Master thesis

Analysing equilibria on simple electricity markets with agent-based Q-learning

In partial fulfilment of the requirements for the degree of

"Diplom-Ingenieur"

Within the Master's Programme

"Stoffliche und energetische Nutzung nachwachsender Rohstoffe (NAWARO)"

Submitted by

Thomas Haydn, BSc

01340270

University of Natural Resources and Life Sciences, Vienna

Department of Economics and Social Sciences

Institute for Sustainable Economic Development

Supervised by: Ass.Prof. Dipl.-Ing. Dr. Johannes Schmidt

Co-Advisors: Dr. Claude Klöckl, Mag. Sebastian Wehrle

Vienna, February 2020

Statutory declaration

I declare that I have developed and written the enclosed Master Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Master Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

Place, Date Signature, Thomas Haydn

Acknowledgement

I would like to thank my family, especially my parents, for supporting me not only during my thesis, but for the whole time of my studies and education. Thank you not only for your financial but also and even more for your moral support.

Furthermore, I am very grateful for the constant and patient work of my supervisor Ass.Prof. Dipl.-Ing. Dr. Johannes Schmidt and my co-advisors Dr. Claude Klöckl and Mag. Sebastian Wehrle. I highly benefited from their motivation and their helpful inputs.

Abstract:

Q-learning has become one of the most popular reinforcement learning methods for simulating market participants. Traditional optimization methods are not effective in all instances since some of the underlying optimization problems are non-convex and therefore not well solvable. To address this challenge, Q-learning continuously improves player actions, thus converges towards the optimal solution. In this thesis we implement a Q-learning algorithm and use it on a simplified model of a hypothetical electricity market with uniform pricing. We assess if strategy combinations, derived from Q-learning, are pure Nash equilibria. Further, we want to determine limits of the method. We show that Q-learning is indeed capable of finding pure Nash equilibrium states for simple electricity market situation. Moreover, we learn that the model converges to certain pure Nash equilibria more frequently. Finally, we analyze and explain the distribution of results.

Zusammenfassung:

Q-learning ist eine der beliebtesten Methoden des verstärkenden Lernens um Marktteilnehmer zu simulieren. Herkömmliche Optimierungsmethoden sind nicht in allen Fällen dafür geeignet da die zugrundeliegenden Aufgabenstellungen nicht immer konvex sind und daher nur schwer gelöst werden können. Q-learning ist in der Lage diese Probleme zu umgehen indem nicht die optimale Lösung gesucht wird, sondern das Spielerverhalten kontinuierlich verbessert wird. In dieser Arbeit implementieren wir einen Q-learning Algorithmus und wenden diesen am Beispiel eines einfachen Elektrizitätsmarktes an. Wir analysieren in wie weit simulierte Strategiekombinationen reinen Nash-Gleichgewichten entsprechen. Weiters untersuchen wir, wo die Grenzen der Methode liegen. Wir zeigen, dass Q-learning tatsächlich in der Lage ist reine Nash-Gleichgewichte für einfache Marktsituationen zu finden. Außerdem stellen wir fest, dass das Model zu gewissen Gleichgewichten öfters konvergiert als zu anderen. Diese Ergebnisverteilung wird analysiert und erklärt.

Table of contents

The implemented Q-learning algorithm can be found here:

https://github.com/Thomas-Haydn/Q-learning-electricity-market

1. Introduction

1.1 Aim of the thesis

Understanding the dynamics of complex markets is fundamental for regulators to design market rules which prevent market participants from gaining excessive rents. Thus, different methodologies for analysing markets, like optimization and game theoretical concepts, have been developed. However, markets become more and more complex and therefore very difficult to analyse. Traditional methods like optimization algorithms are pushed to their limits due to the non-convex character of some underlying problems (Papageorgiou et al., 2015). Finding the optimal solution for complex market models with optimization can take an infeasibly large amount of computational time (Tesfatsion, 2006). In order to deal with this issue heuristics are needed.

As such, reinforcement learning deals with the idea of agents solving problems by trial and error (Kaelbling et al., 1996). On the one hand reinforcement learning can be used to represent realistic real-life individuals with the aim to analyse their behaviour. On the other hand, reinforcement learning can be utilized in a more technical sense in order to find the most effective actions. In this thesis we will mainly focus on the technical aspect of optimizing the selection of actions.

According to Osborne and Rubinstein (1994) a player is classified as rational, if she satisfies four constraints. These are: A clear preference, awareness of all alternatives and constraints, optimization of behaviour and expectations about unknowns. For reinforcement learning an agent's preference is defined by a distinct reward function. Further, the agent has all information about her possible actions. Thus, the first two properties of rationality are given for reinforcement learning. However, optimization of actions and expectations of the unknown are only satisfied to a certain extent. An agent is not motivated to find the optimal available action in the first place, but improves her decision making slowly over time by trial and error. Furthermore, the only considered information about possible future events is based on the rewards of

the past. Agents do not know how to deal with risk or even uncertainty and do not use economic expectation values.

This restricted form of rationality fits well to simulate players on a complex market, since real-life individuals often have clear preferences and know their possibilities but are limited in predicting future rewards and finding optimal actions. Moreover, the assumption of completely rational market participants is not realistic, like behavioural economics show (for instance: Wilkinson, 2012).

The need for suitable market analysing methodologies and the appealing properties of reinforcement learning strategies are the fundamental motivation for writing this thesis. We implement a simple Q-learning algorithm suggested by Watkins (1989) and apply it to a basic model of the electricity market. This reinforcement learning method allows modelled players to evaluate chosen actions and to pick future actions according to past evaluations. In specific, the following questions are addressed: Is it possible to determine pure Nash equilibria with a basic Q-learning algorithm using the example of simple electricity market situations? Where are the computational limits of determining Nash equilibria with a Q-learning algorithm in terms of computational time and size?

In order to answer these research questions we simulate different possible market situations using a Q-learning algorithm, implemented with the programming language Python. Each parameter setting is run multiple times. Then we compare the results with pure Nash equilibria and analyse the computational time needed to find those. We expect to find Nash equilibria for simple set-ups like Krause et al (2004) and others have found for similar approaches. Limits of the method are analysed in order to improve future work on this topic.

1.2 Thesis structure

Following the introduction, we give a brief summary of relevant related topics including game theoretical equilibria, reinforcement learning and electricity markets. Afterwards we present the implemented Q-learning algorithm with all specifications and parameters. Then the results of three different experimental set-ups are shown

and analysed. Finally, we compare our simulation with similar approaches in literature, identify weaknesses and limits and give a short outlook on possible future work.

2. Theoretical foundations

This chapter will illustrate the theoretical foundations of game theory and Q-learning. At the end of the chapter, we will also describe the Austrian electricity market, since a very simplified version of it is used as example application throughout the thesis.

2.1 Equilibria in game theory

Non-cooperative game theory is used to describe and understand situations in which decision-makers interact in a way that one player's pay-off depends on the actions of other players. It is the formalized study of the strategic choice of actions. Decision-makers are called players and try to maximize their pay-offs by choosing actions from an available action-set. This can either happen simultaneously for static games or sequentially for dynamic games (Gibbons, 1992). The algorithm implemented in this thesis reflects a static game. As an example, a player may be a company, deciding on a price for its product (action) in order to maximize its profit (pay-off). The outcome of the game is determined by the interaction of all actions chosen by all players. Therefore, a player's pay-off is influenced by all players' decisions and not only by her own. All chosen actions for one iteration of a game are summarized in the action profile.

One main aim of game theory is to study which actions are optimal for decision-makers to choose in a strategic game. We assume that players want to pick an action which maximizes their pay-offs. Moreover, all players are assumed to behave rationally, i.e. they pursue a policy of pay-off maximization. Such behaviour can lead to one or more equilibrium states. These are referred to as Nash equilibria. By definitions a Nash equilibrium is an action profile a^* with the property that no player i can do better by choosing an action different from a^*_i, given that every other player j adheres to a^*_j (Osborne, 2003). Thus, each player's action is a best response to each other player's. A Nash equilibrium represents a state in which no player has an incentive to deviate from her chosen strategy. By this definition, Nash equilibria are not necessarily unique (Gibbons, 1992).

We can distinguish between pure and mixed strategy Nash equilibria. Pure strategy Nash equilibria describe certain combinations of single best response actions. However, this concept does not allow players to vary actions. More general concepts of equilibria allow a player's choice of actions to vary as long as the pattern of choices remains the same. That means actions are chosen probabilistically according to an unchanging distribution. We can find Nash equilibrium states with this approach as well. These are called mixed strategy Nash equilibria (Osborn, 2003). Players in the implemented Q-learning algorithm do not randomize over strategies. Therefore, the algorithm can find pure Nash equilibria, but cannot find mixed Nash equilibria.

If a game theoretical model gives a prediction about the actions each player will choose for a certain game, this combination of strategies has to be a Nash equilibrium. For any strategy profile, which is not a Nash equilibrium, at least one player has an incentive to improve her position by switching strategy. A pay-off maximizing player will always choose the best response action. This property makes the concept of Nash equilibria very useful (Gibbons, 1992).

Even before the concept of Nash equilibria were first suggested and proven by John Nash in 1950, other economists anticipated his concept. Augustin Cournot did so in his famous model of oligopoly in "Researches into the mathematical principles of the theory of wealth" (1838). The model considers n players who sell identical products. Every unit produced is sold at the same price, determined by the inverse demand function and the players' total output (i.e. total supply). Each player can choose which quantity to offer in order to maximize his profit. For sufficiently simple (inverse) demand functions, Cournot's oligopoly games can be solved algebraically for the ideal quantity produced by each player. Indeed these solutions are Nash equilibria (Osborne, 2003).

In 1883 Joseph Bertrand suggested that players rather choose prices and not quantities. From the basis of this assumption he developed his own model for oligopoly games. Bertrand's model is similar to Cournot's, but differs in strategy spaces and pay off functions. Here, the demand is a function of the current price (Gibbons, 1992). The solution of Bertrand's model is a Nash equilibrium as well. The calculated equilibrium price equals the marginal costs of each player. Therefore,

players cannot generate economic profit in Bertrand's model, whereas in Cournot's model players are indeed able to generate positive profits (Osborne, 2003).
Both presented oligopoly models act on the assumption that demand and price are influencing each other. Cournot assumed that the price is dependent on the current quantity of demand and total output, whereas Bertrand assumed that demand depends on the price. However, the algorithm implemented in this thesis considers an inelastic, constant demand. Therefore, it deals with a simplified special case.

2.2 Q-learning

In this thesis we will take a closer look at the Q-learning algorithm applied on an agent-based system as a heuristic to find pure Nash-equilibria on electricity markets. Q-learning was first suggested by Watkins (1989). It was initially designed for learning through interaction with a Markov Decision Process. Q-learning is counted among the reinforcement learning methods, which deal with the problem of agents learning from experience (Krause, 2004). Through reinforcement learning agents learn about their dynamic environment by trial and error. In each iteration the agent receives an input, which is an indication of the current state. The agent then chooses an action. This action changes the current state and this state transition is communicated to the agent through a scalar reinforcement signal. In the long-run the agent tries to maximize the total value of reinforcement signals (Kaelbling et al., 1996). Sutton and Barto (2017) compare reinforcement learning to an infant who does not have an explicit teacher but a connection to its environment. This connection produces plenty of information about cause and effect, about consequences and about what to do in order to achieve goals.
In Q-learning, agents evaluate actions not only depending on their immediate consequences, but also on the new situation they lead to. Reinforcement signals are used to evaluate Q-values of state-action pairs (Watkins, 1989). The evaluation value $Q(s, a)$ is defined as the discounted sum of future rewards obtained by choosing an action a, in situation s (Weidlich, 2008).

Q-values are updated after every played iteration based on the received reward, according to the Bellman equation:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \cdot \left[r_t + \gamma \max_{a_t, t} Q_t(s_{t+1}, a_t) - Q_t(s_t, a_t) \right] \quad (1)$$

The learning rate $0 < \alpha < 1$ regulates how strong new rewards influence the according Q-value. The discount factor $0 < \gamma < 1$ describes how strong expected future rewards are represented in the according Q-value. Agents assume that state-action pairs with high Q-values lead to high rewards. Therefore, they are more likely to be picked in the future. Watkins and Dayan (1992) were able to prove that Equation 1 converges to an optimal solution $Q^*(\mathrm{s}, \mathrm{a})$ under the condition that all actions and states were visited infinite times and the learning rate satisfies certain requirements. However, this only applies to situations with exactly one learning agent. In this thesis we want to analyse Q-learning under multiplayer settings. For a deeper insight into the theoretical convergence behaviour of Q-learning also see Tsitsiklis (1994), Bertsekas and Tsitsiklis (1996) and Borkar and Meyn (2000).

Q-learning has been used to analyse complex markets under multiplayer settings before. Many of these Q-learning simulations had the aim to compare different pricing concepts. For instance, Xiong et al. (2004) compared uniform pricing and pay-as-bid pricing for electricity markets. Other papers discus Q-learning's ability to find Nash equilibria. Naghibi-Sistani et al. (2006) and Krause et al. (2005) were able to find Nash equilibria with their simulations. We will take a closer look on their implementations in chapter 5.2.

There are different possible strategies for a player to select an action for each iteration. To find the best possible strategy in a given experimental set-up, the agents need to explore the set of available actions. This is achieved by choosing random actions first and learning their value over time. On the other hand, players have to exploit an effective strategy, after some time, in order to generate profit. This balancing problem is called the exploration-exploitation trade-off. Both, exploration and exploitation, are necessary for players to be effective and need to be coordinated. At the beginning of the experiment exploration should dominate, since the players

have gathered little information yet. In later stages, exploitation is more relevant to maximize profits and to allow converging to a final strategy.

2.3 The electricity market

Starting in 1996 the European electricity market liberalization was initiated with the aim to foster competition in the previously quasi-monopolistic electricity sector. The means to do this were to separate companies along the main electricity value chain (Hofbauer, 2006). Distribution and transmission were organizationally split from energy generation and retail supply, since the electricity grid is a natural monopoly while generation and retail are not. Under the new market regulations, all power plant owners, irrespective of their size, are able to offer their electrical power on the market and consumers are able to choose their provider individually (Weidlich, 2008).

In Austria, the Energy Exchange Austria AG (EXAA) is operating the main power exchange market. Different forms of power markets exist where various forms of exchange contracts are negotiated. One important type are exchange-traded day-ahead contracts in which certain amounts of energy are delivered for a specific time of the next day. Together with intraday and balancing markets these day-ahead contracts are traded on the so called spot-markets. Single hours, or even quarters of an hour, can be traded. Block bids, for subsequent hours, are common as well (EXAA, 2019). A strongly simplified version of a day-ahead market will be considered in this thesis, as it is the most liquid electricity spot market.

Other products traded at the power exchange are derivatives like futures and forwards. These are contracts for the future delivery of a specified quantity of energy for a certain time and location. The main difference between spot markets and future markets is the duration and time till contract expiry. While for spot markets delivery is typically at most for the next day, future markets consider time spans from one month to years ahead. Changing weather conditions, fuel prices, availability of generator capacity and other production costs can lead to high price volatility on spot markets. Futures and forwards are an option for generator owners to decrease long-term risks on the volatile electricity market (Weidlich, 2008).

On Austrian and German day-ahead markets prices are determined by uniform pricing. That means that the price for every market participant is set by the most expensive bid, which is still necessary to cover the current demand completely. Thus, the day-ahead market can be perceived as a game in which the outcome of one player's action depends on other player's actions (Tierney et al., 2008).

3. Specification of Q-learning

In the following chapter the concrete implementation of the Q-learning algorithm for this problem is shown. Initially, we introduce the model environment and explain all necessary steps for one iteration of the Q-learning algorithm. At the end of the chapter, hyper parameter settings and quality parameters are discussed.

3.1 Simulation model design

Any Q-learning algorithm is placed within its own model environment. This environment is a simplification of the real-life phenomenon to be simulated. The goal of our Q-learning algorithm is to optimize player behaviour in a strongly simplified electricity market.

The electricity market bidders are labelled as players. Each player has a certain number of power generators with a specific maximum capacity and marginal generating costs. These players are bidding on a uniform day-ahead-market with the goal to maximize their profits. We decided to simulate a uniform market, since it is the dominant auction form in European day-ahead electricity markets right now.

The duration of one experiment is predefined by a given number of iterations. Each iteration covers one bidding round. The current electricity demand for each iteration is defined by a perfectly inelastic constant demand function. Generator capacity and costs stay constant for the whole experiment. Players are only allowed to bid the full capacity of a generator.

Our model is a highly simplified version of a real power market. It only contains fully flexible thermal power generators, and no storage hydro reservoirs of pumped storage hydro power stations. Also, there is no nodal pricing implemented (comparable to some European markets without nodal pricing, such as Germany or Austria). Implementing storage and inflexibilities into thermal power generators (such as combined-heat and power plants, ramping restrictions and minimum generator constraints) would have made the simulation unnecessarily more complex.

3.2 Players and their properties

In the introduced Q-learning algorithm agents represent players on the electricity market. A player's behaviour is depending on properties which can be made heterogenous in the simulation. In our simulation, players' properties are described by their thermal generators' characteristics (i.e. costs and capacities). The players' behaviour is embodied in their selection of actions. All possible actions for one player are available in her action-set. The size and the elements of the action-sets of different players can differ within one experiment.

We experimented with diverse numbers of players for different set-ups. Small experiments with two or three players seem to be best suited since they allow to derive equilibria also by brute force methods.

Brute force methods were necessary, because a literature research did not yield non-trivial multiplayer games with clear theoretical predictions, suitable for the purpose of this thesis. Indeed, a considerable part of the literature focus on two player games. Already, three player games are studied much more rarely. There are some N-player games that do not specify a specific number of players, but frequently this generality comes with a simplified problem set-up. Games that have been specifically designed with specific but low number of players larger than 3 are particularly hard to find.

Bigger set-ups tend to become intractable and are therefore not used in our analysis.

3.3 Actions and states

In order to create a functional Q-learning model, players need to take actions. Available actions for one player are defined by her discrete action-set. The size of the action-set depends on the game. For simple discrete models two actions can be sufficient. For example, a coin flip game with the actions: choose head or choose tail. In this case using more actions would be meaningless. For more complex games, the right amount of actions may be difficult to define. A high number of possible actions enhances the authenticity of a model, since continuous real-world problems offer an infinite number of actions. However, the implemented Q-learning algorithm can only

represent discrete actions steps. Thus, a reasonable simplification is necessary, since a large action space increases the size of Q-tables and therefore computational time immensely.

In the presented Q-learning model, players' actions represent price bidding combinations. Every player can choose a price for each generator she owns. The minimum price equals the generator's marginal costs for one unit of produced energy. The maximum price is the same for all generators. It equals a multiple of the minimum price of the most expensive generator in the game. The total number of potential price bids can be varied from experiment to experiment, but is the same for all generators. All price steps are evenly distributed in between minimum and maximum price. Table 1 presents possible price bids for two different generators and 6 allowed prices.

Table 1: Possible price bids for two generators with different marginal costs and six allowed prices

	Marginal costs	Possible price bids
Generator 1	*50*	50, 72, 94, 116, 138, 160
Generator 2	80	80, 96, 112, 128, 144, 160

For players with one generator the number of possible actions equals the number of allowed price bids per generator. Players with multiple generators can choose any combination of generator price bids. The number of possible actions per generator A_{Gen} and the number of total player actions A_i for a player i with n generators obeys to the rule:

$$|A_i| = |A_{Gen}|^n \qquad (2)$$

Due to the exponential characteristic of this relation the number of player actions increases very fast for high numbers of generators and generator actions.

In many games, agents face changing states of the environment. The best action for state A is in many cases a bad action in another state B. Therefore, it is reasonable for an agent to distinguish between different states. For Q-learning, there are no exact rules what a state may represent. The algorithm implemented in this thesis defines states as different demand levels within a certain range. The number of states can be

varied from experiment to experiment and needs to be appropriate for the given demand function. For a constant demand, over all iterations, one state is adequate. For changing demands an increased number of states will enhance the players' performance. Since we want to analyse very basic market situations a constant demand and therefore a single state is sufficient. For situations with changing demands the state segmentation would be evenly distributed in between the minimum and the maximum value of the demand function. For instance: The demand alters between 100 and 150 units of energy per iteration. Five different states are allowed. Table 2 shows the state segmentation for the given case.

Table 2: State segmentation for a demand in range 100-150 units of energy and five allowed states

State	Range of demand [Units of energy]
State 0	100 - 110
State 1	111 - 120
State 2	121 - 130
State 3	131 - 140
State 4	141 - 150

Agents remember past market outcomes by evaluating action-state pairs depending on the received reward. The evaluation value is referred as Q-value $Q(s, a)$, where s represents the current state and a represents the chosen action. All Q-values for potential action-state pairs are listed in an agent's individual Q-table. An agent with an action-set of the size A in a system with S different states owns a Q-table with S rows and A columns. Figure 1 represents a visualized Q-table.

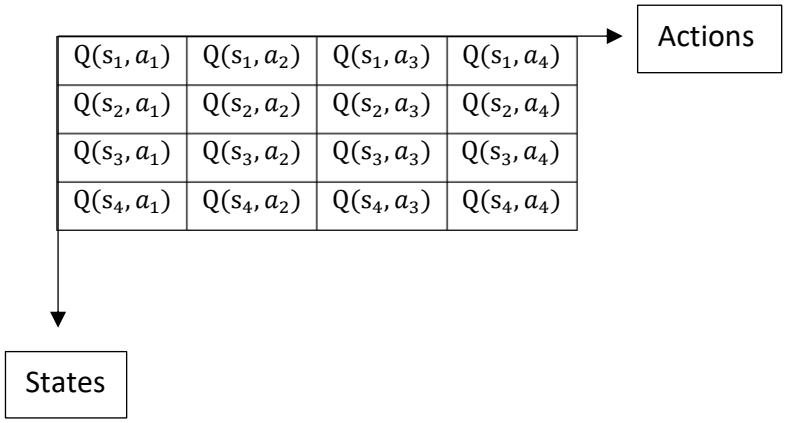

Figure 1: Visualization of a Q-table (own representation)

As mentioned before, a high number of states can improve players' decision making. However, each action has to be evaluated for each state individually. Thus, the number of iterations needed to test all combinations properly increases linearly with the number of states, if we assume that all states appear the same amount of times. Equation 3 demonstrates the linear relation between the total number of state-action pairs n_Q and number of states S for a given amount of player actions A_i.

$$n_Q = S \cdot A_i \qquad (3)$$

3.4 Description of the auction process

The auction process is the market clearing mechanism of the implemented electricity market model. It determines which generators are allowed to sell their capacity. In order to simulate a uniform market all generators are ranked depending on their price bid. The cheapest generator is ranked at first place and will sell first. This process is

repeated until the sum of electricity bids is equal to the current demand. If the remaining demand D_{rem} is bigger than the currently selling generator's i capacity Cap_i, it sells its whole generation. If D_{rem} is smaller than Cap_i , then the generator will only sell D_{rem} in order to exactly satisfy the total demand. This is the only case where a generator can sell less than its total capacity.

If two or more generators have a price bid equal to the clearing price, a tie-breaking algorithm is necessary. In this case first all generators with price bids under the clearing price are allowed to sell their capacity. Then D_{rem} and the total capacity of all generators which sell at clearing price are determined. Each of these generators is allowed to sell the same percentage share of their generation, which is equal to the proportion of D_{rem} and the calculated total capacity. That way the demand is exactly satisfied.

Once the total demand is satisfied the auction process stops and the profit Π_i for each generator i is calculated depending on its sold quantity Q_i, according to Equation 4. The final clearing price p_{clear} equals the highest selling price bid of the current auction round. c_i represents a generator's marginal costs for one unit of produced energy.

$$\Pi_i = Q_i \cdot (p_{\text{clear}} - c_i) \qquad (4)$$

We assume that unsold units of energy are not produced at all. Therefore, generators with zero sold units receive a profit of zero. The profit is directly used as reward for the Q-value updating in the last step of each iteration.

3.5 The learning process

In chapter 2.2 the Bellman equation was mentioned as part of Q-value updating. For the purpose of this thesis, the discount factor γ will be set to zero, since the outcome of one iteration does not affect the outcome of the following ones. Therefore, we adjust the Bellman equation and get to Equation 5.

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \cdot \left[r_t + \gamma \max_{a_t, t} Q_t(s_{t+1}, a_t) - Q_t(s_t, a_t) \right] \rightarrow$$

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \cdot [r_t - Q_t(s_t, a_t)] \qquad (5)$$

Other Q-learning approaches for the electricity market, as Krause et al. (2004), used the same simplification of the Bellman equation for their models. Due to this simplification experimental set-ups with a single state (constant demand) are sufficient, since the outcome for one state cannot influence the outcome for another state.

The learning process is the last step of each iteration. In our implementation, the profit calculated at the end of the auction process, is directly used as reward for the Q-value update. Q-values will increase and decrease over simulation time. A high Q-value $Q(s_t, a_t)$ indicates a high profit for choosing action a_t in state s_t.

The exploration-exploitation trade-off is coordinated by the probability variable ε, which represents the probability to choose a random action. At the beginning of an experiment ε equals 1. Over time this value decreases. Thus, the probability for exploiting the highest known Q-value rises. This sort of action selection is typically called ε-greedy. We will take a closer look on the ε-decay process in chapter 3.6.2.

3.6 Hyperparameters

3.6.1 Learning rate

The learning rate α is part of the Bellman equation (see Equation 1 and 5). It influences how strongly one single reward affects the according Q-value. The learning rate can take values from zero to one. It is necessary to find a reasonable compromise between both extremes. Important indicators, which should be considered while setting the value of the learning rate, are the number of possible actions per agent and the number of iterations. Figure 2 shows the impact of different learning rates on the Q-value over time when rewards are constant.

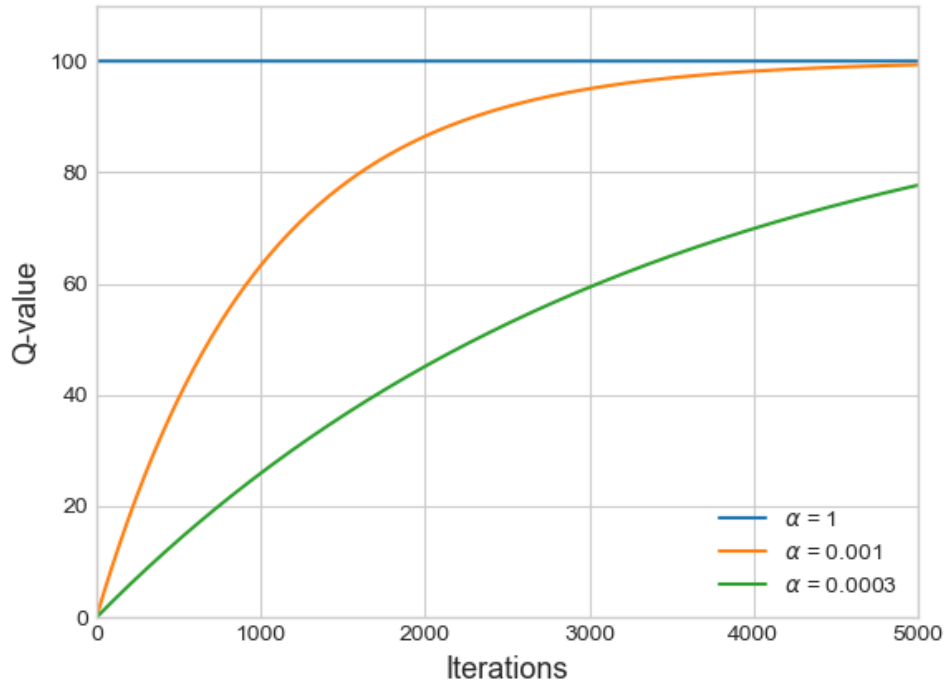

Figure 2: Change in Q-values for a constant rewards of 100 monetary units over 5000 iterations

A learning rate of 1 will push a Q-value to its maximum within a single iteration (100 monetary units in Figure 2). However, such an extreme choice leads to the algorithm completely disregarding all past observations. In contrast, small values will lead to a slow learning process over many iterations.

Using a reasonable learning rate is a key factor for generating convergence situations within a Q-learning simulation. The learning rate is directly linked to the number of iterations and number of possible actions within an experiment. Therefore, it is difficult to give general advice on how to set it correctly. For experiments with a low number of actions per generator a small learning rate is necessary in order to avoid random Q-value maximizations. For high numbers of actions an increased learning rate can reduce the required amount of iterations. We propose to use the maximization parameter, introduced in chapter 3.7.1, as an indicator for reasonable learning rates.

3.6.2 Epsilon decay rate

In any Q-learning model the exploration-exploitation trade-off needs to be balanced in order to maximize efficiency. In the implemented model the temporal distribution of exploration and exploitation is described by the variable ε which is defined by Equation 6. This equation takes on the form of an exponential decay function which provides a reasonable exploration-exploitation trade-off.

$$\varepsilon := \varepsilon_{Min} + (\varepsilon_{Max} - \varepsilon_{Min}) \cdot e^{(-\varepsilon_{Decay} \cdot i)} \qquad (6)$$

ε_{Min} and ε_{Max} represent the minimum and maximum ε can reach. If ε_{Min} equals 0 and ε_{Max} equals 1, ε represents the probability for a player to choose an action randomly. The probability to exploit the best known action is defined as $1 - \varepsilon$. With each additional iteration i, the probability for exploration decreases and therefore converging towards ε_{Min} while the probability for exploitation increases in each step. The parameter ε_{Decay} regulates the speed of probability change and needs to be balanced with the number of iterations. In Figure 3 the effect of different ε-decay-rates on the exploration-exploitation trade-off is shown.

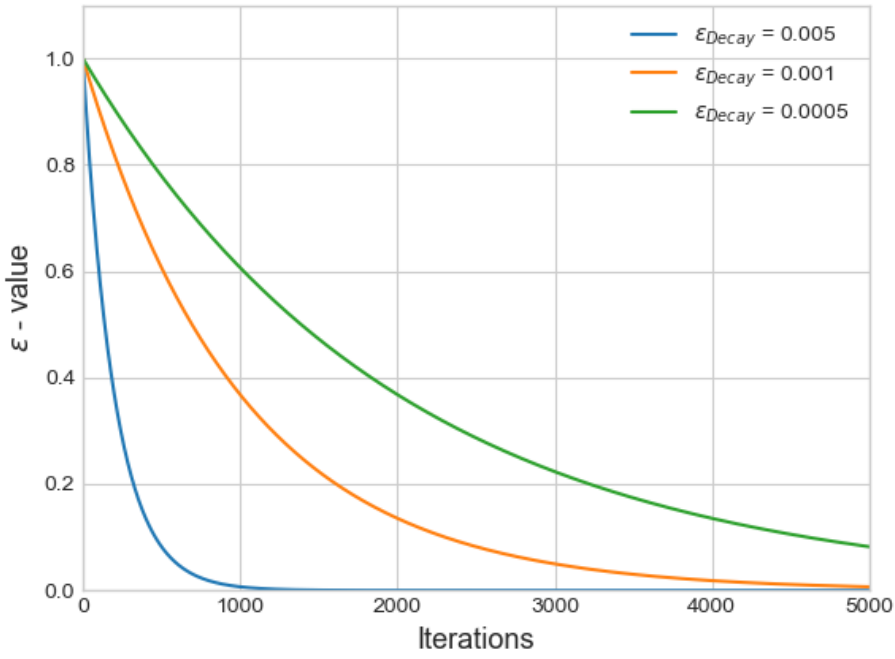

Figure 3: Effect of different ε-decay-rates over 5000 iterations.

A high ε-decay-rate leads to a shortened exploration phase, which causes incomplete information about possible actions. Thus, the probability for not representative Q-values increases. Very low decay-rates generate a shortage of exploitation. Therefore, we expect high randomness in the final stages and the experiment most likely will not converge to an equilibrium. In the course of this thesis many decay-rates were tested. Experience has shown that a balanced ε-decay-rate ε_{Decay} for I iterations can be found according to Equation 7. It provides just enough exploration time to evaluate all actions properly, at the same time randomness is low enough to enable convergence. For instance: For I = 5000 iterations we would suggest a ε-decay-rate of 0.001 (see Figure 3).

$$\varepsilon_{Decay} = \frac{5}{I} \tag{7}$$

All upcoming decay-rates in this thesis were determined according to Equation 7. So the experimental set-ups can be compared more easily.

3.7 Quality parameters

3.7.1 Possible maximizations

A working Q-learning experiment depends on many parameters, which were described in the previous chapters. Many of these parameters affect each other and influence convergence behaviour, computational time and robustness of the experiment. However, no rules for how to best determine these exist. Therefore, we defined an experiment evaluation parameter called the maximization parameter M_Q. It represents how many times a player's Q-values could be maximized each, within a given number of iterations, number of actions per generators and learning rate for a constant profit.

$$M_Q := \frac{M_{total}}{A_i} \tag{8}$$

In order to calculate M_Q the total number of possible maximizations M_{total} and the number of actions A_i per individual player i are required (If A_i is different for players, the highest value is used). M_{total} is defined as the quotient of the expected amount of exploitation iterations $E(\varepsilon, n)$ and $I_{99.5\%}$, the amount of iterations needed to raise a Q-value up to 99.5% of its theoretical maximum. Exploration is excluded since it hardly maximizes single Q-values.

$$M_{total} := \frac{E(\varepsilon, n)}{I_{99.5\%}} \tag{9}$$

For the calculation of $I_{99.5\%}$ a constant profit Π and a constant learning rate α are assumed. 99.5 percent of the maximum Q-value was chosen as a threshold since the full maximum is actually never reached due to the asymptotic character of the updating equation. The simplification of the Bellman equation, introduced in chapter 3.5, dictates the Q-value change from iteration to iteration. However, for constant Π and α we can transform the recursive character of the equation into an explicit form.

First we refactor Equation 5 in order to get to Equation 10:

$$Q(i+1) = (1 - \alpha)Q(i) + \alpha \cdot \Pi \tag{10}$$

First order difference equations of the general form $x_{n+1} = ax_n + b$ with constant a and b follow the rule:

$$x_n = \begin{cases} a^n x_0 + b\dfrac{a^n - 1}{a - 1} & , a \neq 1 \\ x_0 + bn & , a = 1 \end{cases} \tag{11}$$

In our case we have $a = (1 - \alpha)$ and $b = \Pi \cdot \alpha$. The condition $a = 1$ is only satisfied if $\alpha = 0$. A learning rate of zero is not reasonable, since no learning process is possible in this case. Therefore, cases which fulfil the first condition are more interesting in this context. At the beginning of each experiment all Q-values equate zero. Thus, we conclude $Q(0) = 0$. Substituting these values into Equation 11 results in:

$$Q(i) = (\Pi \cdot \alpha)\frac{(1-\propto)^i - 1}{(1-\propto) - 1} \tag{12}$$

After further rearranging, we receive the final explicit form:

$$Q(i) = \Pi - \Pi(1 - \alpha)^i \tag{13}$$

Equation 13 is known in mathematics as limited growth function with a certain limit and growth factor. In this case Π represents the limit and α stands for the growth factor. From here we can calculate $I_{99.5\%}$ by substituting $Q(i)$ for 99.5 percent of the growth limit.

$$0.995 \cdot \Pi = \Pi - \Pi(1 - \alpha)^{I_{99.5\%}} \tag{14}$$

$$I_{99.5\%} = log_{(1-\alpha)}0.005 \tag{15}$$

Equation 15 shows that $I_{99.5\%}$ is only depending on the learning rate and not on the profit as long as it is constant. Since the learning rate is the same for all players and is

constant for each experiment, we conclude that all Q-values of all players can be maximized within the same amount of iterations.

The exact number of exploitation iterations for one experiment cannot be calculated definitely since the exploration-exploitation trade-off is based on probabilities. However, we can calculate an expected value $E(\varepsilon, n)$. For the trade-off these possible results can either be zero or one exploitations per iteration. Therefore, we can calculate $E(\varepsilon, n)$ by adding up the probabilities for exploiting the best action of one iteration i, for an experiment with n iterations. This probability is defined as the complimentary probability of ε which was introduced in chapter 3.6.2. For Equation 16 we assume an ε_{Max} of 1 and an ε_{Min} of 0, as we did in our implementation.

$$E(\varepsilon, n) := \sum_{i=0}^{n} (1 - e^{-\varepsilon_{Decay} \cdot i}) \qquad (16)$$

It is obvious that the value of the maximization parameter M_Q does not represent a realistic number of Q-value maximizations of an experiment. However, it can help to identify unreasonable parameter settings bevor the execution of an experiment. Thus, we mainly use M_Q for experiment design. First, we decide on a maximization parameter value that we want to achieve. Then we modify the learning rate, the amount of actions per generator and the amount of iterations in order to satisfy this condition. Moreover, the maximization parameter is an option to compare the quality of experiments with different parameters to each other.

In the course of optimizing the experimental set-ups, maximization parameter values over 3.0 seem to lead to reasonable agent behaviour and convergence situations. However, there is no guarantee for a working experiment before actually running it.

3.7.2 Dominant combination stability

Another parameter we used to check the quality of an experiment is the dominant combination stability s. It is defined as the percentage of appearances of the most frequent strategy combination within the last percent of iterations. A dominant combination stability of almost 1 indicates a stable experiment state. We can assume that the converging process is completed. However, a value of exactly 1 is unrealistic

since there is still a small chance for random action selection within the last percent of iterations of the experiment. Values close to zero hint at an unstable state without convergence. In that case parameter settings need to be modified.

4. Results

In this chapter we show the results of our experiments. First, we work on a simple two player game with the aim to show the general approach. Later on, we discuss a more complex two player game. As final example we take a look at the results' quality and computational time of a symmetric two player game with a changing number of actions per generator.

4.1 Symmetric two player game with two generators

The initially presented experimental set-up considers a symmetric two player game. Both players have only one generator with identical properties. Table 3 showcases the chosen generator specifications.

Table 3: Capacities and marginal costs for a symmetric two player game

	Capacity [Units of energy]	Marginal costs [Monetary units]
Generator 1 (Player 1)	50	20
Generator 2 (Player 2)	50	20

Furthermore, we presume a constant demand of 70 energy units per auction round. The total generator capacity equals 100 units. Therefore, it is impossible for both players to sell their total load within one auction round. Thus, players are forced to compete with each other in order to maximize their profits. This happens by choosing price-policies. For this experiment both players can select one out of 7 possible price bids for each auction round. The maximal price is 40 monetary units.

The goal of this simple experimental set-up is to analyse if the action-selection, of two learning agents converges to a certain pattern after a defined number of iterations.

In order to enable convergence the number of iterations, the learning rate and the ε-decay rate need to be set properly. For small numbers of action-combinations a low learning rate is necessary in order to balance the increased probability for picking a certain action-combination randomly. High learning rates would lead to frequent random strategy changes, and therefore prohibit convergence. Considering these

facts we decided on a learning rate of 0.002. Experience has shown that this value works fine for experiments of this size.

Furthermore, the number of iterations was set to $1.5 \cdot 10^5$. Consequently, the maximization parameter M_Q equals 6.49. In other words: Each player theoretically has enough time to maximize each of her actions' Q-values 6.49 times.

The ε-decay rate equals $3.3 \cdot 10^{-5}$ and was determined by using Equation 7. Thus it is coordinated with the number of iterations and provides a reasonable exploration-exploitation trade-off.

For small experimental set-ups a closer look on the pay-off matrix can help to get a deeper understanding of the situation. Table 4 visualizes the profit for each possible action-combination for both players. The pay-offs of 6 action combinations are highlighted in blue. While playing these action-combinations a single player cannot improve her position by changing her price policy. Therefore, these action-combinations are pure Nash equilibria according to the definition in 2.1. The described experimental set-up was run 300 times.

Table 4: Pay-off matrix for both players for one auction round. Entries shown in *blue represent* pure Nash equilibria.

Player2 \ Player1	Action 0	Action 1	Action 2	Action 3	Action 4	Action 5	Action 6
Action 0	0/0	67/167	133/333	200/500	267/667	333/833	400/1000
Action 1	167/67	117/117	133/333	200/500	267/667	333/833	400/1000
Action 2	333/133	333/133	233/233	200/500	267/667	333/833	400/1000
Action 3	500/200	500/200	500/200	350/350	267/667	333/833	400/1000
Action 4	667/267	667/267	667/267	667/267	467/467	333/833	400/1000
Action 5	833/333	833/333	833/333	833/333	833/333	583/583	400/1000
Action 6	1000/400	1000/400	1000/400	1000/400	1000/400	1000/400	700/700

Table 5: Converged action-combinations of 300 runs [amount of appearances]

	0	1	2	3	4	5	6
0							84
1							69
2							1
3							
4							
5							
6	81	65					

Table 5 presents the distribution of results. Since we are considering a symmetric game we can summarize symmetric results. In doing so we find that in 165 runs one player chose action 0 and the other player chose action 6. In 134 runs one player chose action 1 and the other player chose action 6. Only in a single run the combination of action 2 and action 6 was played. All other action-combinations did not appear at all after the convergence process finished. We will have a closer look at this uneven distribution in the conclusions.

All appearing action-combinations lead to a profit of 400 monetary units for one player (action 6) and 1000 monetary units for the other one (action 0, 1 or 2). By comparing the pay-off matrices with the results' distribution we find that all 300 results are pure Nash equilibria.

In average the dominant combination stability s equals 0.9866. This indicates that there were enough iterations to finish the converging process.

Figures 4 and 5 show the change in Q-values over time for one representative run, where player 1 chose action 0 and player 2 chose action 6. This was the most frequent result. It is noticeable that the winning actions dominate other actions quite fast.

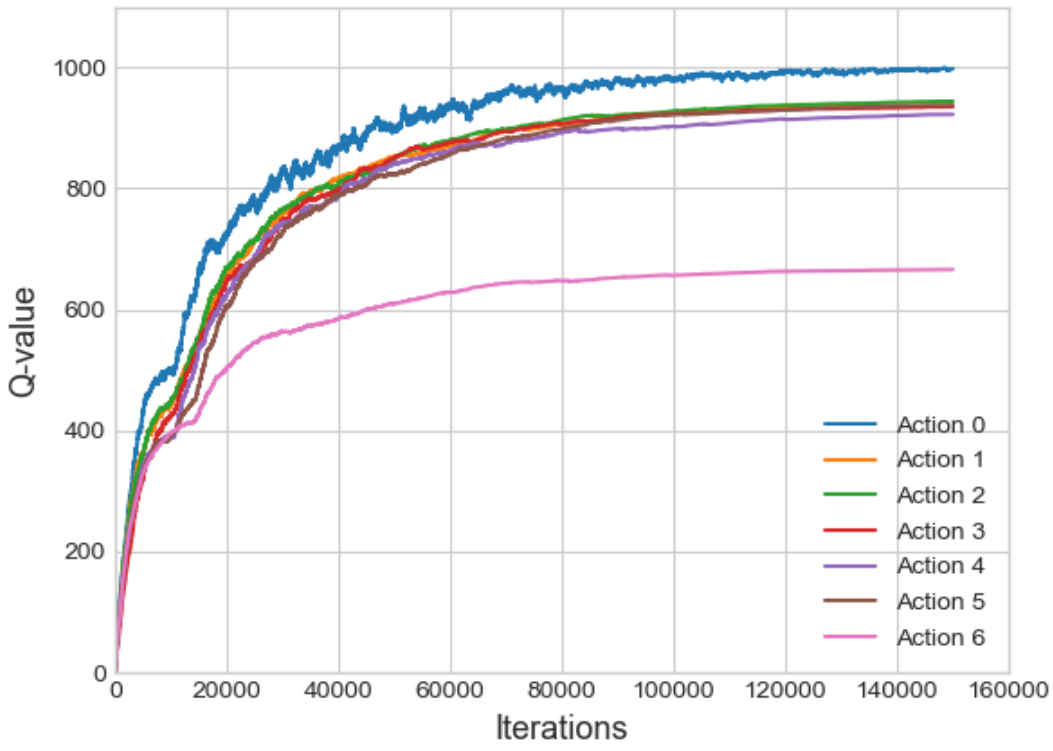

Figure 4: Q-value change over time for player 1

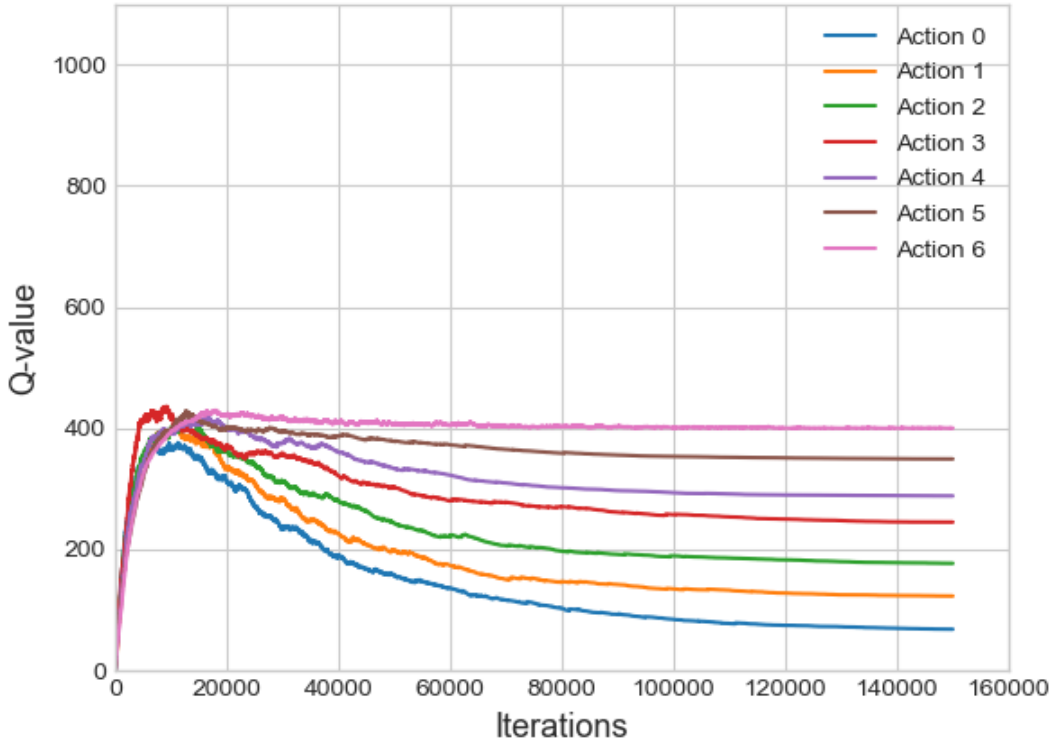

Figure 5: Q-value change over time for player 2

Figure 6 visualizes the averaged profit for each iteration for both players over time.

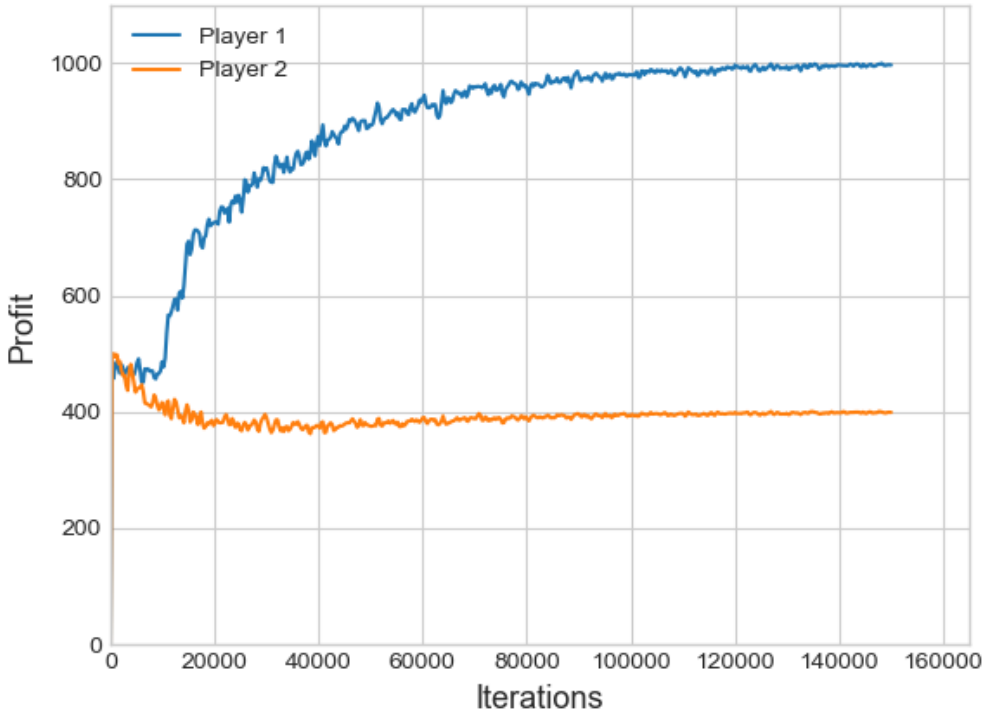

Figure 6: Averaged profit per iteration over time (moving average, window size: 100 iterations)

Player 1's (action 0) profit converges to a profit of 1000 monetary units, whereas player 2's (action 6) profit is near to 400 units. We see that both profits approach the according value of the pay-off matrices. This indicates that the experiment worked correctly.

Figure 7 presents average price bids and the change in the average clearing price. We observe that player 2 sets the bid to the highest possible value. Visual differences in the clearing price and the price deciding bid occur from the moving average, which is necessary in figure 7 for a clearly laid out presentation.

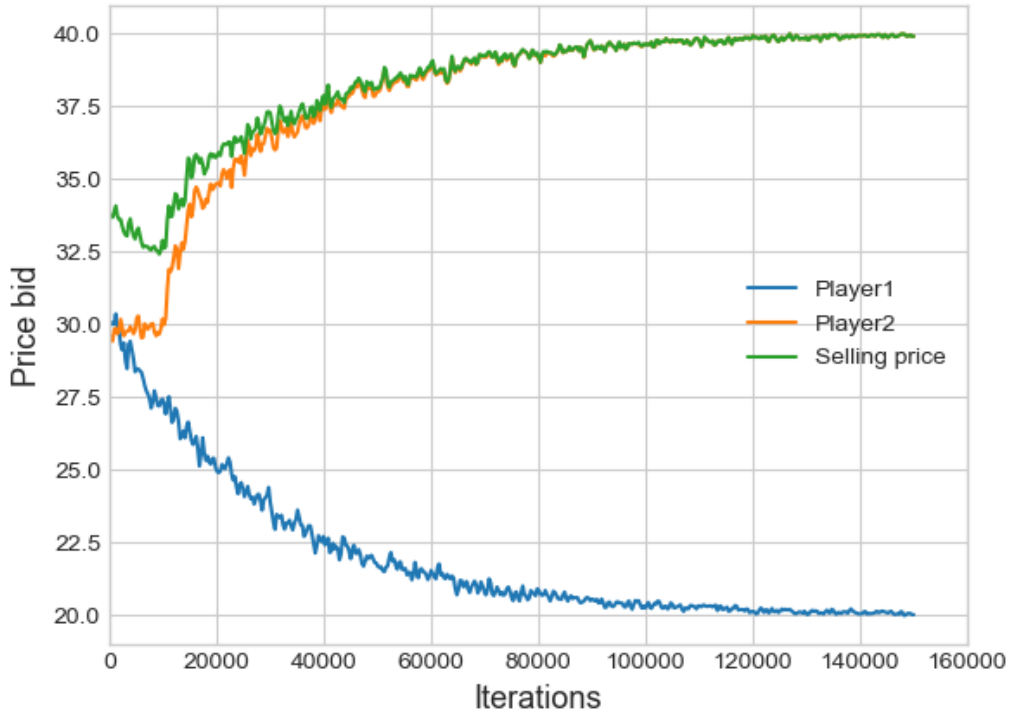

Figure 7: Price bids and clearing price (moving average, window size: 100 iterations)

4.2 Non-symmetric two player game with four generators

Now we analyse a non-symmetric game with interesting results. Both players have two generators with different properties. Player 1 owns the cheapest and the most expensive generator, player 2 owns two medium priced generators. Table 6 shows the exact generator properties.

Table 6: Capacities and marginal costs for a non-symmetric two player game

Generator	Capacity [Units of energy]	Marginal costs [Monetary units]
Generator 1 (Player 1)	50	20
Generator 2 (Player 1)	50	50
Generator 3 (Player 2)	50	25
Generator 4 (Player 2)	50	25

The demand stays constant at 125 units of energy per iteration. Each generator has 4 possible levels of bids. Therefore, each player has 16 different bid combinations to choose from. The maximum bid equals double the highest marginal costs, which is 100 monetary units.

The learning rate is set to 0.003. This quite low learning rate is needed in order to prevent random Q-value maximizations, which could happen for 16 possible action-combinations per player.

One run of the experiment has $1.5 \cdot 10^5$ iterations. Thus, we have a maximization parameter of 4.26. We can assume that all players have enough time to test their possible action-combinations properly.

The ε-decay rate, which regulates the exploration exploitation trade-off, was set to $3.3 \cdot 10^{-5}$ according to equation 7.

The experiment was repeated 300 times with the presented parameter settings. This experimental set-up has 29 different Nash equilibria. Since tractability of pay-off matrices become increasingly hard as experiments grow bigger, we will only show action-combinations which actually appeared after convergence. We distinguish between results that are Nash equilibria and results which are not.

Table 7 summarizes the outcome. In the appendix you can find an enumeration of all actions and the according prices for this set-up (Table 11).

Table 7: Winning action combinations. Nash equilibrium results listed separately

Nash equilibrium results			Non-Nash equilibrium results		
P1 action	P2 action	Quantity	P1 action	P2 action	Quantity
3	3	24	11	3	22
3	7	22	11	7	21
3	11	5	11	11	5
3	12	23	11	12	35
3	13	15	11	13	12
3	14	3	11	14	3
7	3	29			
7	7	23			
7	11	4			
7	12	31			
7	13	21			
7	14	2			
Total:		202	Total:		98

Only 67.3% of the runs ended in a pure Nash equilibrium. This is surprising since other experiments (not shown in the thesis) of the same size lead to higher percentage shares. In order to understand these unexpected results, we have to analyse the pay-offs. For this reason, we take a closer look at the profits for the case that player 1 chooses action 11 and player 2 chooses action 12. This action-combination was the most frequent result and not a pure Nash equilibrium. In this scenario player 1 makes profit of 4625 monetary units. In fact, there is no better option for player 1 as long as player 2 sticks to action 12. In this setting, player 2 earns 4687.5 monetary units. Player 2 could improve her position by changing to strategies 0, 1, 4 or 5 in order to increase her profit to 4833.33 monetary units. However, in the simulation player 2 does not change actions and misses some profit. We assume the difference between the two

profits is too small to be represented by the according Q-values properly. This can lead the simulation to remain in non-Nash equilibrium states. All occurring non-Nash equilibrium states of this experiment exhibit the same property. In the conclusion we take a closer look at this phenomenon.
On average, the stability parameter equals 0.976. Thus, we can assume that the convergence process was finished.

Figures 8 and 9 visualize the Q-value development over time for one representative run, in which player 1 chose action 11 and player 2 chose action 12. Only winning actions' Q-values are presented:

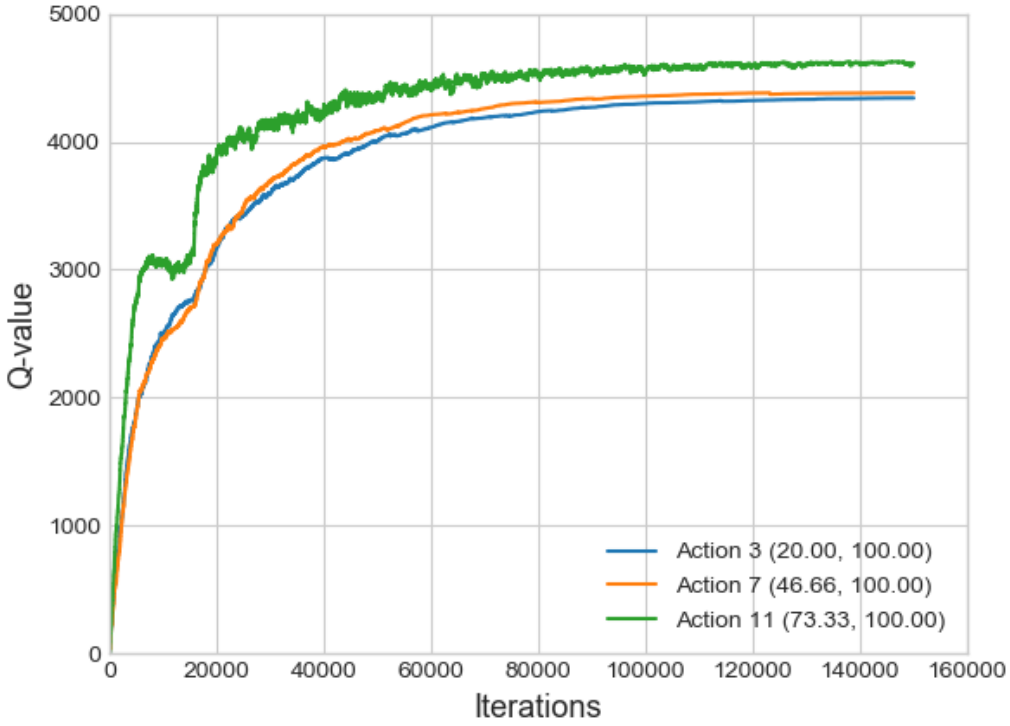

Figure 8: Change of winning actions' Q-values over time for player 1

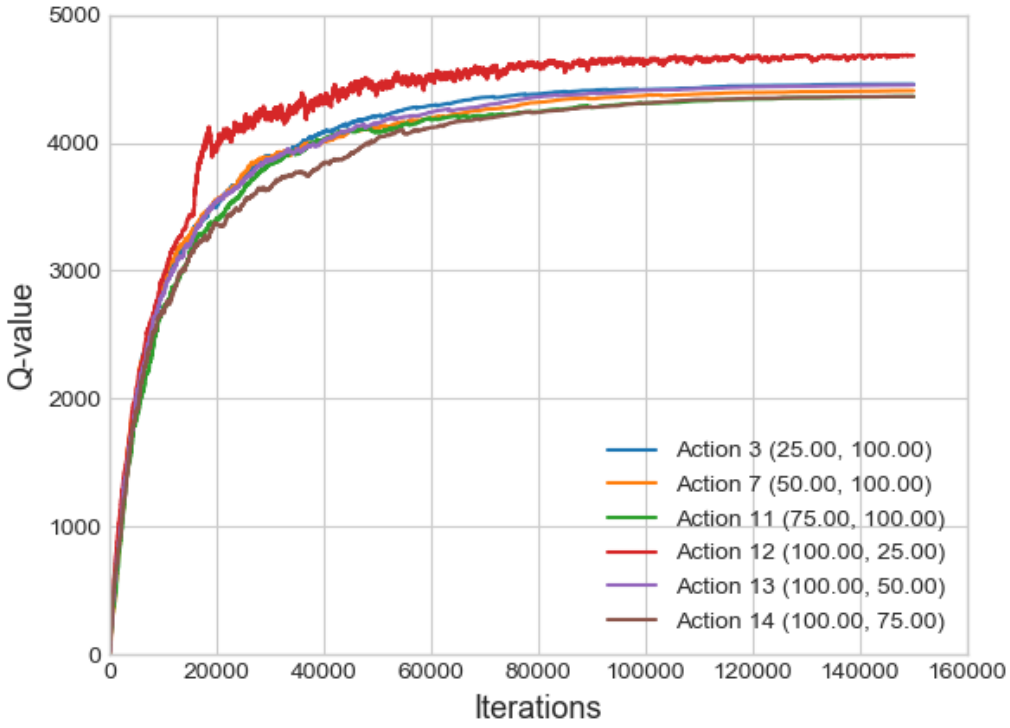

Figure 9: Change of winning actions' Q-values over time for player 2

Throughout all runs Player 1 choses three different actions (3, 7 and 11). In the presented run action 11 is dominating other actions. In all three cases generator 2 is bidding at the maximum price while generator 1 bids below it. For player 2 six different actions (3, 7, 11, 12, 13 and 14) seem to be reasonable. However, two actions are always identical, since both generators are symmetric. One of the two generators always bids at the maximum price while the second generator bids below it. In other words, both players learn to increase the clearing price by biding the maximum price with one of their generators. All converged action-combinations lead to the same profit distribution. Player 1 earns 4625 and player 2 makes 4687.5 monetary units. Figure 10 shows the average profit per iteration over time. We can observe that both profits approach the calculated pay-off values. Player 1 has a slight disadvantage in terms of profit, due to the high marginal costs of generator 2. Moreover, we find that both players strongly increase their profit per iteration around the 18,000th auction round. This happens in the period of time when the winning actions (actions 11 and 12) take the lead.

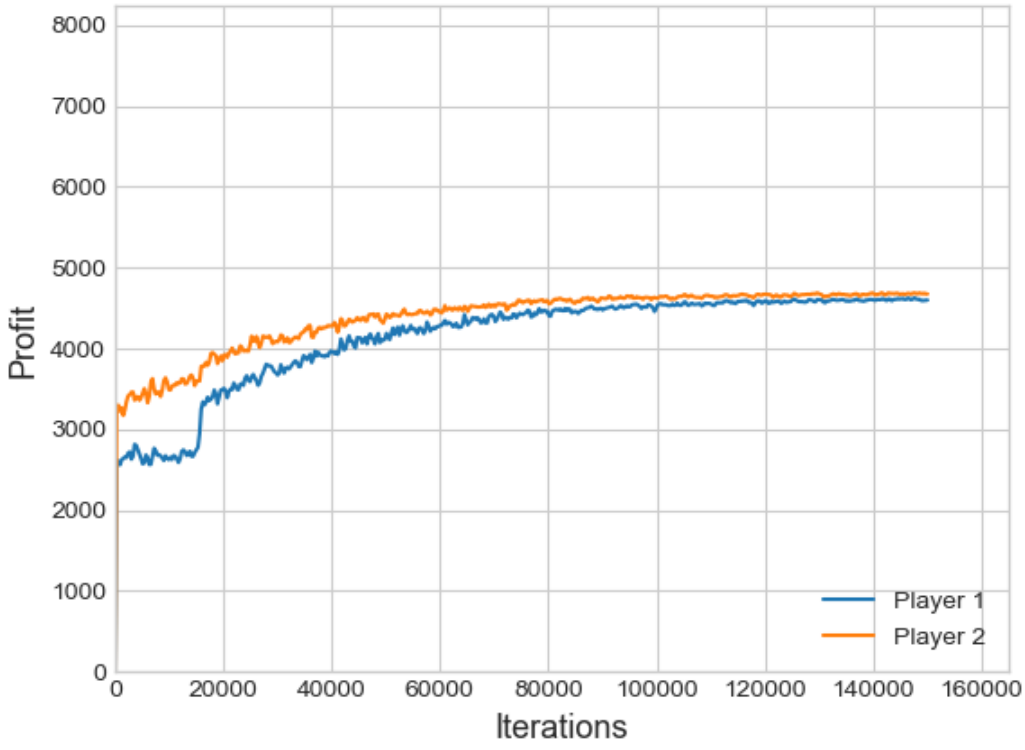

Figure 10: Average profit per iteration of both players over time (moving average, window size: 100 iterations)

Figure 11 visualizes the price bids of each generator and the clearing price with moving average. Again we find that generator 3 and 2 learn to bid at the maximum price, which is 100 monetary units, while generator 1 and 4 bid below it. This is reasonable because that way the uniform market grants all generators the maximum price. Again we can find strong shifts in bids and clearing prices around auction round 18.000.

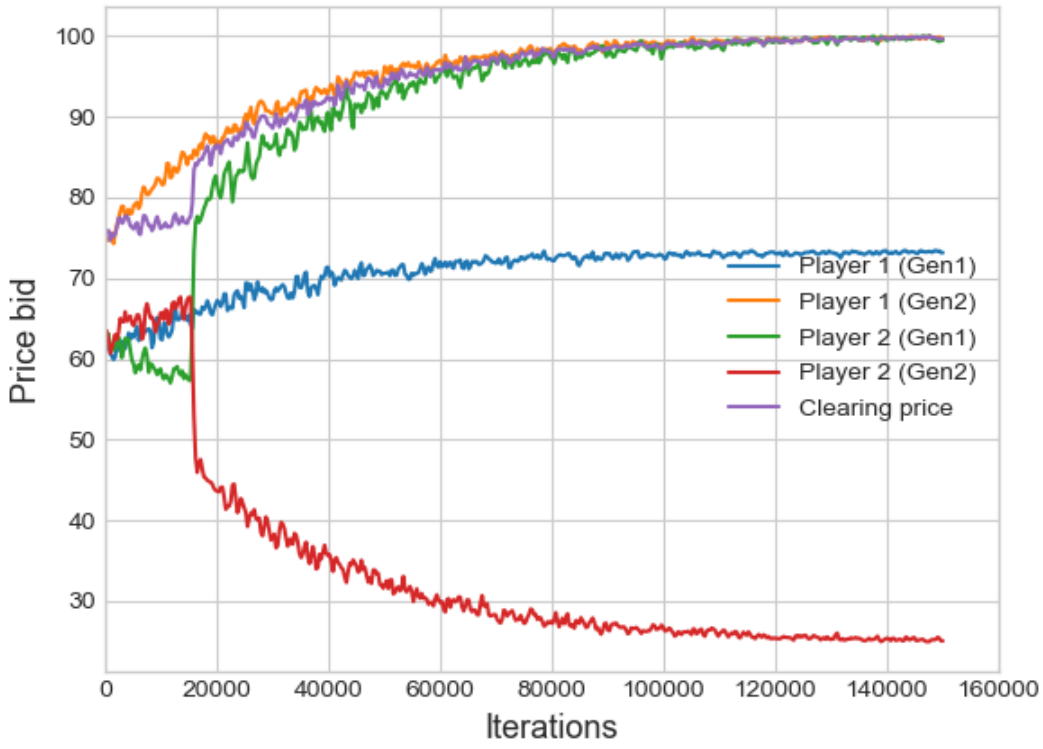

Figure 11: Average price bids and clearing price (moving average, window size: 100 iterations)

4.3 Symmetric two player games with changing actions per generator

Finally, we look at a symmetric scenario under the condition of changing parameter settings. Both symmetric players own two generators (see Table 8).

Table 8: Capacities and marginal costs for a symmetric two player game with changing parameter settings

	Capacity [Units of energy]	Marginal costs [Monetary units]
Generator 1 (Player 1)	50	20
Generator 2 (Player 1)	50	15
Generator 3 (Player 2)	50	20
Generator 4 (Player 2)	50	15

The constant demand is set to 150 units of energy per iteration. The maximal price for all generators equals two times the highest marginal costs, which is 40 monetary units. The set-up was run in 9 different modifications with a changing number of allowed actions per generator. The aim is to determine the influence of the allowed number of actions per generator on the results' quality. Table 9 shows the parameter setting for each modification.

Table 9: Parameter settings for a symmetric two player game

	Actions per gen	Total actions	Iterations	Learning rate	ε-decay rate
Set up 1	3	9	50,000	0.004	0.0001
Set up 2	4	16	75,000	0.00475	0.000066
Set up 3	5	25	100,000	0.0055	0.00005
Set up 4	6	36	125,000	0.00625	0.00004
Set up 5	7	49	150,000	0.007	0.000033
Set up 6	8	64	175,000	0.00775	0.0000286
Set up 7	9	81	200,000	0.0085	0.000025
Set up 8	10	100	225,000	0.00925	0.000022
Set up 9	11	121	250,000	0.01	0.00002

For set-ups with more than 1 generator per player the total number of allowed actions increases exponentially with increasing actions per generator. Equation 2 in chapter 3.3 describes this relation. The increasing complexity of the presented experimental set-ups is balanced with a higher learning rate and a higher number of iterations. However, the learning rate should not be set higher than 0.01 to avoid inconsistences. Therefore, for even higher numbers of actions per generator, a strongly increased number of iterations is needed to achieve satisfying values for the maximization parameter. This extends computational time massively.

Table 10 presents the resulting quality parameters, the number of runs which ended in a Nash equilibrium and the computational time for one run.

Table 10: Stability parameter (s), maximization parameter (M_Q), number of total runs, Nash equilibrium results and computational time (CT) for a symmetric two player game

	s	$\mathbf{M_Q}$	Total runs	Nash equilibrium	CT [sec]
Set up 1	0.9876	3.37	300	299	3.99
Set up 2	0.9867	3.37	300	265	6.45
Set up 3	0.9861	3.34	300	261	8.89
Set up 4	0.9859	3.29	300	259	11.39
Set up 5	0.9861	3.25	300	260	14.55
Set up 6	0.9860	3.22	300	261	17.71
Set up 7	0.9859	3.19	300	283	21.34
Set up 8	0.9861	3.16	300	194	26.08
Set up 9	0.9861	3.14	300	189	30.57

The maximization parameter M_Q was used as an indicator for the set-up design. We tried to keep M_Q higher than 3 for all set-ups. The averaged stability parameter s is between 0.98 and 0.99 for all set-ups. Thus, we can assume that the convergence process has finished.

Figure 12 visualizes the change in computational time. The number of iterations has the main influence on computational time. Due to the constantly increasing learning rate and constantly increasing number of iterations we find an almost linear increase

in computational time. However, since we do not want to increase the learning rate any higher, we expect exponential growth in computational time for numbers of actions per generator higher than 11.

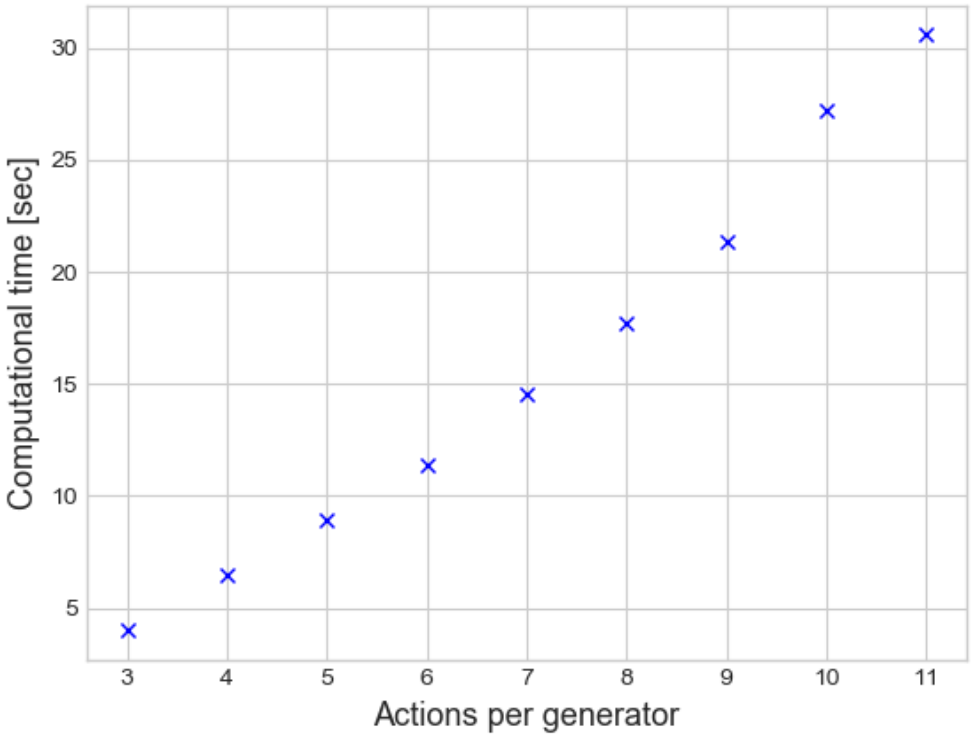

Figure 12: Computational time for different actions per generator

The amount of runs which ended in a Nash equilibrium are shown in Figure 13. Only the set-up with 3 allowed actions per generator reaches nearly 100% of Nash equilibrium results. All other set-ups show at least some non-Nash equilibrium results. In general, it appears that a higher number of actions per generator increase the total number of existing pure Nash-equilibria but lowers the amount of runs which end in a Nash equilibrium. However, set-up 7 reaches 94.3% Nash equilibrium results, which is more than most other set-ups with less actions per generator do. Therefore, we assume that not only complexity but also the individual characteristics of each set-up can influence the amount of Nash equilibrium results found. In the conclusion we discuss what these characteristics might be.

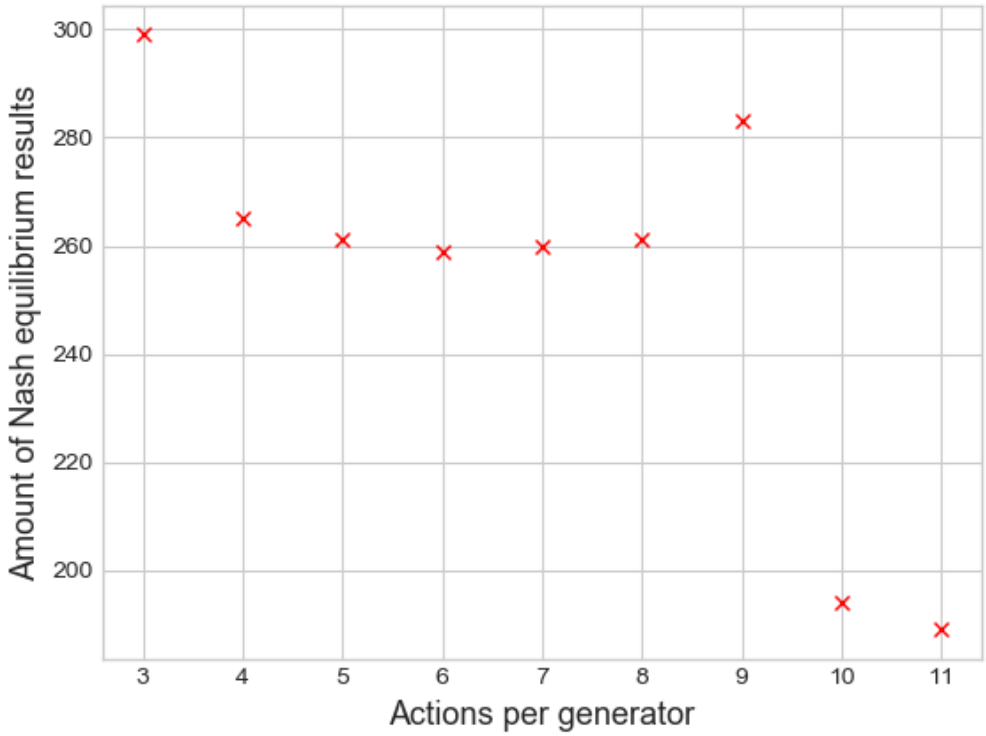

Figure 13: Amount of Nash equilibrium results (out of 300 runs) for different actions per generator

To stay concise, we do not show the concrete distribution of winning action combinations for these experimental set-ups. Nevertheless, it is necessary to mention, that the Nash equilibrium results are again unevenly distributed among possible pure Nash equilibria. Another fact we are going to discuss in the upcoming chapter.

5. **Conclusion**

In this chapter we will first explain why the found Nash-equilibria are not uniformly distributed. Second, we will compare our results to other applications of Q-learning in similar settings. Finally, we will point out limits and weaknesses of the algorithm.

5.1 Understanding the frequency of Nash equilibria found

As we saw in the first presented example, the implemented Q-learning algorithm is indeed capable of finding Nash equilibria for simple experimental set-ups. However, we find that not all existing Nash equilibria are found with the same frequency. Some equilibria do not occur at all. We assume that the uneven distribution of results is due to the specific characteristics of an experiment's pay-off matrix in conjunction with the used Q-learning algorithm. In the following we present some of these characteristics.

The first influencing factor we found is the average profit of an action. The average profit $\overline{\Pi}_a$ is calculated by averaging the profits of all possible combinations including action a. $\overline{\Pi}_a$ represents an action's average profit as long as all possible combinations are played the same amount of times. At the beginning of an experiment's run this condition is approximately satisfied, since all actions are chosen randomly with the same probability. Thus, we assume that actions with high average profits are more likely to build up strong Q-values and dominate the random early stages of the game. Indeed, we find that actions with high average profits tend to dominate the exploration phase. As we saw in the first and second presented experimental set-ups, winning strategies are taking the lead quite early (see figures 4, 5, 8 and 9). This advantage seems to be enough to dominate many runs.

Further, we found that the average profit of an action does not only influence the early stages of an experiment, but can also affect the late Q-value maximization. This is especially the case if the profits of two action combinations are similar, like it is the case in the second game presented. Intuitively one would guess that the average profit is not relevant in exploitation dominated stages of an experiment, since random

actions selection happens rather rarely. However, we have to consider that almost maximized Q-values grow very slow for a constant profit, but can shrink rapidly for a single low profit. This is due to the limited growth characteristic of the up-dating formula. Thus, we conclude that actions with a slightly better pay-off but a worse average profit are not able to reach the theoretical limit of their Q-values since they are more likely to receive random low profits. This is the reason why the algorithm cannot find Nash equilibria in some cases in the second presented experiment.

Beside the average profit an action's stability can influence the distribution of results. The stability can be described by the difference between an action combination's maximal Q-value and the maximal Q-value of the best response action for the current situation. For small differences it takes the algorithm longer to find the best response action than for big differences since the best response Q-value needs to be higher in order to surpass the current Q-value. Due to the variable stability, some actions are more likely to be retained. Therefore, best response actions and consequently Nash equilibria are the most stable configurations in the first place. In the first presented symmetric game this phenomenon can be observed. The average profit of the strongest actions (0, 1 and 2) is roughly the same. Thus, we assume that these actions win in the early exploration phase approximately the same amount of times. However, action combinations [0 6] and [6 0] are significantly the most frequent results since 0 is the most stable action and forces the other player with a higher action to change to action 6. In the end, action 2 is almost never winning, because its stability is lower than the stability of action 0 and action 1.

The third shown game presents the same experimental set-up for different numbers of actions per generator. As we expected, the computational time increases with rising actions per generator. By increasing the learning rate at the same time the amount of additional iterations needed can be limited. However, the learning rate can only be raised to a certain point in order to avoid disruptive randomness. Therefore, computational time grows exponentially for higher numbers of actions per generators and numbers of generators per player.

On average, we find that a higher number of possible price bids leads to a lower number of Nash equilibria being identified, but there seem to be other influencing factors too. Especially set-up 7 surprised with high numbers of Nash equilibrium

results. We suppose this is the case because of favourable pay of matrices. Factors like average profit and stability of actions seem to benefit Nash equilibrium results in this set-up.

It is worth mentioning that even though the maximization parameter M_Q is approximately the same for all set-ups in the third presented experiment, the amount of Nash equilibrium results differ. This is the case since M_Q only considers the theoretically possible Q-value maximizations and not the specific characteristics of pay-off matrices in conjunction with the used Q-learning algorithm. Therefore, we conclude that the maximization parameter is useful for creating comparable set-ups but can't predict the amount of Nash equilibrium results. Moreover M_Q can be calculated easily, whereas determining pay-off matrices for complex games with multiple players and actions can be very time consuming.

5.2 Comparison to similar simulations

Q-learning has already been used multiple times in electricity market analysis over the last years. Nevertheless, it is difficult to find comparable work due to the high number of specifications possible for Q-learning. Krause et al (2005) used a similar approach. Their approach is best comparable to the algorithm we implemented. Players choose mark-ups on their clearing price in order to maximize their profit. The model considers 3 players with ether 3 or 4 possible actions. As mentioned before, the same simplification of the Bellman equation was used for Q-value updating. Further, they used a ε-greedy action selection with a constant ε, whereas we chose to use a ε-decay function. Krause et al (2005) included a restricting grid into their model and analysed simple games with one or two Nash equilibria. They found that agents are able to find Nash equilibria quite fast for small set-ups. For set-ups with two Nash equilibria they found cyclic behaviour. For a constant ε we can confirm their findings.

Naghibi-Sistani et al. (2006) managed to produce similar results. In their model two players can choose three different slopes for their linear bid supply function. Two states represent high and low production costs. The action selection differs from our model. Here a softmax action selection rule was used. This type of action selection

uses the Boltzmann distribution and a temperature-parameter. It favours strong Q-values even in the exploration phase. In this model players' action combinations converge into a Nash equilibrium as well.

Xiong et al. (2004) designed a similar model as well. Since the aim of their work was a comparison of pay-as-bid and uniform market, it is not beneficial to compare results but the model's structure. They defined their Q-learning system's states to be last round's clearing price. 10 agents are able to choose one out of 21 different prices. The agents' goal is to maximize profit and reach a certain utilization rate. Both factors are considered in the Q-value updating. Again, a ε-greedy action selection strategy was used. Similar to our model generators are only allowed to sell their full capacity. Set-ups were run with a price-inelastic and a responsive demand function.

5.3 Limits and weaknesses of the implemented Q-learning algorithm

First, it is necessary to mention that the implemented algorithm is a rather simple Q-learning system. It does not fully utilize the possibility to take rewards of future actions in consideration. This could be a connection point for further improvement of the algorithm.

In chapter 4.3 we show that the algorithm finds fewer Nash equilibria with increasing complexity. Especially set-ups with multiple generators per player grow rapidly for an increasing number of actions per generator. Thus, the implemented algorithm is not appropriate to simulate set-ups with more than three players or more than 100 actions per player. We assume that the quality could be improved by an extended learning phase with more iterations. However, this also increases computational time. Since pure Nash equilibria can be found faster by brute force calculations, the algorithm should not be used to find Nash equilibria for a certain game, but rather to analyse Nash equilibria distribution over many runs**.**

A general problem is the double impact of the learning rate α on results. On the one hand α needs to be low in order to prohibit random Q-value maximizations, on the other hand α should be high in order to detect small differences in possible profits

fast. For complex experimental set-ups it seems that a constant α can't do both at once.

The implemented Q-learning algorithm is not able to find mixed strategy Nash equilibria due to the fact that each agent is restricted by a single winning action. To get a better insight into the full spectrum of pure and mixed Nash equilibria other reinforcement learning approaches like the Erev-Roth algorithm, which assigns probabilities to all actions, could be used (Erev and Roth, 1988).

6 Outlook

In the course of this thesis we confirmed that basic Q-learning algorithms are able to find pure Nash equilibria of simple market situations. Moreover, we found certain characteristics of pay-off matrices which influence the convergence behaviour. At the same time we analysed weaknesses and limits of the system. Furthermore, we showed that computational time is only a limiting factor for large experimental set-ups. Still, the amount of Nash equilibria found decreases with the number of players and number of actions per generator.

Q-learning is an attractive method to simulate game theoretical problems. Even though many different Q-learning specifications have been used in the past, we think that there is still potential for improvement. This thesis can be seen as foundation for prospective work with similar simple Q-learning algorithms. For future enhancement we advise to remodel the use of states. Moreover, a more flexible use of learning rates could be beneficial. Changing the action-selection rule would be interesting too.

Especially interesting are the uneven distributions of Nash equilibrium results and the influencing factors behind it. A deeper look in these factors can help to understand and improve Q-learning mechanics. Whether the influencing effect of average profit and the stability of an action can describe real life behaviour of market participants is a question worth considering.

7 Appendix

Table 11: Possible actions and according prices for a non-symmetric two payer game (chapter 4.2)

Action	**Player 1**		**Player 2**	
	Generator 1	Generator 2	Generator 3	Generator 4
0	20.00	50.00	25.00	25.00
1	20.00	66.66	25.00	50.00
2	20.00	83.33	25.00	75.00
3	20.00	100.00	25.00	100.00
4	46.66	50.00	50.00	25.00
5	46.66	66.66	50.00	50.00
6	46.66	83.33	50.00	75.00
7	46.66	100.00	50.00	100.00
8	73.33	50.00	75.00	25.00
9	73.33	66.66	75.00	50.00
10	73.33	83.33	75.00	75.00
11	73.33	100.00	75.00	100.00
12	100.00	50.00	100.00	25.00
13	100.00	66.66	100.00	50.00
14	100.00	83.33	100.00	75.00
15	100.00	100.00	100.00	100.00

8 References

Bertsekas, D., Tsitsiklis, N., J.m 1996. Neuro-Dynamic Programming. Athena Scientifc.

Borkar, V., S., Meyn, S., P., 2000. The O.D.E. method for convergence of stochastic approximation and reinforcement learning. Society for Industrial and Applied Mathematics.

Cournot, A., 1838. Researches into the mathematical principles of the theory of wealth.

E-Control, 2019. www.e-control.at. Access date: 20.08.2019.

Energy Exchange Austria (EXAA), 2019. www.exaa.at. Access date: 21.08.2019.

Erev, I., Roth, A., E., 1988. Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria. The American Economic Review 88, 848-881.

Gibbons, R., 1992. A Primer in Game Theory. Prentice-Hall.

Hofbauer. I., 2006. Liberalisation, privatisation and regulation in the Austrian electricity sector: Country report on liberalisation and privatisation processes and forms of regulation.

Kaelbling, L., P., Littman, M., L., Moore, A., W., 1996. Reinforcement Learning: A survey. Journal of Artificial Intelligence Research 4, 237-285.

Krause, T., Andersson, G., Ernst, D., Vdovina-Beck, E., Cherkaoui, R., Germond, A., 2004. Nash equilibria and reinforcement learning for active decision maker modelling in power markets. Proceedings of the 6th IAEE EuropeanConference: Modelling in Energy Economics and Policy.

Naghibi-Sistani, M., Akbarzadeh - Tootoonchi, M., Javidi-Dashte Bayaz, M., Rajabi-Mashhadi, H., 2006. Application of Q-learning with temperature variation for bidding strategies in market-based power systems. Energy Conversion and Management 47, 1529–1538.

Osborne, M., 2003. An Introduction to Game Theory. Oxford University Press, USA.

Osborne, M., Rubinstein, A., 1994. A Course in Game Theory. The MIT Press.Cambridge.

Österreichs Energie, 2019. www.oesterreichsenergie.at. Access date: 02.09.2019.

Papageorgiou, M., Leiblod, M., Buss, M., 2015. Optimierung – Statistische, dynamische, stochastische Verfahren für die Anwendung. Springer-Verlag.

Sutton, R., S., Barto, A., G., 2017. Reinforcement Learning An Introduction. Second edition. The MIT Press.

Tesfatsion, L., 2006. Handbook of computational economics. Chapter Agent-Based Computational Economics: A Constructive Approach to Economic Theory. Agent-Based Computational Economics, vol. 2. North-Holland.

Tierney, S., F., Schatzki, T., Mukerji, R., 2008. Uniform-Pricing versus Pay-as-Bid in Wholesale Electricity Markets: Does it Make a Difference? New York ISO.

Tsitsiklis, N., J., 1994. Asynchronous stochastic approximation and Q-learning. Machine Learning, 16, 185-202.

Watkins, C. J. C. H., 1989. Learning from delayed rewards. Ph. D. thesis, Cambridge.

Watkins, C. J. C. H., Dayan, P., 1992. Q-learning. Machine Learning 8: 279 - 292.

Weidlich, A., 2008. Engineering Interrelated Electricity Markets - An Agent-Based Computational Approach. Springer Physica.

Wilkinson, N., Klaes, M., 2012. An introduction to behavioral economics. Palgrave Macmillan.

Xiong, G., Okuma, S., Fujita, H., 2004. Multi-agent based experiments on uniform price and pay-as-bid electricity auction markets. Proceedings of the IEEE International Conference on Electric Utility Deregulation, Restructuring and Power Technologies (DRPT2004), Hong Kong

9 List of figures and tables

Figures:

Tables: