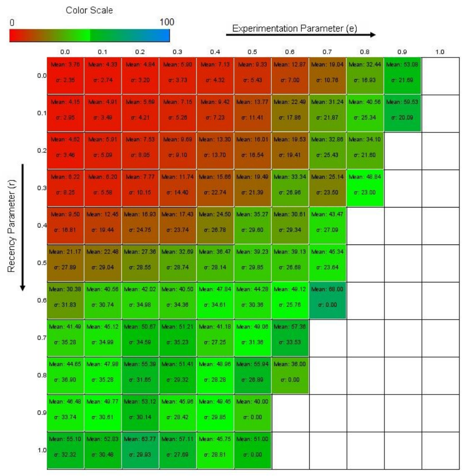
(a) Roth-Erev Algorithm

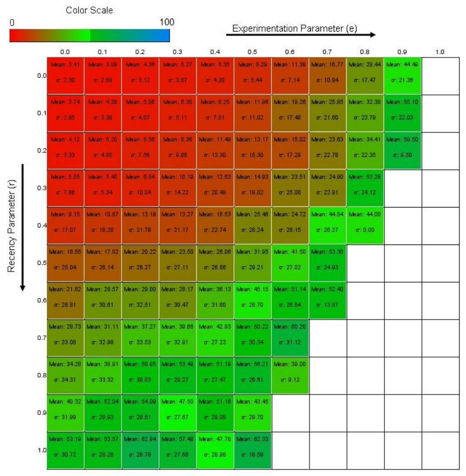


(b) Variant Roth-Erev Algorithm

Figure 4.43 Convergent Action for generator G5 in AMESModel-III: Experiment 1 (High Initial Propensity)

In the maximum profit zone, most of the simulations tend to converge to a subset of actions that obtain a larger profit.

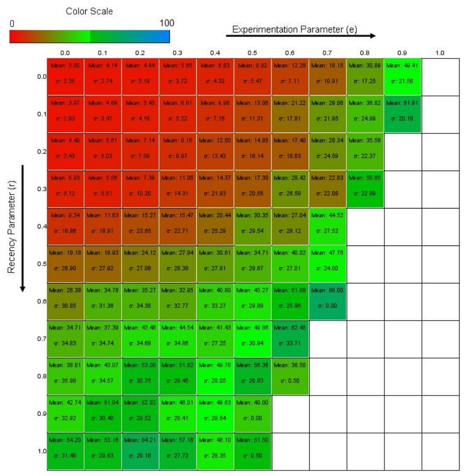




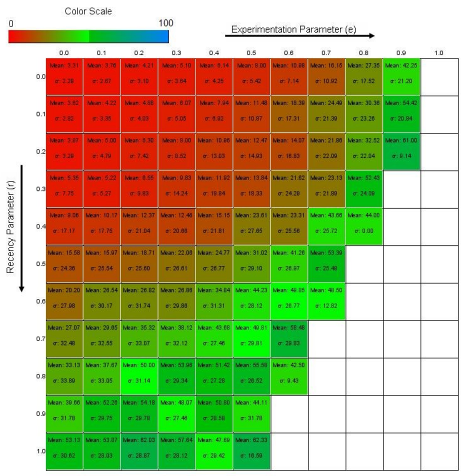
(a) Roth-Erev Algorithm (b) Variant Roth-Erev Algorithm

Figure 4.44 Average Time Period for the Convergent Action choice probability reaching 0.6 for generator G5 in AMESModel-III: Experiment 1 (High Initial Propensity)

Since the average daily profits are much higher than in the case of AMESModel-I and AMESModel-II, the slower rate of reduction in the initial propensity causes larger experimentation in choosing different actions and a longer convergence time. The average convergence is about 42 days as compared to around 25 days for AMESModel-I and AMESModel-II.

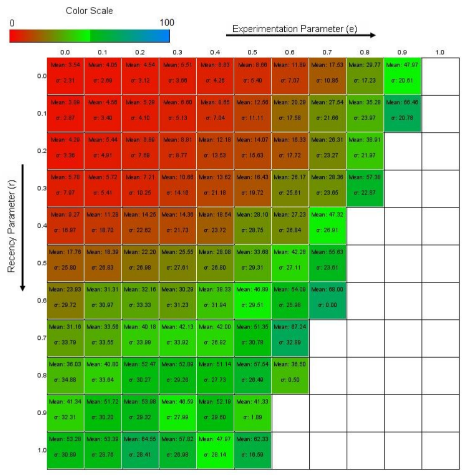


(a) Roth-Erev Algorithm

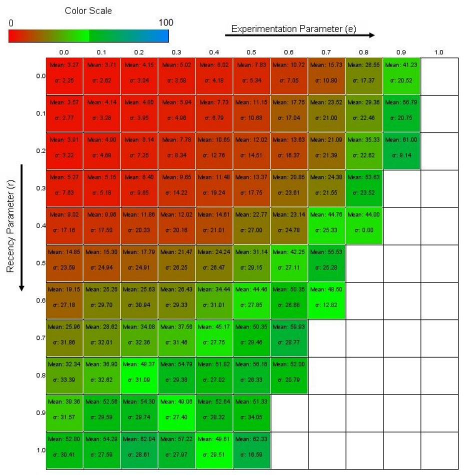


(b) Variant Roth-Erev Algorithm

Figure 4.45 Average Time Period for the Convergent Action choice probability reaching 0.9 for generator G5 in AMESModel-III: Experiment 1 (High Initial Propensity)



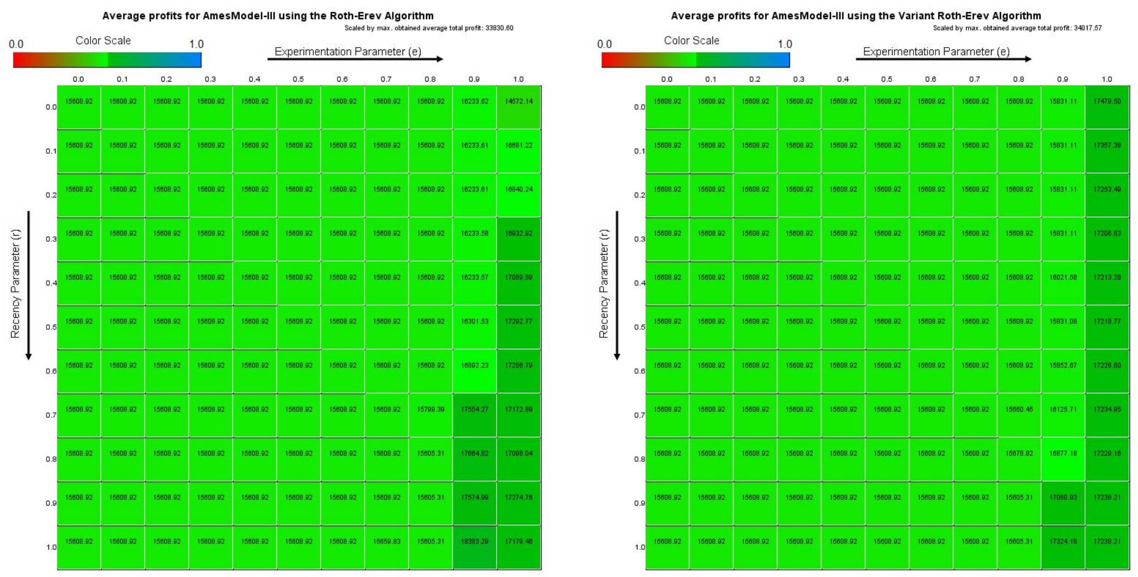
(a) Roth-Erev Algorithm



(b) Variant Roth-Erev Algorithm

Figure 4.46 Average Time Period for the Convergent Action choice probability reaching 0.999 for generator G5 in AMESModel-III: Experiment 1 (High Initial Propensity)

4.5.2 Experiment 2: Low initial propensity

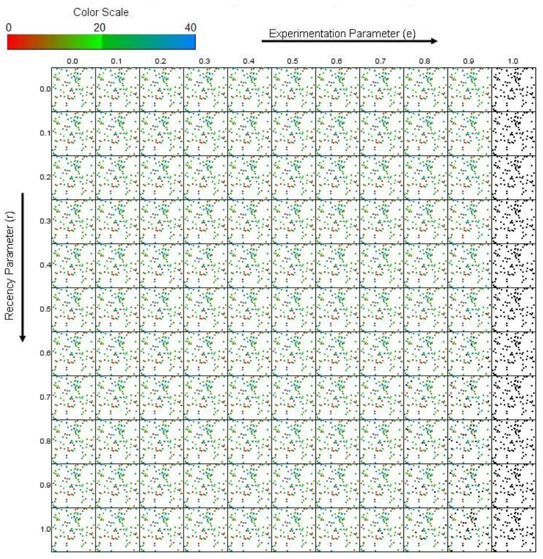


(a) Roth-Erev Algorithm (b) Variant Roth-Erev Algorithm

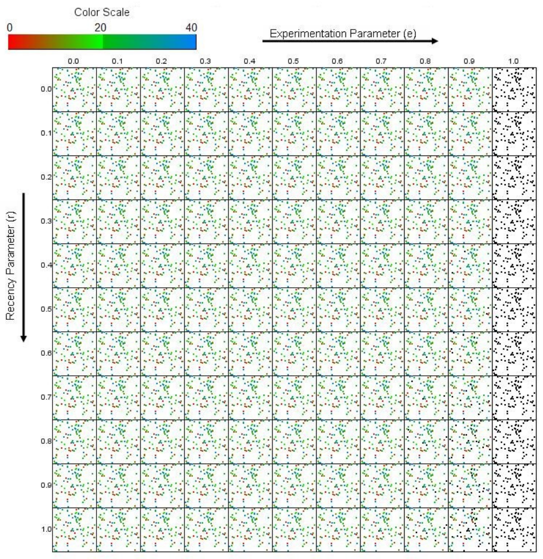
Figure 4.47 Scaled Average Total Profits for generator G5 in AMESModel-III: Experiment 2 (Low Initial Propensity)

The profits shown in the cells of the heat map are scaled by a factor of 10^3.

Once again, the profits shown can be considered as the profits obtained for a 'zero intelligence' learning agent. The agent converges to an action in the first time period itself. Since the other agents are choosing among forty actions, all of which are equal to higher than their default non-learning price, the LMP values tend to be much higher than in AMESModel-I and II. The learning generator (G5) has almost the same commitment as in AMESModel-I and II, but since the LMPs are much higher, the profits obtained are also comparatively higher.

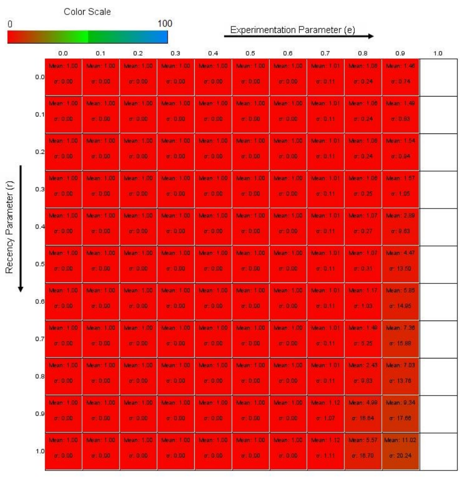


(a) Roth-Erev Algorithm

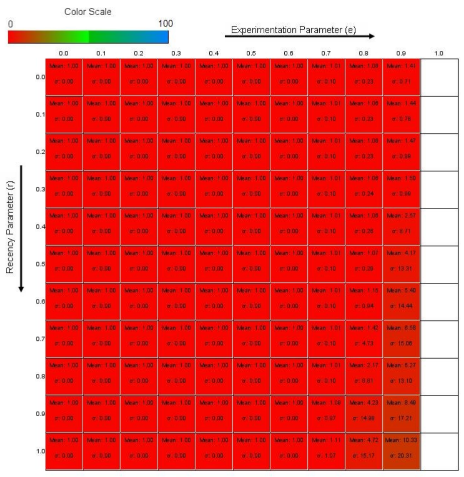


(b) Variant Roth-Erev Algorithm

Figure 4.48 Convergent Action for generator G5 in AMESModel-III: Experiment 2 (Low Initial Propensity)

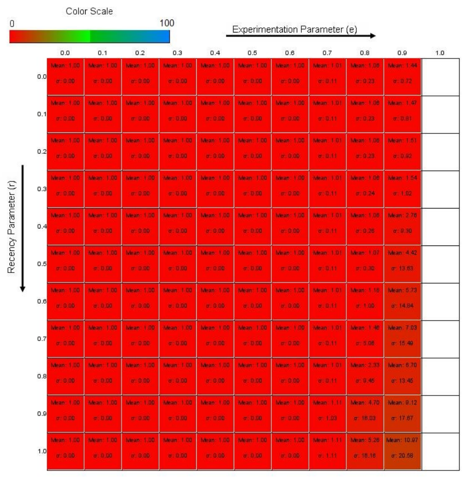


(a) Roth-Erev Algorithm

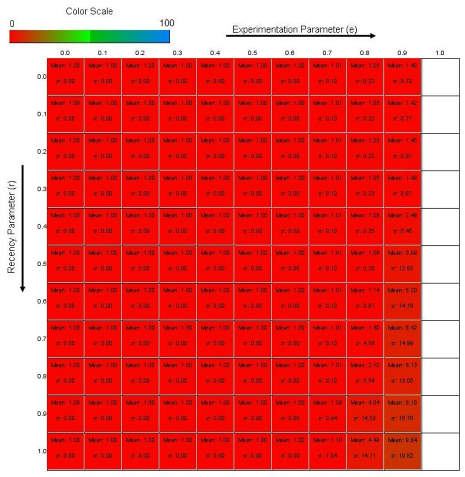


(b) Variant Roth-Erev Algorithm

Figure 4.49 Average Time Period for the Convergent Action choice probability reaching 0.6 for generator G5 in AMESModel-III: Experiment 2 (Low Initial Propensity)

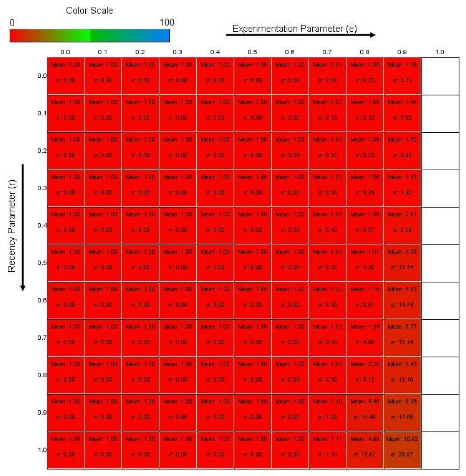


(a) Roth-Erev Algorithm

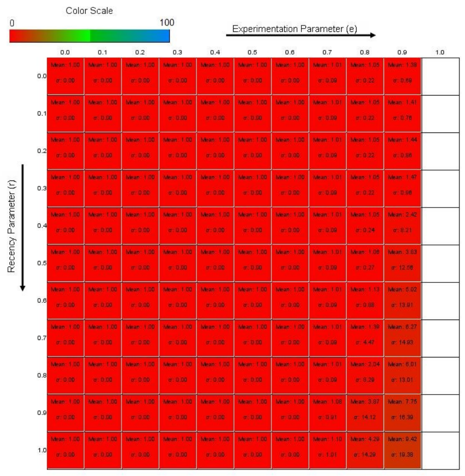


(b) Variant Roth-Erev Algorithm

Figure 4.50 Average Time Period for the Convergent Action choice probability reaching 0.9 for generator G5 in AMESModel-III: Experiment 2 (Low Initial Propensity)



(a) Roth-Erev Algorithm



(b) Variant Roth-Erev Algorithm

Figure 4.51 Average Time Period for the Convergent Action choice probability reaching 0.999 for generator G5 in AMESModel-III: Experiment 2 (Low Initial Propensity)

Table 4.1 Best parameter settings from the AMESModel experiments

Experiment	**Recency** (r)	**Experim.** (e)	**Average Total Profits** $(\times 10^5)$	**Ticks for** $p_b > 0.6$	**Ticks for** $p_b > 0.9$	**Ticks for** $p_b > 0.999$
RE and AMESModel-I	0.5	0.0	117.0	$\times$	$\times$	$\times$
	0.4	0.1	116.4	25.93	26.51	26.51
	0.4	0.0	115.4	19.55	19.80	19.80
VRE and AMESModel-I	0.5	0.0	117.0	$\times$	$\times$	$\times$
	0.4	0.1	116.2	27.03	27.59	27.59
	0.4	0.0	115.4	19.72	19.80	19.80
RE and AMESModel-II	0.3	0.5	188.9	20.88	22.05	25.12
	0.4	0.4	186.4	28.73	29.17	29.56
	0.4	0.3	183.1	17.75	18.21	19.91
VRE and AMESModel-II	0.3	0.5	188.3	22.93	22.09	22.83
	0.4	0.4	186.3	29.15	29.49	31.60
	0.3	0.6	181.8	40.78	40.22	41.77
RE and AMESModel-III	0.5	0.6	338.4	39.13	40.02	42.28
	0.9	0.3	336.8	45.96	51.62	46.59
	0.7	0.4	335.2	41.18	41.43	42.00
VRE and AMESModel-III	0.4	0.7	340.0	44.84	43.66	44.76
	0.5	0.6	335.6	41.50	41.26	42.25
	0.7	0.4	335.0	42.93	43.68	45.17

CHAPTER 5. MATHEMATICAL ANALYSIS OF THE ALGORITHMS

This section provides some mathematical analysis for both the original Roth-Erev reinforcement learning (RL) algorithm and the modified Roth-Erev (MRE) RL algorithm within the context of the SimpleModel-I test case introduced and discussed in earlier sections.

Recall that the SimpleModel-I test case has a single learning agent with $N \geq 2$ possible action choices denoted by $j = 0, 1, \ldots, N-1$. One of these action choices, denoted by 0, always results in a particular positive reward π_0. All other action choices j always result in zero rewards. However, the learning agent does not know this pattern of rewards at the initial time 0. The key issue of interest is whether the Roth-Erev or MRE reinforcement learning algorithms permit the learning agent to converge to a choice of best action 0 and, if so, how rapidly does this convergence occur.

The Roth-Erev RL algorithm equations are given in 2.1 and the MRE RL algorithm equations are given in 2.4. Throughout this section it will be assumed that the value e for the experimentation parameter lies strictly between 0 and 1, the value r for the recency parameter lies strictly between 0 and 1 and the initial choice propensity $q_j(0)$ for each of the learning agent's action choices j is strictly positive.

5.1 Roth-Erev RL Algorithm

5.1.1 Positive Propensity Values

Under the maintained parameter restrictions given above for r, e, the initial choice propensity values $q_j(0)$ and the reward π_0 for the best action choice, the Roth-Erev learning agent's choice propensities $q_j(t)$ are always positive. To see this, first recall that all the initial choice propensities are positive by assumption. Suppose the best action 0 is chosen at time 1. Then, for the best action 0,

$$q_0(1) \ = \ (1-r) \times q_0(0) + (1-e) \times \pi_0 \ > \ 0$$

and for any other non-chosen action j,

$$q_j(1) \ = \ (1-r) \times q_j(0) + \pi_0 \times e/[N-1] \ > \ 0$$

Suppose, instead, that action j is chosen with $j \neq 0$. Then for action j,

$$q_j(1) \ = \ (1-r) \times q_j(0) + (1-e) \times 0 \ > \ 0$$

and for action 0,

$$q_0(1) \ = \ (1-r) \times q_0(0) + 0 \times e/[N-1] \ > \ 0$$

It follows by a simple induction argument that all of the learning agent's choice propensities are positively valued at all times $t = 1, 2, \ldots$.. The corollary is that the probability for any action can never be strictly zero.

$$p_j(t) > 0 \ , \ 0 \leq j < N, t = 0, 1, \ldots \tag{5.1}$$

5.1.2 Experimentation limits for learning

As earlier remarked by [11], if the value e of the experimentation parameter is set to $[N-1]/N$ in the Roth-Erev RL algorithm, the choice propensities at each time t will be exactly the same as at time $(t+1)$. In other words, no learning will occur from one time period to the next. If e is greater than $[N-1]/N$, the choice probability for the best action 0 will actually decrease each time it is selected and a positive reward is obtained, thus tending to push this choice probability to zero.

5.1.3 Limiting values of probabilities

From the computational experiments conducted with SimpleModel-I, it appears that the choice propensity for the best action 0 converges over time. Here we analyze the Roth-Erev RL algorithm to see whether this computational finding can be given any mathematical support.

Consider the case in which, for all sufficiently large t, the learning agent always chooses the best action 0. Without loss of generality, it can be assumed that this persistent choice of action 0 starts in time period 1. Following the choice of action 0 at time 1, the initial choice propensity $q_0(0)$ for action 0 is updated as follows:

$$q_o(1) = (1-r)q_0(0) + \pi_0(1-e)$$

At time $t = 2$, this becomes

$$q_0(2) = (1-r)\Big((1-r)q_0(0) + \pi_0(1-e)\Big) + \pi_0(1-e)$$

At a general time t the formula is

$$q_0(t) = (1-r)^t q_0(0) + \Big((1-r)^{t-1} + (1-r)^{t-2} + \ldots + (1-r)^0\Big)\pi_0(1-e)$$

As time $t \to \infty$, this becomes

$$q_0(\infty) = \frac{1}{1-(1-r)}\pi_0(1-e)$$

or

$$q_0(\infty) = \frac{\pi_0(1-e)}{r} \tag{5.2}$$

This implies that there exists a positive limiting value for the choice propensity of the best action if it is persistently chosen.

Now consider what happens to the choice propensities $q_j(t)$ of any other action j while the best action 0 is persistently being chosen starting at time t=1.

$$q_j(1) = (1-r)q_j(0) + \pi_0 \times \frac{e}{N-1}$$

.

Similarly, at time $t = 2$,

$$q_j(2) = (1-r)\Big((1-r)q_j(0) + \pi_0(0) \times \frac{e}{N-1}\Big) + \pi_0(0) \times \frac{e}{N-1}$$

At a general time t, then,

$$q_j(t) = (1-r)^t q_j(0) + \sum_{i=0}^{t-1} \pi_0 \times \frac{e}{N-1} \times (1-r)^i$$

As $t \to \infty$,

$$q_j(\infty) = \pi_0 \times \frac{e}{N-1} \times \frac{1}{r} \tag{5.3}$$

It follows from equations (5.2) and (5.3) that, if the best action 0 is persistently chosen for all sufficiently large t, then the choice propensities for the learning agent for its N possible actions converge to well-defined limits. This implies that the choice probabilities for the learning agent also converge to well-defined limits. Specifically, for the best action 0,

$$\begin{aligned} p_0(\infty) &= \frac{q_0(\infty)}{\displaystyle\sum_{i=0}^{N-1} q_i} \\ &= \frac{\frac{\pi_0(1-e)}{r}}{\sum_{i=1}^{N-1} \frac{\pi_0(e)}{(N-1)(r)} + \frac{\pi_0(1-e)}{r}} \\ &= (1-e) \end{aligned}$$

Or more succinctly,

$$p_0(\infty) = (1-e) \tag{5.4}$$

Similarly, the choice probability for each of the $N-1$ non-best actions j converges to

$$p_j(\infty) = \frac{e}{N-1} \tag{5.5}$$

Now suppose that some action j *other* than the best action 0 is also chosen infinitely often. Each time that this action j is chosen, the only effect is to uniformly shrink *all* propensity values by a factor of $[1 - r]$, so that *all* choice probabilities are unaffected by this choice. It follows that (5.4) and (5.5) are the limits for the choice probabilities as long as the best action 0 is chosen infinitely often, whether or not any other action j is chosen infinitely often as well.

For example, if $e = 0.8$ and N=4 and if the best action 0 is persistently chosen for all sufficiently large t, then the choice probability for the best action converges to 0.2 and the choice probability for each of the three non-best actions converge to 0.8/3. This is seen in figure 5.1, which depicts a sample run from SimpleModel-I assuming the learning agent uses the Roth-Erev RL algorithm with the following parameter settings:

- initial choice propensity values $q_j(0)$: 6000.0
- Number of actions N: 4
- experimentation e: 0.8
- recency r: 0.9

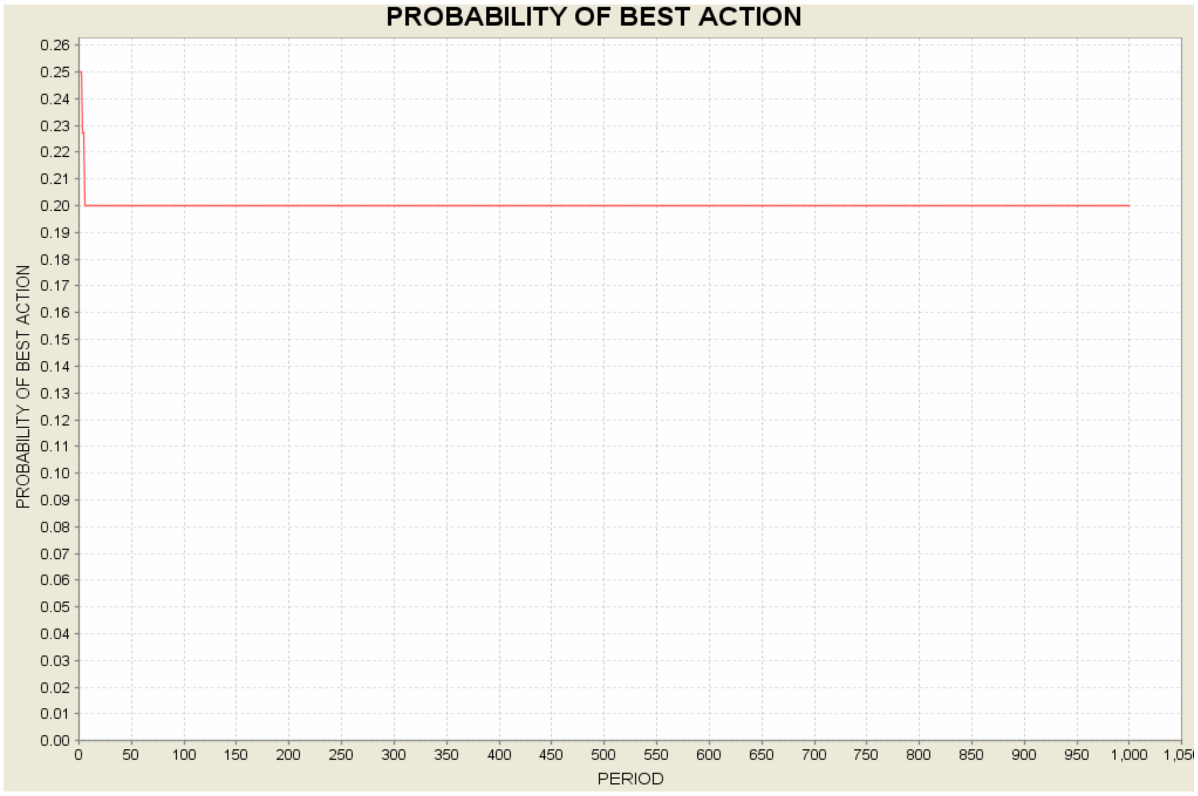


Figure 5.1 Choice Probability of the profitable action for the learning seller in the SimpleModel-I experiment using the Roth-Erev RL algorithm (I)

5.2 Modified Roth-Erev RL Algorithm

5.2.1 Positive Propensity Values

Under the maintained parameter restrictions given above for r, e, the initial choice propensity values $q_j(0)$ and the reward π_0 for the best action choice, the MRE learning agent's choice propensities $q_j(t)$ are always positive. To see this, first recall that all the initial choice propensities are positive by assumption. Suppose the best action 0 is chosen at time 1. Then, for the best action 0,

$$q_0(1) \;=\; (1-r)\times q_0(0) + (1-e)\times \pi_0 \;>\; 0$$

and for any other non-chosen action j,

$$q_j(1) \;=\; (1-r)\times q_j(0) + e/[N-1]\times q_j(0) \;>\; 0$$

Suppose, instead, that action j is chosen with $j \neq 0$. Then for action j,

$$q_j(1) \;=\; (1-r)\times q_j(0) + (1-e)\times 0 \;>\; 0$$

and for action 0,

$$q_0(1) \;=\; (1-r)\times q_0(0) + e/[N-1]\times q_0(0) \;>\; 0$$

It follows by a simple induction argument that all of the learning agent's choice propensities are positively valued at all times $t = 1, 2, \ldots$.

5.2.2 Necessary Conditions for Best Action Choice Probability to Increase

For the MRE RL algorithm, in order for the choice probability for the best action 0 to increase from period t to t+1, the following has to hold:

$$\frac{q_0(t+1)}{\sum_{i=0}^{N-1} q_i(t+1)} \geq \frac{q_0(t)}{\sum_{i=0}^{N-1} q_i(t)}$$

$\therefore$

$$\frac{(1-r)q_0(t)+[1-e]\pi_0}{(1-r)q_0(t)+[1-e]\pi_0+\sum_{i=1}^{N-1}[(1-r)q_i(t)+\frac{eq_i(t)}{N-1}]} \geq \frac{q_0(t)}{\sum_{i=0}^{N-1} q_i(t)}$$

$\therefore$

$$\Big((1-r)q_0(t)+[1-e]\pi_0\Big)\times\sum_{i=0}^{N-1}[q_i(t)] \geq q_0(t)\times\Big((1-r)q_0(t)+[1-e]\pi_0+\sum_{i=1}^{N-1}[(1-r)q_i(t)+\frac{eq_i(t)}{N-1}]\Big)$$

$\therefore$

$$\Big((1-r)q_0(t)+[1-e]\pi_0\Big)\times q_0(t)\;+\;\Big((1-r)q_0(t)+[1-e]\pi_0\Big)\times\sum_{i=1}^{N-1}[q_i(t)]\geq$$

$$q_0(t)\times\Big((1-r)q_0(t)+[1-e]\pi_0\Big)+q_0(t)\times\Big(\sum_{i=1}^{N-1}[(1-r)q_i(t)+\frac{e\times q_i(t)}{N-1}]\Big)$$

This inequality reduces to

$$\Big((1-r)q_0(t)+[1-e]\pi_0\Big)\times\sum_{i=1}^{N-1}[q_i(t)]\geq q_0(t)\frac{[(N-1)(1-r)]+e)}{N-1}\times\sum_{i=1}^{N-1}[q_i(t)]$$

Since the sum of the propensities for the actions 1 to $N-1$ cannot be zero, this inequality further reduces to

$$(1-r)q_0(t)+[1-e]\pi_0\geq q_0(t)\frac{[(N-1)(1-r)]+e}{N-1}$$

$\therefore$

$$(1-r)q_0(t)+[1-e]\pi_0\geq(1-r)q_0(t)+q_0(t)\frac{e}{N-1}$$

$\therefore$

$$(1-e)\times\pi_0\geq q_0(t)\frac{e}{N-1}\tag{5.6}$$

$\therefore$

$$\frac{[1-e]\pi_0[N-1]}{e}\geq q_0(t)\tag{5.7}$$

These derivations imply that the choice propensity for the best action has to be less than or equal to some constant dependent on $\{e,N,\pi_0\}$ in order for the choice probability for the best action 0 to increase from time t to $(t+1)$.

5.2.3 Propensity limits

The results of the computational experiments conducted with the SimpleModel-I test case suggest that the choice propensity for the best action converges over time. Here we analyze the MRE RL algorithm to explore whether this computational finding can be given any mathematical support.

For the best action 0, the updated choice propensity following an action choice at time $t = 1$ can take one of two forms depending on whether action 0 was selected or not:

$$\begin{aligned} q_0(1) &= (1-r)q_0(0) + \pi_0(0)(1-e) \text{ or} \\ q_0(1) &= (1-r)q_0(0) + q_0(0)\frac{e}{N-1} \end{aligned}$$

The updated choice propensity for the best action 0 following an action choice at time $t = 2$ can therefore take one of four forms depending on what actions were chosen at times $t = 1$ and $t = 2$:

$$\begin{aligned} q_0(2) &= (1-r)\Big((1-r)q_0(0) + \pi_0(0)(1-e)\Big) + \pi_0(0)(1-e) \text{ or} \\ q_0(2) &= (1-r)\Big((1-r)q_0(0) + \pi_0(0)(1-e)\Big) + \Big((1-r)q_0(0) + \pi_0(0)(1-e)\Big)\frac{e}{N-1} \text{ or} \\ q_0(2) &= (1-r)\Big((1-r)q_0(0) + q_0(0)\frac{e}{N-1}\Big) + \pi_0(0)(1-e) \text{ or} \\ q_0(2) &= (1-r)\Big((1-r)q_0(0) + q_0(0)\frac{e}{N-1}\Big) + \Big((1-r)q_0(0) + \pi_0(0)(1-e)\Big)\frac{e}{N-1} \end{aligned}$$

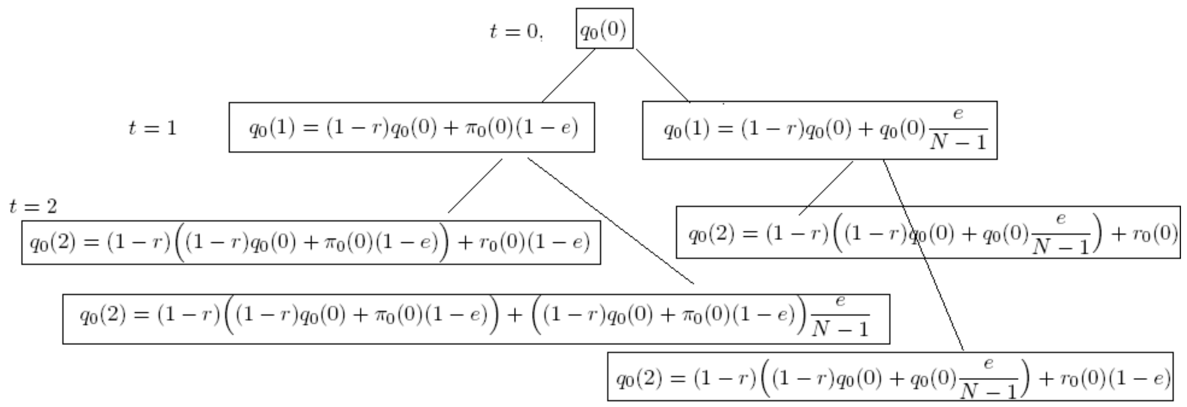

$t = 0,$
$q_0(0)$
$t = 1$
$q_0(1) = (1-r)q_0(0) + \pi_0(0)(1-e)$
$q_0(1) = (1-r)q_0(0) + q_0(0)\frac{e}{N-1}$
$t = 2$
$q_0(2) = (1-r)\Big((1-r)q_0(0) + \pi_0(0)(1-e)\Big) + r_0(0)(1-e)$
$q_0(2) = (1-r)\Big((1-r)q_0(0) + q_0(0)\frac{e}{N-1}\Big) + r_0(0)$
$q_0(2) = (1-r)\Big((1-r)q_0(0) + \pi_0(0)(1-e)\Big) + \Big((1-r)q_0(0) + \pi_0(0)(1-e)\Big)\frac{e}{N-1}$
$q_0(2) = (1-r)\Big((1-r)q_0(0) + q_0(0)\frac{e}{N-1}\Big) + r_0(0)(1-e)$

Figure 5.2 Possible initial propensity values for the profitable action of the learning agent in the SimpleModel-I experiment using MRE RL algorithm

Continuing in this fashion for t periods, we observe that the choice propensity for the best action 0 will be within the limits given by

$$q_0(t) = (1-r)^t q_0(0) + \sum_{i=0}^{t-1} \pi_0(1-e) \times (1-r)^i \text{ and}$$
$$q_0(t) = (1-r)^t q_0(0) + \sum_{i=0}^{t-1} q_0(0)\frac{e}{N-1} \times (1-r)^i$$

Letting t approach ∞, these limits become

$$q_0(\infty) = \frac{\pi_0(1-e)}{r} \text{ and} \tag{5.8}$$
$$q_0(\infty) = \frac{q_j(0)(e)}{(N-1) \times r} \tag{5.9}$$

5.2.4 Recency limits for learning

The spillover function for the MRE RL algorithm has some interesting consequences. Suppose the spillover function is set such that the choice propensity for each action always increases.

If the propensities increase irrespective of the rewards and the actions chosen, this may cause the propensities to grow exponentially large. We consider if there is a relation between the parameters to prevent the choice propensities from increasing without limit.

Suppose that the choice propensities for a non-best action j satisfy

$$q_j(1) \;>\; q_j(0)$$

$\therefore$

$$(1-r)q_j(0) + q_j(0) \times \frac{e}{N-1} \;>\; q_j(0)$$

$\therefore$

$$(1-r) + \frac{e}{N-1} > 1$$

$\therefore$

$$\frac{e}{N-1} \;>\; r \tag{5.10}$$

The following graphs show the probability of the best action with the following settings:

- initial propensity values $q_j(0)$: 6000.0
- Number of actions N: 4
- experimentation e: 0.75
- recency r: 0.26

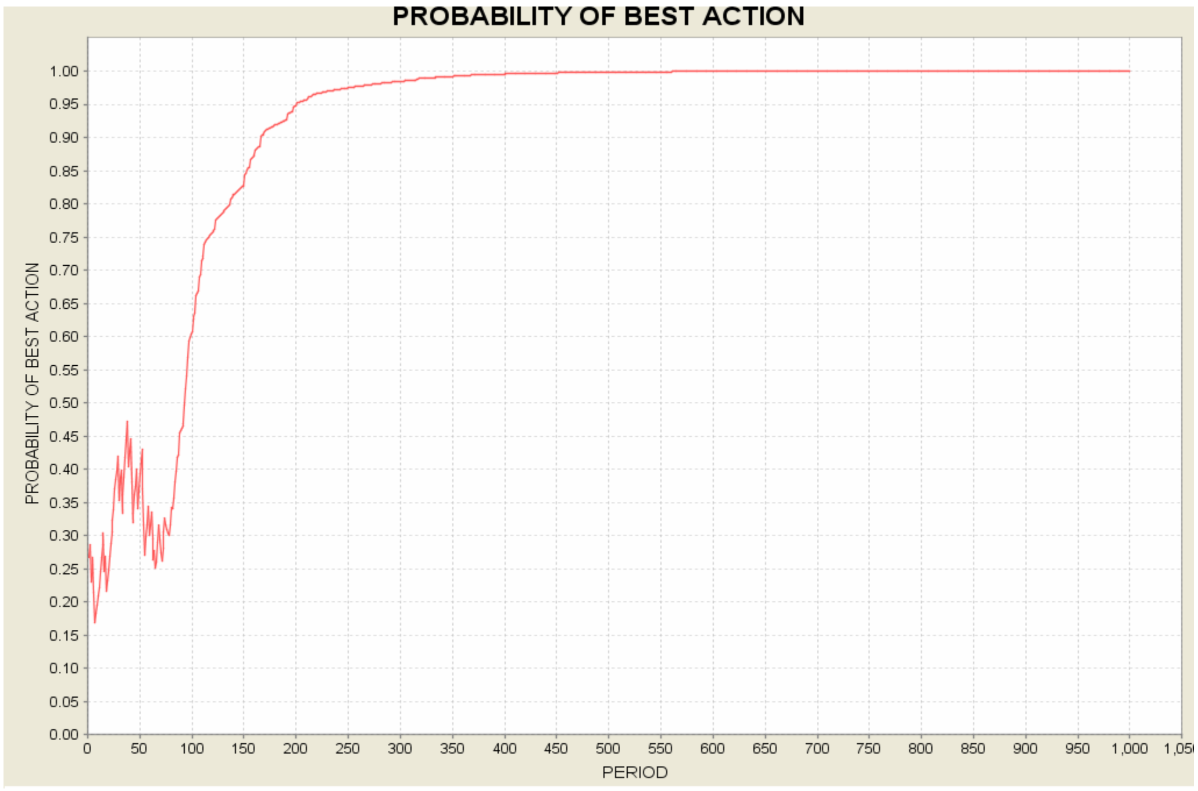


Figure 5.3 Choice Probability of the profitable action for the learning seller in the SimpleModel-I experiment using the MRE algorithm (I)

and

- initial propensity values $q_j(0)$: 6000.0
- Number of actions N: 4
- experimentation e: 0.75
- recency r: 0.24

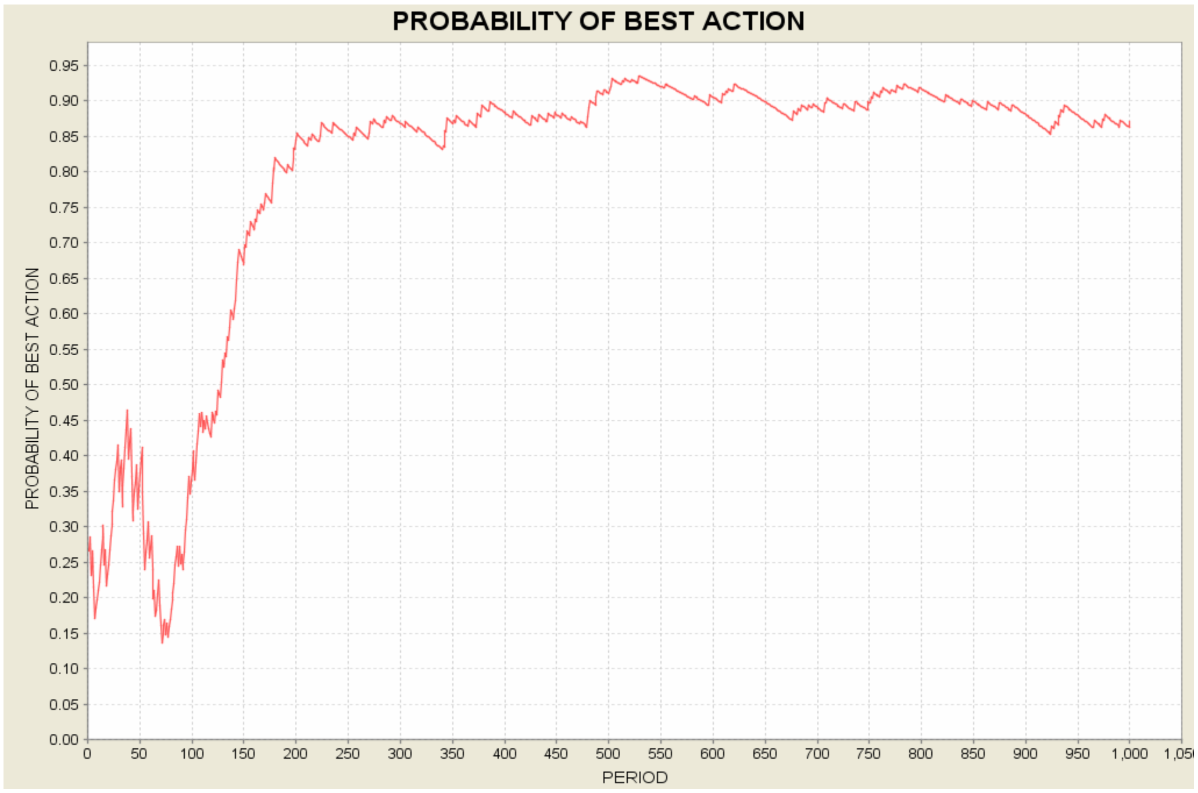


Figure 5.4 Choice Probability of the profitable action for the learning seller in the SimpleModel-I experiment using the MRE algorithm (II)

CHAPTER 6. CONCLUSION AND FUTURE WORK

We have conducted a series of experiments to study and compare the Roth-Erev reinforcement learning algorithm and its variants in double auction scenarios. In the above scenarios, the modified Roth-Erev algorithm is comparable to the original Roth-Erev algorithm.

We make a case for computational study of algorithms, especially in multi-agent contexts. For example, mathematical analysis of the algorithms may not show the the existance of a band of maximum profits that can be seen in the scaled average total profit graphs in the AMESMarket scenarios. Mathematical analysis was done only for very simple cases, and in the original Roth-Erev algorithm, this is used to show that it in certain scenarios, the algorithm will converge to a strictly dominated mixed strategy.

6.1 Cooling parameter

The Gibbs-Boltzmann distribution causes a significant impact on the learning of the agents. Since the cooling parameter is fixed, the continuous reduction in the propensities causes the difference between the probabilities to be reduced. Exploring mechanisms to update the cooling factor similar to simulated annealing should make the learning algorithm capable of learning over a wide range of parameter settings.

6.2 Evolutionary reinforcement learning

Presently work is being done in the Economics Department of Iowa State University to study Q-learning in the AMES market package [26]. An evolutionary learning online reinforcement learning algorithm implementation can be added to compare highly sophisticated learning algorithms against the existing relatively simple and myopic reinforcement learning.

6.3 Variable demand

The next release of the AMES market framework contains the implementation for the real time market for contingency power (as documented in the FERC white paper) and variable load schedules. This work can be extended to the new framework to study the impact of the environment in learning.

APPENDIX A. DEFINITIONS

Definitions

1. Double Auction: A double auction is an double-sided auction in which both buyers and sellers have to post their offers. Buyers post their bid offers, while sellers post their ask offers. Buyers and sellers are matched by constructing a supply-demand curve. Assuming a single-commodity auction, the buyer with the highest bid price is matched with the seller with the lowest ask price. If there is any quantity of goods left over for the seller after this transaction, he is matched with the buyer with the next largest bid price. If the seller is unable to fully satisfy the demand of the buyer, the seller with the next lowest ask price is matched with the buyer and so on. This process continues until the only buyers in the market are those whose ask price is too low to be matched by any seller and sellers whose bid prices are too high to be matched by any buyer. The traders who are matched in the auction are called 'infra-marginal'. Traders who are not matched in the auction are called 'extra-marginal'.

   There are two types of double auctions: Continuous and clearing house auctions. In continuous auctions, matching is done as and when bids and asks are posted, while in a clearing house auctions, matching occurs only at discrete time intervals. [13]

2. Reservation Price: The reservation price for a seller is the price below which the seller is unwilling to trade. Similarly, the reservation price for a buyer is the price above which he is not willing to trade. [13]

3. Ask price: The price quoted by the seller in an auction. [13]

4. Bid price: The price quoted by the buyer in an auction. [13]

5. Supply-demand curve: The demand curve is constructed by ordering the buyers in descending order of their bid prices. The supply curve is constructed by ordering the sellers in ascending order of their ask prices. [13]

6. Clearing price: The intersection point of the supply and demand curves is known as the clearing price. All sellers with ask prices above the clearing price do not trade. All buyers with bid prices below the clearing price do not trade. In other words, these sellers and buyers fail to be matched with any trader. The clearing price is used to compute the profits for the traders. [13]

APPENDIX B. LITERATURE REVIEW

Reinforcement Learning

In the book ‘Reinforcement Learning: An Introduction’ [7], the authors define the characteristics of a reinforcement learning algorithm to be: *policy*, *reward function*, *value function* [7, Page 7-8]. Since the RE algorithm is essentially stateless, it does not have any means of maintaining a history and defining a value function. It does however have the basic requirements of a reinforcement learning algorithm: ‘the law of effect’and ‘the power law of practice’ [1].

In [14], the authors construct a game theoretic approach to a multi-agent system and define 0-level, 1-level and 2-level learning agents. Applying the definition to the Roth-Erev family of algorithms, the algorithms are called ‘reactive’ and ‘0-level non-estimating agents’ that do not attempt to model the other agents.

Multi-agent Learning

While attempting to categorize the different aspects of multi-agent learning in [15], the authors classify the Roth-Erev RL algorithm as ‘descriptive’ because the algorithm attempts to model natural agents learning in an environment with other interacting agents.

While there is a clear difficulty in attempting to extend Q-learning to multi-agent context, there are some algorithms that attempt to incorporate the effect of other interacting agents into a learning algorithm. The minimax Q-learning [16] and the Nash-Q learning [17] are two examples of this attempt.

Roth-Erev Algorithm

In [12], the author analyzes the Roth-Erev algorithm, notes that the ‘weight of history’ makes the algorithm unable to adjust quickly in case of change in the payoff of the actions. The author also notes that setting the initial propensity greater than the payoff for all actions causes greater experimentation.

In [18], the authors define a General Reinforcement Learning algorithm by merging the Bush-Mosteller and the Roth-Erev equations into a single form. The authors give a study of the GRL algorithm in the context of two-person zero-sum games.

MRE RL Algotihm

In [19], the authors use the AMES market framework of Nicolaisen et al to measure the market efficiency obtained by evolving trading strategies using genetic programming instead of the MRE algorithm.

APPENDIX C. SOFTWARE

Software

An introduction to the software used in this work is given below.

Simple Double Auction

The Simple Double Auction software was developed as an extension to the JReLM package. It contains an implementation of a simple uniform price double auction. It simulates a single-commodity market where the sellers and buyers can choose among several actions and a clearing house auctioneer who computes the supply-demand curve , the uniform price and the set of infra-marginal sellers and buyers. The sellers and buyers can be set up to have a reinforcement learning component with multiple actions.

JReLM

The Java Reinforcement Learning Module is a software package developed by Charlie Geiseler [20]. The package implements the RE, MRE and the VRE RL algorithms within a Repast framework. It can be used as a stand-alone extension to the RepastJ package. JReLM is used as the reinforcement learning module within the AMES framework.

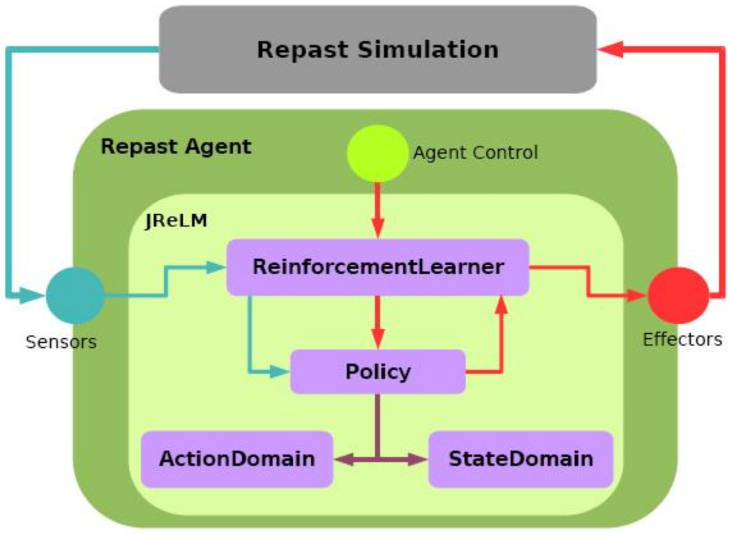


Figure C.1 JReLM interaction with Repast [20]

RepastJ

The Recursive Porous Agent Simulation Toolkit is a

> specification for agent-based modeling services or functions

[21]. It was developed primarily to model social agents. From the Repast 3 homepage [1],

> ...the toolkit gives users complete flexibility as to how they specify the properties and behaviors of agents

. This allows the modeler to use Repast in a wide variety of domains. In the AMES market package [22], the software is built as a Repast model. In this work, we use the pure Java implementation of Repast, RepastJ v3.1

AMES Market Package

The Agent-based Modeling of Electricity Systems (AMES) market package was developed as a wholesale energy market implementation based on the U.S. Federal Energy Regulatory

[1]http://repast.sourceforge.net/repast_3/index.html

Commission in an April 2003 White Paper recommendations [22]. It contains the core components required for a wholesale power market platform. In this work, we use version 1.31 of the AMES framework for our test bed.

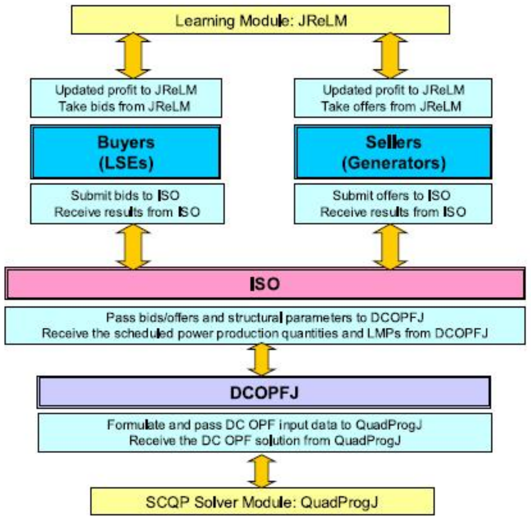

Learning Module: JReLM
Updated profit to JReLM
Take bids from JReLM
Updated profit to JReLM
Take offers from JReLM
Buyers
(LSEs)
Sellers
(Generators)
Submit bids to ISO
Receive results from ISO
Submit offers to ISO
Receive results from ISO
ISO
Pass bids/offers and structural parameters to DCOPFJ
Receive the scheduled power production quantities and LMPs from DCOPFJ
DCOPFJ
Formulate and pass DC OPF input data to QuadProgJ
Receive the DC OPF solution from QuadProgJ
SCQP Solver Module: QuadProgJ

Figure C.2 Core Modules of the AMES Market Package [22]

DCOPFJ

DC-OPFJ stands for the DC-Optimal Power Flow [23]. The software package is a Java-based OPF solver which takes in as input the electricity demand and supply at different nodes in a transmission grid and returns the locational marginal price (LMP) at each node. The solver finds the minimum LMP for the network. The DCOPFJ software is used in the AMES market package to determine the power required from each generator and therefore the profits for each generator.

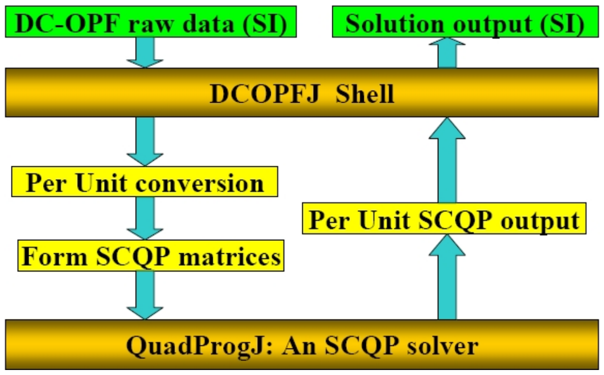


Figure C.3 DC-OPFJ Solver [23]

APPENDIX D. TEST BED IMPLEMENTATIONS FOR THE FIVE TEST CASES

D.1 Simple Double Auction Testbed Implementation for SimpleModel-I and II

D.1.1 Introduction

The software simulates a simple repeated clearing house double auction with uniform pricing. At the start of the simulation, the following parameters are read from a user-specified file.

- Number of sellers
- Number of buyers
- Reservation values of each seller and buyer
- Number of actions with the corresponding bid and ask price/ quantities for each seller and buyer
- Type of reinforcement learning component for each seller and buyer.
- Learning parameters for each seller and buyer.
- Number of time periods for which the simulation is to be run.

If no actions are specified for a seller or buyer, the seller or buyer always bids using their true reservation values as the bid or ask prices. It is not possible to modify the market parameters after initialization, so there is no way to add or remove agents or their actions in the middle of the simulation. Also, no buyer can switch to be a seller, or vice versa.

The simulation can be run in two ways.

- Single Run: In this type of run, all parameters are fixed in the parameter file and the simulation runs for the specified time period. The XML file generated contains the results of the run.

- Multi-run: A multi-run is a feature of RepastJ. In this type of run, not all parameters are required to be set in the parameter file. Instead a file name can be provided that specifies the range in which the parameters can be modified. The file has to be specified in the Repast multi-run parameter format [24].

During a single period of the double auction, based on the bid and ask prices, the market operator constructs the supply-demand curve and determines the clearing price. The market operator then reports the clearing price to all agents. The market operators also reports to each agent if it was infra-marginal or extra marginal in that time period. If the agent is infra-marginal, then the profit for the agent in that period is the difference of the uniform clearing price and the agent's reservation value. If the agent is extra marginal, the profit for that agent in that period is zero.

Using this testbed, it is possible to study the interrelatedness of the experimentation, recency and initial propensity and action domain size.

D.1.2 Architecture

The RE, MRE and VRE RL algorithms are already implemented in the JReLM package. An additional package edu.iasate.jrelm.demo.doubleauction was created on the lines of the existing edu.iastate.jrelm.demo.bandit package. Also, a few classes were developed in the edu.iastate.jrelm.demo package to duplicate the functionality of the existing classes to support the usage of the MRE RL algorithm.

The classes developed are:

- DoubleAuctionModel: This class contains the *main()* method to invoke the simulation. This class extends the SimpleModel from the RepastJ framework. The *main()* function loads the user-specified parameter file. The class also contains functions to log the input parameters and intermediate results in an XML file.

- DASellerAgent: This class extends the RothErevAgent<DAAgentAction> class present in JReLM. This class has a reinforcement learning component. A seller agent contains an object of type DAAgentActionDomain.

- DABuyerAgent: This class extends the RothErevAgent<DAAgentAction> class present in JReLM. This class has a reinforcement learning component. A buyer agent contains an object of type DAAgentActionDomain.

- DAAgentActionDomain: The class is a container of objects of type DAAgentAction.

- DAAgentAction: An object of this class defines an action that can be taken by the DASellerAgent or DABuyerAgent. Each DAAgentAction contains the quantity and price associated with the action.

- FixedSeller: This class has no learning component and always sells at a fixed price as determined by its reservation value.

- Seller: This interface contains the functions common to both the fixed seller and the DASeller.

- Buyer: This interface contains the functions common to both the fixed buyer and DABuyer.

- FixedBuyer: This class has no learning component and always buys at a fixed price as determined by its reservation value.

- Market: This class contains the functions required to compute the demand supply curve and notifies the agents that are infra-marginal in the demand-supply curve.

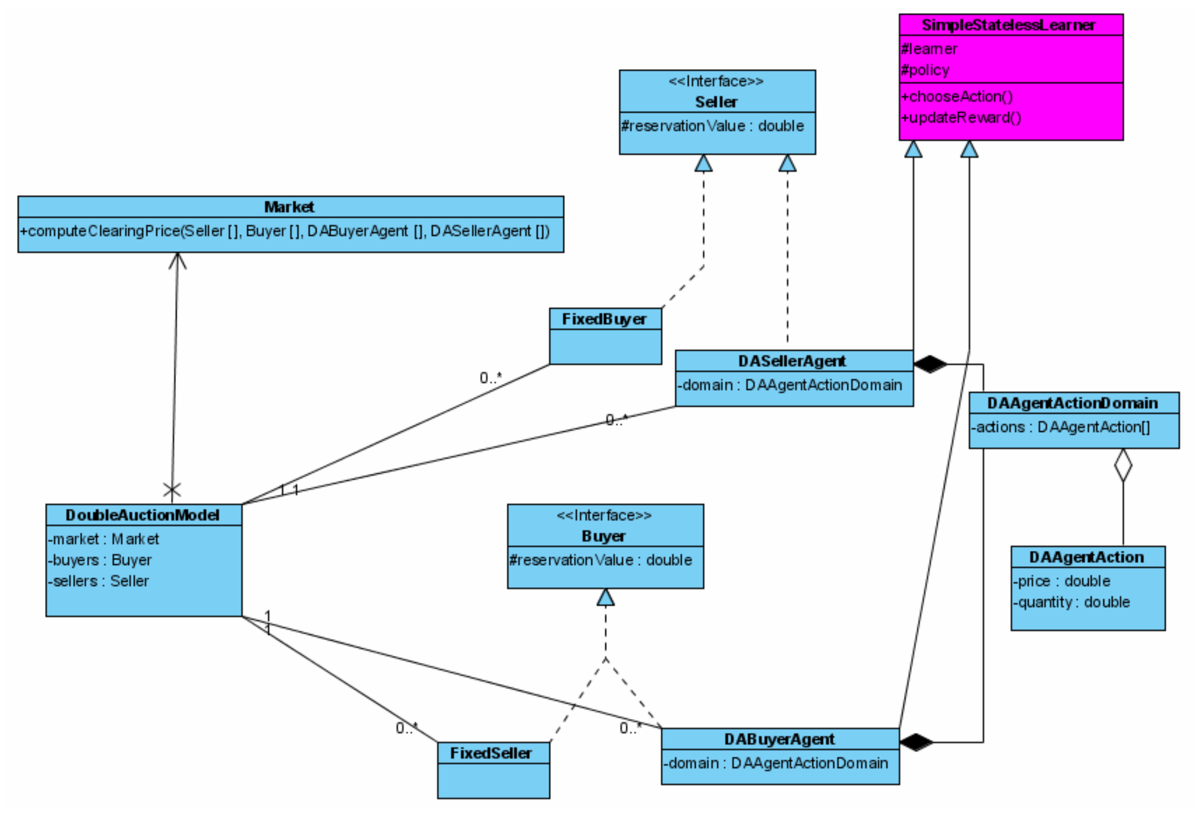


Figure D.1 Class Diagram of the Simple Double Auction Demo package

D.1.3 Running the model

The DoubleAuctionModel has a *setup()* function in which based on the user-specified number and type of agents, the corresponding DASellerAgent, DABuyerAgent, FixedSeller and FixedBuyer agents are initialized. For each learning agent, a number of DAAgentActions is created based on the user-specified input parameters. The XML log file is created in this function.

The *buildSchedule()* function specifies that the *step()* function is to be invoked at each tick and that the *stop()* function is to be invoked at the user specified time period. The *stop()* function stops the simulation.

In the *step()* function, first each agent specifies an action which has a price and quantity offer. For a FixedBuyer or FixedSeller agent, the price and quantity corresponds to its reservation value and fixed quantity. For the learning agents, the actions are specified by their policy. All the price/quantities specified by the seller and buyer agents are provided to the Market object to compute the demand-supply graph and determine the uniform price. Also, the market object notifies the agents that are infra-marginal. If the agent is infra-marginal, the profit is computed as the difference between the market clearing price and the agent's reservation value. Each learning agent updates its internal propensity and probability values based on the clearing price declared by the Market object.

The *stop()* function closes the XML log file. The log file contains all the parameters used to set up the simulation as well as the propensity and probabilities of the various actions of each agent, the action selected at each tick and the reward obtained at each tick.

D.2 AMES Testbed Implementation for AMESModel-I through III

D.2.1 Introduction

The testbed is based on the AMES market package developed by Sun and Tesfatsion [3]. Version 1.31 of the software is used for this work [1] [22].

The package is an implementation of a uniform price double auction market in the deregulated electricity markets environment. It is based on the FERC White paper in which the specifications are collected as the Wholesale Power Market Platform (WPMP).

At the heart of the implementation is a day-ahead market, wherein all sellers (generators) and buyers (load serving entities (LSEs)) specify their supply and demand commitments for the next day. This supply and demand schedule is provided indicating the hourly demand and supply required for the next day. The market operator known as the Independent System Operator (ISO), or Regional Transmission Organizations (RTOs) are responsible for the security and maintenance of the transmission grid. The market operator collects the supply and demand offers from all agents in the market. From the given supply and demand offers, the market operator computes the minimum cost of supplying power using a sophisticated non-linear optimization problem solver. This minimum cost of power is known as the locational

[1] The software can be downloaded from *www.econ.iastate.edu/tesfatsi/AMESMarketHome.htm*

marginal pricing (LMP) since it depends not only on the supply and demand offers, but also the location of the generators and LSEs on the transmission grid and the network capacity constraints.

Based on the minimum cost of power, the market operator computes the hourly commitments from each generator and therefore, the profits accrued by each generator.

The ISO calculates the profits for each generator for the day and pays the money to the generator. The generator updates its revenues with the profits. There is no penalty for the revenues to be negative. It is assumed that the generator is still able to function with no impact. The system is assumed to have no shocks, so that there are no outages and no need for a real-time trading to mitigate lost generators.

Typically, the generators have a reinforcement learning component and they modify their behavior based on the profits generated by the actions.

A modification in the existing software is done to allow multiple-runs using properties files as specified in the RepastJ manual.

Using the parameters M1, M2 and M3, it is possible to adjust the number of actions for a given generator. We can restrict the action domain of the generator to contain a single action. The single action of the generator corresponds to its true supply offer of the generator. Since there are no options for the agent to learn, the generator always selects the single action and hence is a non-learning agent.

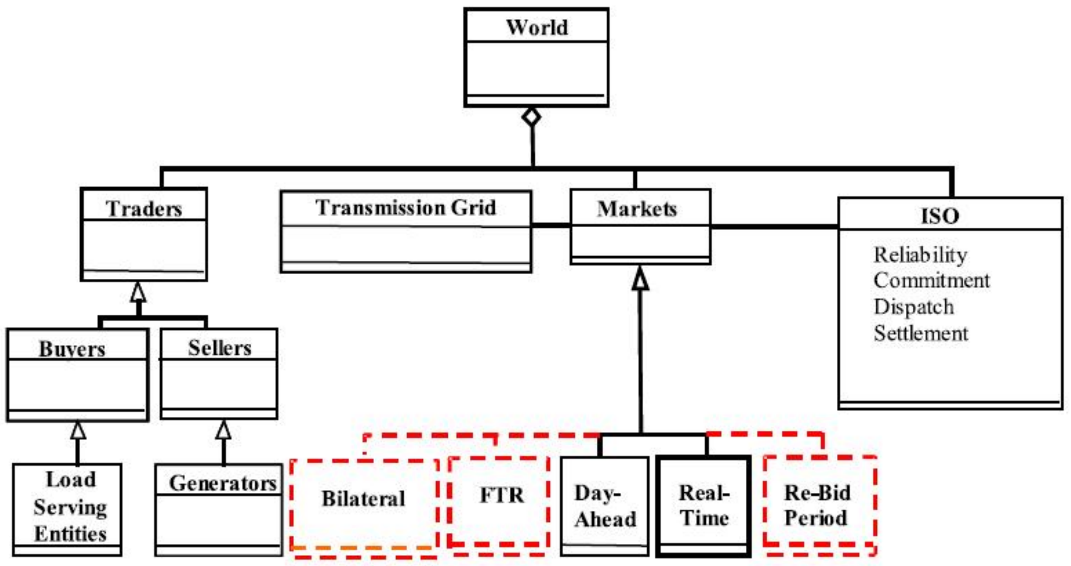


Figure D.2 AMES Architecture [3]

D.2.2 Architecture

The architecture of the AMES framework is built as a RepastJ model.

The main class of the AMES framework is the AMESMarket class which extends the SimpleModel class from RepastJ. This class has *step()* function which runs the actions to be performed at each time period and the *stop()* function, which stops the simulation and is used for any clean-up activities.

The AMESFrame class is a GUI front end to the AMESMarket class and allows the user to create a test case, or load an existing test case. On reading the test case parameters, it constructs an AMESMarket object and initializes it with the test case parameters. The test case consists of the number of generators, the LSEs and the parameters associated with the generators and the LSEs. It also contains the network layout of the transmission grid, including the locations of the generators and LSEs and the transmission line characteristics (reactances and line capacities).

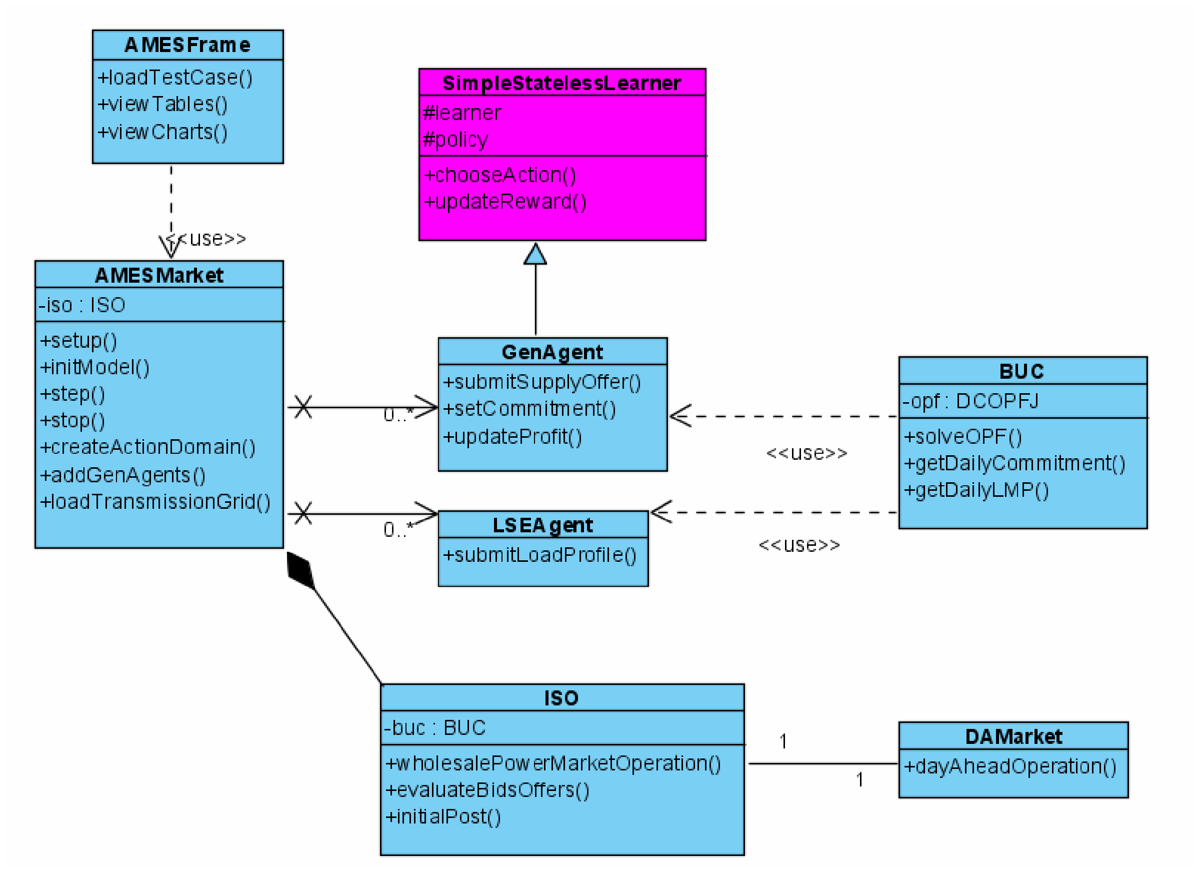


Figure D.3 Class Diagram for the AMES market package

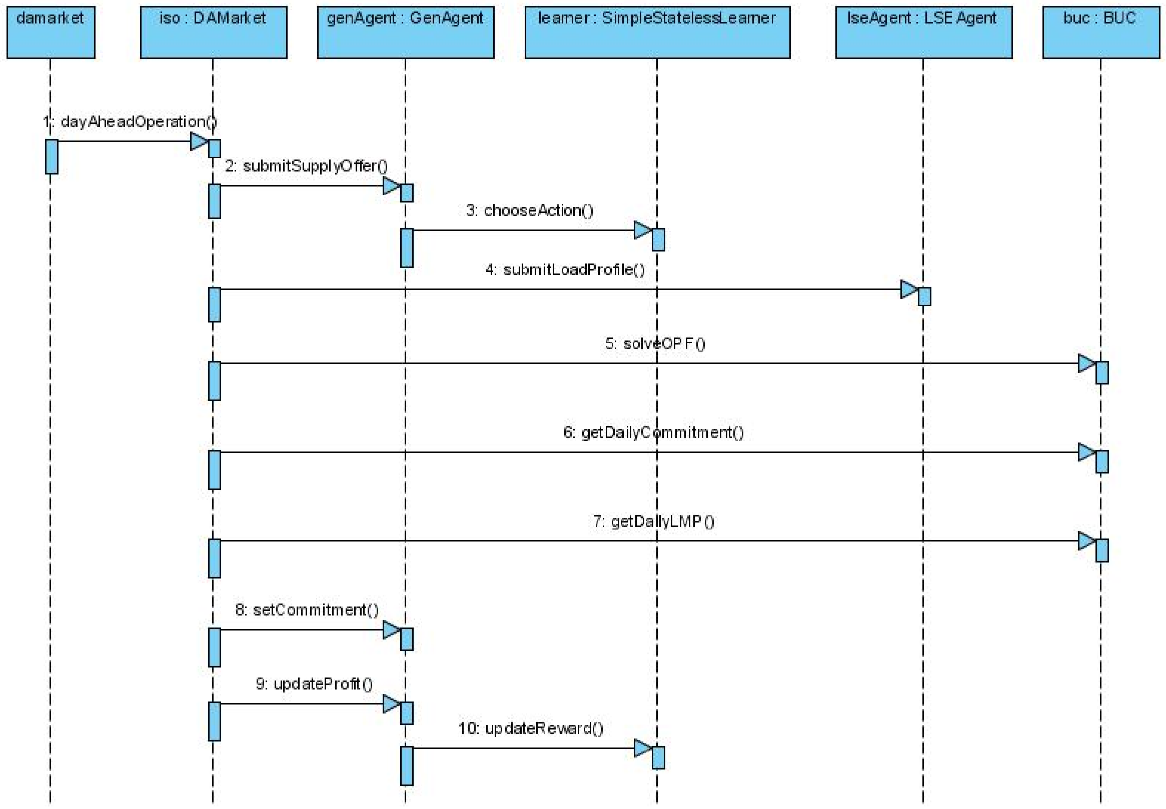


Figure D.4 Sequence Diagram for the *step()* function in the AMES market package

D.2.3 Modifications

Since the main class of the AMES framework is AMESGUIFrame.AMESFrame, it is not possible to run the application as a RepastJ multi-run based on parameter files. Hence, a few modifications were made to the classes to support this.

- AMESMarket.AMESMarket.main function: A new main function was added to the class to provide an entry point. Also, the main function can take in parameter file as an argument which allows it to run the model as a multi-run simulation.
- AMESMarket.AMESMarket.log functions: An XML file is generated for each simulation

- AMESMarket.AMESMarket.addGenAgents: This function was modified to allow the user to specify the number of learning agents allowed in the simulation. If the number of non-learning agents is specified to be greater than 0, then the action domain for the non-learning agents GenAgent has only one action. The regular non-learning agents can have their action domain size specified by the M1, M2 and M3 parameters. The last agent in the list is always the learning agent which we are interested in. We can specify the reinforcement learning algorithm used by this agent.

- AMESMarket.GenAgent.GenAgent constructor: A new constructor is provided to allow it to accept an object of type REParameters. This is to allow the GenAgent to have a Roth-Erev reinforcement learning component. The implementation to allow the GenAgent to have a Modified Roth-Erev learning component is already present in the AMES framework.

D.2.4 Running the model

The model can be run in two ways:

- Single run: The single run is initialized by running the AMESFrame main class. This loads the GUI for the AMES market framework. On loading a test case, the simulation can be started by clicking on the 'Start' button. This loads the test case as a transmission grid with the grid information, the location of the generators and LSEs and the generator parameters and the LSE daily load schedule. In v1.31 of the AMES market, the LSEs are assume to have a constant load for every hour of every day. Once the stopping rule has been reached, the GUI can be used to view the information of the run in a graphical or tabular format.

- Multi-run: In a multi-run, the AMESMarket class is invoked using a RepastJ parameter file. The parameter file contains the interval to be used for the learning parameters and the random seed list used to invoke the run. For each run of the simulation, an XML file is generated with the parameters used to set up the simulation as well as the intermediate learning parameters.

REFERENCES

[1] Erev, I. and Roth, A. E. (1988). Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria. *The American Economic Review*, *88*(4), 848–881.

[2] Nicolaisen, J. and Petrov, V. and Tesfatsion, L. (2001). Market power and efficiency in a computational electricity market with discriminatory double-auction pricing. *Evolutionary Computation, IEEE Transactions on*, *5*(5), 504–523.

[3] Sun, J. and Tesfatsion, L. (2007). Dynamic Testing of Wholesale Power Market Designs: An Open-Source Agent-Based Framework *Computational Economics*, *30*(3), 291–327.

[4] Gode, D. K. and Sunder, S. (1993). Allocative Efficiency of Markets with Zero-Intelligence Traders: Market as a Partial Substitute for Individual Rationality. *The Journal of Political Economy*, *101*(1), 119–137.

[5] Whiteson, S. and Stone, P. (2006). Evolutionary Function Approximation for Reinforcement Learning. *The Journal of Machine Learning Research 7*, 877–917.

[6] Stanley, K. O. and Miikkulainen, R. (2002). Evolving Neural Network through Augmenting Topologies. *Evolutionary Computation*, *10*(2), 99–127.

[7] Sutton, R. S. and Barto, A. G. (1998). Reinforcement Learning: An Introduction. *MIT Press*

[8] Li, H., Sun, J. and Tesfatsion, L. (2008). Dynamic LMP Response Under Alternative Price-Cap and Price-Sensitive Demand Scenarios. *Proceedings, IEEE Power Engineering Society General Meetings, Carnegie-Mellon University, Pittsburgh, July 20-24, 2008*, to appear.

[9] Sun, J. and Tesfatsion, L. (2007). An Agent-Based Computational Laboratory for Wholesale Power Market Design. *Proceedings, IEEE Power Engineering Society General Meeting, Tampa, Florida, June 2007*

[10] Rejeb L., Guessoum Z. and M'Hallah R. (2005). An adaptive approach for the exploration-exploitation dilemma and its application to economic systems. *in proa. AAMAS 05, Workshop 20: Learning and Adaptation in MAS(LAMAS)*, 97–106.

[11] Koesrindartoto, Deddy (2002). Discrete Double Auctions with Artificial Adaptive Agents: A Case Study of an Electricity Market Using a Double Auction Simulator. *Working papers series (Iowa State University. Dept. of Economics) no. 02005.*

[12] Shor, M. (2004). Learning to Respond: The Use of Heuristics in Dynamic Games. *Mainted online at* http://econweb.tamu.edu/workshops/Theory and Experimental Economics/Mikhael Shor.pdf*. Current as of January 2008*

[13] Parsons, S. and Marcinkiewicz, M. and Niu, J. (2005). Everything you wanted to know about double auctions but were afraid to (bid or) ask. *Technical report, Department of Computer & Information Science, Brooklyn College, City University of New York, 2005.*

[14] Hu, J. and Wellman, M. P. (1998). Online learning about other agents in a dynamic multiagent system. *Proceedings of the second international conference on Autonomous agents*, 239-246.

[15] Shoham, Y. , Powers, R. and Grenager, T. (2002). If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, *171*(7) 365–377.

[16] Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. *Proceedings of the Eleventh International Conference on Machine Learning, 157163*

[17] Hu, J. (2004). Nash Q-Learning for General-Sum Stochastic Games. *Journal of Machine Learning Research, 4*(6), 1039–1069.

[18] Flache, A. and Macy, M. W. (2002). A more general model of cooperation based on reinforcement learning: Alignment and Integration of the Bush-Mosteller and the Roth-Erev model. *Maintained online at* http://cfpm.org/m2m/papers/Flache_M2M.pdf*. Current as of January 2008.*

[19] Phelps, S.,Parsons, S., McBurney, P. and Sklar, E. Co-evolution of auction mechanisms and trading strategies: Towards a novel approach to microeconomic design. *Proc. GECCO-02 Workshop Evolutionary Computation in Multi-Agent Systems* 65 –72.

[20] Gieseler, C. J. (2005). A Java reinforcement learning module for the Repast toolkit: Facilitating study and experimentation with reinforcement learning in social science multi-agent simulations. *Department of Computer Science, Iowa State University, MS Thesis.*

[21] Repast. Recursive Porus Agent Simulation Toolkit. *Maintained online at* http://repast.sourceforge.net/repast_3/index.html*. Current as of January 2008*

[22] Tesfatsion, L. (2007). The AMES Market Package (Java): A Free Open-Source Test Bed for the Agent-Based Modeling of Electricity Systems. *Maintained online at* http://www.econ.iastate.edu/tesfatsi/AMESMarketHome.htm*. Current as of January 2008*

[23] Tesfatsion, L. (2007). DCOPFJ (Java): A Free Open-Source Solver for Bid-Based DC Optimal Power Flow Problems. *Maintained online at* http://www.econ.iastate.edu/tesfatsi/DCOPFJHome.htm*. Current as of January 2008*

[24] Repast HowTo Documentation and Third Party Tutorials. How to use Parameters and Parameter Files. *Maintained online at* http://repast.sourceforge.net/repast_3/tutorials.html*. Current as of January 2008*

[25] Tesfatsion, L. (2007). Repast: A Software Toolkit for Agent-Based Social Science Modeling. Self-Study Guide for Java-Based Repast (RepastJ). *Maintained online at* http://www.econ.iastate.edu/tesfatsi/repastsg.htm*. Current as of January 2008*

[26] Yu, N. and Liu, C. C. and Tesfatsion, L. (2007). Modeling of Suppliers' Learning Behaviors in an Electricity Market Environment. *Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on*, 1–6.