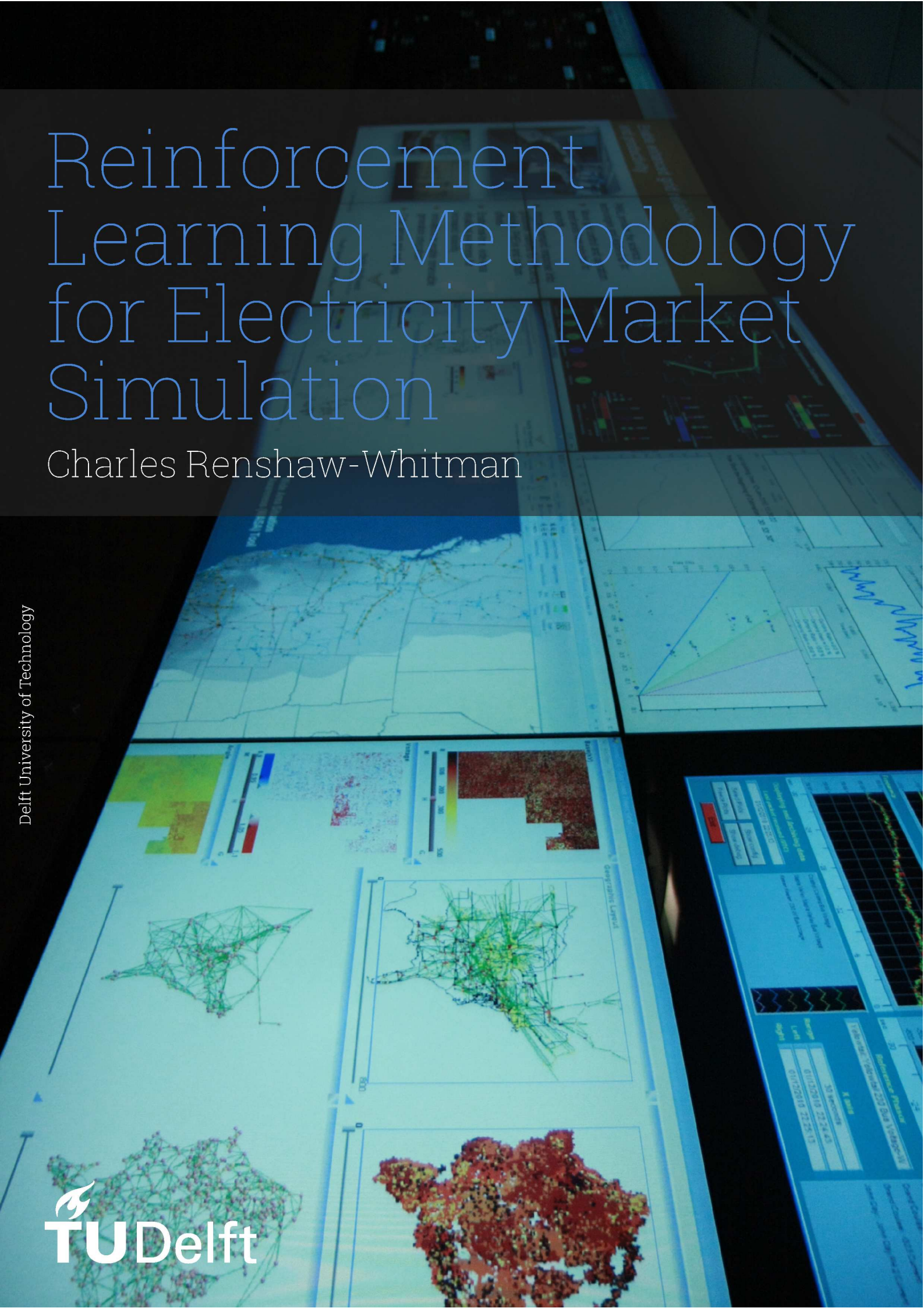
Reinforcement Learning Methodology for Electricity Market Simulation

Charles Renshaw-Whitman

Delft University of Technology



TU Delft

Reinforcement Learning Methodology for Electricity Market Simulation

by

Charles Renshaw-Whitman

Student Name	Student Number
Charles Renshaw-Whitman	5513812

Supervisor: Prof L. de Vries
Supervisor: Prof J. Cremer
Supervisor: V. Zobernig

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Preface

This work seeks to resolve an outstanding problem in the use of reinforcement-learning methods for the simulation of economically-rational agents. We discuss the problem of non-stationarity, and how this subsequently limits market simulation capabilities. After explicating and isolating the source of the problem for a day-ahead electricity market, we demonstrate the application of methods which resolve this problem in simple test-cases, and prove conditions under which similar methods will work in general. Subsequently, we illustrate how these techniques can be used to solve a restricted market-design problem, in the process introducing a framework for discussing adversarial market-design for electricity markets in general. It is hoped that, insofar as they provide a new feedback-loop for market-design, these results will facilitate the design of more complex electricity-markets suitable for the energy transition.

This work was conducted under the overall supervision of Professors Laurens de Vries and Jochen Cremer of T.U. Delft, in addition to the daily supervision of PhD Candidate Viktor Zobernig of the Austrian Institute of Technology. The author expresses his gratitude to these supervisors, as well as to his friends and family, whose support enabled the completion of this work.

Charles Renshaw-Whitman
Delft, August 2023

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
DDPG	Deep Deterministic Policy-Gradient
DSIC	Dominant Strategy Incentive Compatible
EPEC	Equilibrium Program with Equilibrium Constraints
IL	Independent Learning
MADDPG	Multi-Agent Deep Deterministic Policy-Gradient
MARL	Multi-Agent Reinforcement-Learning
MC	Marginal Cost
MDP	Markov Decision Process
MPEC	Mathematical Program with Equilibrium Constraints
RL	Reinforcement Learning
TD	Temporal-Difference

Symbols

Symbol	Definition	Unit (if applicable)
A	Action	
c	Marginal cost	[USD]
CS	Consumer surplus	[USD]
N	Number of agents	
O	Observation	
P	Price	[USD]
PS	Producer surplus	[USD]
Q	Quantity	[MW, MWh]
Q	State-action value, Reward-prediction	[USD]
R	Reward	[USD]
S	State	
SW	Social welfare	[USD]
U	Utility, Reward	[USD]
V	State value, Reward prediction	[USD]
α	Adversariality coefficient	[Unitless]
β	Boltzmann exploration parameter	[N/A]
γ	Time discount factor	[Unitless]
δ	Temporal difference (TD) error	[USD]
ϵ	Epsilon exploration parameter	[Unitless]
λ	Optimality deficit	[USD]
μ	Deterministic policy	
Π	Profit	[USD]
π	Policy	
ρ	Soft-update coefficient	[Unitless]
τ	Trajectory	
Θ	Set of hidden-information	
θ	Parameter-vector	

Notation

Symbol	Designation	Meaning
A, S, O	Capital letter	Random variable of corresponding letter
$\mathcal{A}, \mathcal{S}, \mathcal{O}$	Calligraphic Letter	Set of all (joint) objects of corresponding letter
a, s, o	Lower-case letter	Particular realization of random variable of corresponding letter
$\mathbb{E}[\cdot\|\cdot]$	Blackboard "E"	Expectation of quantity left of pipe with respect to probability distributions right of pipe
ΔS	Capital delta	Acting on arbitrary set S, The space of probability distributions over the arbitrary set S
$(\cdot)^{-i}$	Superscript "$-i$"	Object corresponding to all agents except agent i
$(\cdot)^{i}$	Superscript "i"	Object corresponding to agent i
$(\cdot)_{\theta}$	Subscript θ	Function parameterized by parameter-vector (usually neural-network weights) θ
$(\cdot)_{t}$	Subscript "t"	Object corresponding to time-step t
$\mathcal{L}[\cdot]$	Calligraphic "L"	Loss-function of some parameterized function (typically a neural-network)

1

Introduction and Literature Review

1.1. Introduction: Market Design and Reinforcement Learning for the Energy Transition

The design of electricity markets is both increasingly important and increasingly complex as the world undertakes the transition to a low-emissions power system. Increasing penetration of intermittent renewables ensures that secondary markets in, i.a., inertia, balancing, and capacity, play an ever more important role in generators' portfolios. From a societal perspective, more complex markets impose a greater difficulty on market designers wishing to ensure the system's soundness against exploitative bidding strategies - and thus entail a risk that generators will abuse their market power at consumers' expense.

Complex physical constraints and multiple coupled markets make difficult the application of traditional optimization methods for determining the Nash equilibrium bidding strategies [1]. Recent work has applied techniques from single-agent [2] and multi-agent [3][4] reinforcement learning (RL) to address this increase in complexity.

Despite this increased interest in RL techniques, the application to market simulation (and in turn to market design) imposes new constraints on the algorithms used to simulate bidder behavior. In this work, we seek to address the problem of non-stationarity - the effective change in environment faced by an agent due to other agents' learning new strategies - for multi-agent reinforcement learning (MARL) algorithms in the context of electricity markets. This done, we go on to show how MARL methods permit the adversarial design of electricity markets, selecting an approximately optimal market-design from a pre-specified family of designs.

In this work, we examine different reinforcement learning methods in an attempt to determine when they are capable of realistically simulating strategic behavior, and subsequently examine the impact of such capabilities on market design. We analyze three RL algorithms: Tabular Q-learning, Deep Deterministic Policy Gradient (DDPG), and its multi-agent counterpart, Multi-agent Deep Deterministic Policy Gradient (MADDPG). We prove results about when value-learning methods will correctly simulate strategic behavior in multi-agent scenarios. Following this, we introduce the related concept of adversarial market-design, a novel procedure to design electricity markets which mitigate the negative effects of strategic behavior and maximize social welfare.

1.1.1. Problem and Research Contribution

Reinforcement Learning methods show much promise; having been developed on the basis of the same theory of rational agency as modern economics, they are a natural method for simulating agents' behavior in complex market situations. However, present methods struggle to obtain convergence to proper Nash-equilibria due to mis-application of the algorithms used, adding a further source of difficulty to the use of already notoriously finicky DeepRL methods.

The core research question of the present work might be summarized "**under what conditions can MARL methods be productively used to simulate rational agents' behaviour for the purposes of electricity-market design?**"

This work ultimately seeks to contribute to the field of electricity market design on two fronts. First, we offer a methodological contribution, indicating a principled way to select MARL algorithms in order to simulate economic agents' strategic behavior, and explicate the conditions under which convergence can be expected. Second, we develop new methods to design markets *which explicitly rely on the aforementioned convergence proofs*. This new tool offers a perspective combining ideas from mechanism design with those of DRL, in the hopes that it will aid computational market designers in more effectively ensuring socially beneficial market outcomes.

1.1.2. Plan of the Work

The plan of the work proceeds as follows: in this Chapter 1 we have offered an introduction to the problems with which we are concerned, which will be followed subsequently by a literature review highlighting different fields from which the present work draws. The succeeding two sections will offer an overview of game theory and reinforcement learning, resp., sufficient to acquaint the reader with the appropriate terminology and the notation used in this work. In the latter section, we also offer a few remarks on reinforcement learning algorithms, as well as the connection between reinforcement-learning and economic theory.

Chapter 2 focuses on theoretical methods used to attack the problem of non-stationarity. Section 2.1 opens by discussing the problem of non-stationarity in reference to independent-learning algorithms, also providing some examples which illustrate the problem in a traditional co-operative setting. Section 2.2 introduces the TD-error and the optimality-deficit, two keys we will use to test for convergence to true Nash equilibria. Section 2.3 provides proofs for when certain Q-learning algorithms will be guaranteed to converge to true Nash-equilibria (i.e. to bypass non-stationarity). Finally, Section 2.4 introduces the concept of "Adversarial Market Design", a new technique for designing certain types of markets which utilizes our earlier convergence results.

Chapter 3 outlines the form of the experiments which we will perform in order to examine the non-stationarity phenomenon in different environments and show how its solution makes possible new market-design techniques. We run three sets of experiments; the first, in a strictly discrete environment, is discussed in 3.1; the second, in an analogous environment where the bid-space is continuous is outlined in 3.3; this section is preceded by a brief discussion of the DeepRL algorithms we will use, DDPG and MADDPG, in Section 3.2. The third experiment set, specified in 3.4, is aimed at testing the concept of adversarial market design in a simple, illustrative case.

Chapter 4 showcases the results of each of the above experiment classes. For each experiment, two examples are shown, followed by a figure summarizing the relevant data.

Chapter 5 discusses the results of the aforementioned experiments, separately, and then comparatively. After a recapitulation of the work and its contribution to the existing literature, we close with a discussion of the applicability and limitations of the methods employed, suggesting directions for future work accordingly.

As this work focuses in part on RL methodology, there will be occasion to employ some mathematics which may at first blush appear somewhat intimidating; a typical example is an expectation with respect to some distributions which require multiple symbols to specify. While this is necessary for precision, we also wish to illustrate the fundamental economics at issue as much as possible - accordingly we have made every effort to ensure that a reader uninterested in disentangling mathematical formulations can comfortably follow the work at the conceptual level.

1.2. Literature Review

Here we present a brief summary of related research which informs the perspective of the present work. We first discuss the problems faced by electricity-market designers with a focus on research in coupled markets and trends related to the energy transition. Subsequently, we offer a broad overview of some notable applications of MARL methods in solving game-theoretic problems, and contrast these with other methods of computational game-theory. Finally, we discuss some recent applications of RL (single- and multi-agent) in the field of electricity market-design.

1.2.1. Energy Transition and Electricity Markets

The classic pedagogic text on electricity markets is [5]. [6] offers an overview of the field, discussing the key issues and the criteria by which success is to be judged - namely, the ability to "[provide] reli-

able electricity at least cost to consumers". In addition to these traditional economic aspects, market designers must increasingly design for an energy system making heavy use of renewables; an excellent discussion of the economic aspects of intermittent renewable energy sources is offered by [7]. Many of the key insights and difficulties of electricity market-design relate to the interaction between the short-term power- and ancillary- markets and the longer-term impact on generation capacity; an explication of the relevant considerations is offered in [8] in the context of forward markets. It is noteworthy that much of the work in this area is reasoned about conceptually, as opposed to mathematically. This seems, in part, a result of the fact that, while phenomena such as e.g., risk aversion, political uncertainty with respect to permitting, effects of short-term markets on long-term investment etc. can be handled mathematically, this may be done only at the cost of introducing complex models which themselves entail many assumptions. Ideally, mathematical frameworks like that of mechanism design would complement and synergize with the intuitions of economists and policymakers.

1.2.2. MARL Methods in Game Theory and Alternative Approaches

One of the core perspectives from which this work draws is that of game theory. In particular, one understanding of MARL methods is as a general solution technique for games. One class of algorithms with which MARL methods bear comparison is direct-optimization models, which seek to explicitly formulate and solve the mathematical conditions governing optimal agent-behavior; an introduction to such equilibrium solution methods in electricity-market contexts is offered in [9]. Broadly speaking, RL methods may be thought of as very general approximate optimization algorithms - typically, bespoke optimization algorithms perform better than RL within a limited domain, and fail entirely when certain assumptions are not met (e.g., convexity of a reward function); by contrast, RL algorithms suffer from a variety of reliability issues, and do not always have practical guarantees of achieving the true optimal solution, but can be readily employed in complex domains where "good-enough" solutions can be highly valuable.

Game-theory provides an effective framework for discussing problems in MARL - when agents do not share a common reward function, the language of games offers a useful way to understand the incentives facing an agent. A general discussion of the relationship between MARL and game-theory can be found in [10] - the joint-policy-correlation-matrices introduced by the authors' offers a different perspective on very similar issues to those which we discuss here, focusing on the performance of independent-learners trained in different experiments. A broad overview of classical MARL methods is offered in [11], and a more specific treatment of the non-stationarity problem with which we are concerned here is offered in [12]. These works introduce many techniques which have been used to coax MARL agents into solving games to Nash-equilibria; the most principled being the use of other-agent-modelling, which inevitably leads to violating the assumption of hypothesis-realizability underlying RL (though, as discussed below, frameworks like Infra-Bayesian probability have been set out to counteract this issue).

As game- and decision-theorists' models work by making assumptions about agents' behavior, one of the great problems of multi-agent theory is how to model agents reacting to other agents - exotic problems in this domain include the "Open-source prisoner's dilemma" (in which agents are possibly identical AIs with access to each others' source-code, and must choose to co-operate accordingly), as well as Newcomb's problem (in which a superintelligent oracle places $1 in the first of two boxes, and $1000 in the second if and only if it predicts the agent will take only the second box, otherwise leaving the second box empty) [13]. As noted above, one approach is to model agents as having models of other agents (having models of other agents having ... *ad infinitum*); such an approach has been combined with modern DRL methods in [14]. This necessarily entails that an agent model something at least as complicated as itself - an impossibility known as "hypothesis nonrealizability". A fascinating, if inscrutable, alternative is that of Infra-Bayesian decision-theory [15], which explicitly understands both agents and opponents as physical subsets of the environment, and also provides an extensive mathematical framework for decision-making in non-realizable hypothesis spaces. Given the diversity of the work in this direction, we attempt in this work to deal as much as possible with situations in which agents need not directly respond to one another, in order to highlight the core aspects of the problem.

Recent high-profile applications of MARL methods to game-theoretic problems include [16], in which is introduced the MADDPG algorithm which plays an important role in the present work, as well as [17], which combines DeepRL with game-theoretic methods to solve the imperfect-information game of Stratego.

Ultimately, just as economists use game theory to model economic agents as approximately rational, MARL methods in some sense generalize this to offer an account of how rational agents can learn about a multi-agent environment well enough to play effectively. This entails two possible use-cases: MARL methods can be used to simulate the learning procedure of actual actors facing an uncertain strategic environment (this is a relatively rare approach, but see [18] for a relevant discussion, albeit only with mostly implicit connections to RL). Alternatively, they may be used to generate possible (approximate) equilibrium strategies in environments which are too complex to "solve" with the tools of game theory; it is the latter use-case with which we principally concern ourselves in this work.

1.2.3. RL In Electricity Market Design

Within the field of electricity market-design, the most common use of RL and MARL methods is for the simulation of economic agents' strategic behavior - a common use-case is to build a model of some portion of an electricity market and then allow RL agents to operate one or more of the market-participants, with the aim of determining how vulnerable the market is to strategic behavior.

An overview of the state of RL methods in electricity markets is presented in [19]; the authors discuss a number of single- and multi-agent algorithms for doing RL-based analysis of electricity markets.

More directly antecedent to the present work, [1] compare the performance of bi-level optimization algorithms with a DeepRL DDPG agent. A comparison of various bidding strategies, DDPG, and MADDPG is performed in [20], which finds MADDPG to well-approximate the exact solution of a given MPEC, while generally learning more efficiently than DDPG. [4] presents a case study utilizing MARL methods to identify cases in which agents are able to exercise market power thus informing the design of a balancing-market - notably, in this case, the authors augment the state-space of learning agents with past market-clearing data - which data *per se*, by the Markov assumption, *cannot* be decision-relevant for these agents; instead, this augmentation artificially enlarges the state-space, implicitly permitting adaptation to other agents' changing policies (c.f. the theorems proved in Section 2.3 of the present work). While these authors verify their algorithm reproduces the Bertrand solution, an explanation for how or when such artificial augmentations work would be valuable in designing RL setups for similar applications. Finally, [3] provides an analysis of the performance of the DDPG algorithm in a market in which the analytical-solution is known, highlighting, i.a., the issue of non-stationarity which we discuss in the present work.

1.3. Game Theory and Nash Equilibria

One of the most fruitful methods of economic analysis is the theory of games. Game theory describes the strategies of rationally-acting agents seeking to maximize a "utility-function", potentially in conflict with other agents seeking to do the same. The application of game theory to economics will be central to this thesis, as rational agents are also the natural comparison point for any trained agents. In this section, we introduce some of the core principles and some of our notation.

For the purposes of this thesis, we discuss the action of energy-producers within an energy-market as rational actors seeking to maximize their profit; here, the modeling assumption of rationality is justified by an appeal to coherence theorems and money-pump arguments, such as [21], to the effect that a failure of rationality (to oversimplify, a failure to act as if an agent was a Bayesian expected-utility maximizer) would amount to behavior which could be exploited to take arbitrary amounts of utility from the agent. This work will not discuss the ways in which this model fails to accurately model reality, even as these are of prime importance in energy-market design, primarily because their mathematical modelling is a complicated and delicate matter which would take us too far afield.

We model a (single-stage, deterministic, simultaneous, perfect-information) game G as a set of agents $A^1...A^N$, to each of which is available a set of actions $\mathcal{A}^i$; we denote the Cartesian product of all actions $\mathcal{A} = \bigotimes_{i=1}^{N} \mathcal{A}^i$. Each agent has a utility function $U^i : \mathcal{A} \to \mathbb{R}$ - a function which, for any joint action, assigns the agent's valuation of such an outcome.

A joint action $a^* \in \mathcal{A}$ is said to be an equilibrium if no agent can increase its expected utility by a unilateral deviation:

$$\begin{aligned} &\forall i \in \{1, ..., N\}, \forall a^i \in \mathcal{A}^i, \\ &U^i(a^{-i*}, a^i) \leq U^i(a^{-i*}, a^{i*}) \end{aligned} \tag{1.1}$$

(where the "$-i$" is a compact way of representing all actions other than that of agent i).

In general, not all games have such an equilibrium if we require an agent to choose a specific action in advance (a "pure strategy") - they may instead choose a probability distribution over possible actions, known as a "mixed-strategy". In cases where agents employ a mixed strategy, the equilibrium is known as a "Nash equilibrium". It is a well-known theorem [22] that all games of the type we consider here have at least one mixed-strategy Nash equilibrium.

A mixed strategy for an agent i is a probability distribution over possible actions $\sigma^i \in \Delta\mathcal{A}^i$ (ΔS denoting the set of probability distributions over elements of a set S). The criterion for a set of mixed strategies σ^* (note that throughout this work, unindexed quantities refer to the Cartesian product of the relevant quantity over all agents, unless otherwise noted) to be a Nash equilibrium is, similar to the above,

$$\begin{aligned} &\forall i \in \{1, ..., N\}, \forall \sigma^i \in \Delta\mathcal{A}^i, \\ &\mathbb{E}[U^i(a^{-i*}, a^i)|a^{-i*} \sim \sigma^{-i*}, a^i \sim \sigma^i] \leq \mathbb{E}[U^i(a^{-i*}, a^{i*})|a^{-i*} \sim \sigma^{-i*}, a^{i*} \sim \sigma^{i*}] \end{aligned} \tag{1.2}$$

($\mathbb{E}$ denotes the expectation value, while the tilde may be read as "distributed according to") This is the natural extension of the concept of a Nash Equilibrium as above to the case of probabilistic policies.

Finally, in some games, agents have access to hidden information which may inform their actions. In this case, it is necessary to introduce the concept of a Bayes-Nash equilibrium. In this generalization, each agent i is in possession of some hidden information $\theta^i \in \Theta^i$, distributed - we will assume independently - according to a publicly-known distribution $p^i(\theta^i)$; the agents' utility functions U^i and strategies σ^i are then permitted to be functions of this hidden information as well. In this case, a Bayes-Nash equilibrium is defined as a family of strategies $\sigma^i : \Theta^i \to \Delta\mathcal{A}^i$ [22]

$$\begin{aligned} &\forall i \in \{1, ..., N\}, \forall \sigma^i : \Theta^i \to \Delta\mathcal{A}^i, \\ &\mathbb{E}[U^i(a^{-i*}, a^i|\theta^i)|a^{-i*} \sim \sigma^{-i*}(\theta^{-i*}), \theta^{-i} \sim p^{-i}, a^i \sim \sigma^i(\theta^i), \theta^i \sim p^i] \\ \leq &\mathbb{E}[U^i(a^{-i*}, a^{i*}|\theta^i)|a^{-i*} \sim \sigma^{-i*}(\theta^{-i}), \theta^{-i} \sim p^{-i}, a^{i*} \sim \sigma^{i*}(\theta^i), \theta^i \sim p^i] \end{aligned} \tag{1.3}$$

These are the criteria by which we shall evaluate our reinforcement learning agents - these equilibria are the strategies to which a MARL training process should converge.

1.4. Reinforcement Learning

Reinforcement Learning (RL) is a paradigm for machine learning; in contrast to the other major divisions of machine learning, supervised or unsupervised learning, reinforcement learning focuses not on statistical prediction based on data, but instead on the maximization of a "reward function". The mathematical framework for understanding RL is that of Markov Decision Processes (MDPs); the most basic type of MDP is defined by four main objects: $\mathcal{S}$, the set of all possible states of the environment, $\mathcal{A}$ the set of actions available, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the reward function (the quantity to be optimized) and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$, the transition function, which details the probability of arriving at a particular new state given the prior state and an action taken. To these is often appended a time-limit T (such that at most T actions may be taken consecutively) and a discount-factor $0 \leq \gamma \leq 1$, representing the rate at which rewards obtained at later times are discounted compared to those at earlier times (i.e., the reward at time t is discounted by a multiplicative factor γ^t). We refer to ordered lists of state-action pairs $(s_0, a_0, s_1, a_1, \ldots s_T, a_T)$ as "trajectories" (often denoted by the letter τ). Sometimes agents do not know the true state of the environment s_t, but instead must infer it based on observations o_t which contain incomplete information about the full state; unless otherwise noted, we will generally elide states and observations of states, as the distinction is usually clear from context.

The core of the RL problem is to find a (possibly probabilistic) "policy" $\pi : \mathcal{S} \to \Delta\mathcal{A}$ which maximizes the expected cumulative reward $R[\pi]$,

$$R[\pi] = \mathbb{E}\left[\sum_{t=0}^{t=T} \gamma^t r(S_t, A_t)|A_t \sim \pi(S_t), S_{t+1} \sim \mathcal{T}(S_{t+1}|S_t, A_t), t \geq 0\right] \tag{1.4}$$

(for simplicity, we consider the initial state $S_0 = s_0$ to be fixed). Following (the first edition of) [23], we use capital letters to indicate random variables, and lower-case letters to indicate a particular instantiation of the corresponding random variable.

As this notation is quite cumbersome, it is customary to suppress the distributions from which the random variables are drawn, instead indicating that trajectories τ are drawn according to the policy π; for a quantity X, $\mathbb{E}[X(\tau)|\tau \sim \pi]$ or $\mathbb{E}_\pi[X]$. In order to briefly distinguish it from the reward at a particular timestep, $R[\pi]$ is typically called the "return". The policy should choose actions which, in expectation, yield trajectories along which the most possible reward is gathered. When the policy is chosen from a family of policies dictated by some parameters θ, the corresponding policy is denoted π_θ, in which case the return may be considered a function $R[\theta]$.

Some algorithms, such as the eponymous REINFORCE [24], directly optimize this policy by estimating $\nabla_\theta R[\theta]$ and using this quantity to perform gradient descent. Such techniques are known as "policy learning". Alternatively, many algorithms seek to learn a value function which approximates the return, and to subsequently optimize the policy with respect to this - which algorithms are called "value-learning". There are two common value functions, the "state-value function" $V_t^\pi(s)$ and the "state-action-value function" $Q_t^\pi(s,a)$. The definitions of these are as follows:

$$\begin{aligned} V_{t_0}^\pi(s) &= \mathbb{E}\left[\sum_{t=t_0}^{t=T} \gamma^t r(S_t, A_t) | \tau \sim \pi, S_t = s\right] \\ Q_{t_0}^\pi(s,a) &= \mathbb{E}\left[\sum_{t=t_0}^{t=T} \gamma^t r(S_t, A_t) | \tau \sim \pi, S_t = s, A_t = a\right] \end{aligned} \tag{1.5}$$

Thus $Q_t^\pi(s,a)$ may be interpreted as "the expected return of taking action a while in state s at a time t, and operating according to the policy π thereafter", and analogously for V_t^π.

The functions given above are typically not found by directly computation (not least because this requires the evaluation of $|\mathcal{S}|^{|\mathcal{A}|^t}$ trajectories), but are instead estimated on the basis of their values over a small family of sampled trajectories. The selection of these trajectories requires balancing the desire to consider only a limited set of "promising" trajectories ("exploitation" according to the possibly inaccurate value function) with the need to calibrate the value function over a wide family of trajectories ("exploration"). This is commonly referred to as the "explore-exploit" tradeoff. In general, this means that, in sampling trajectories, there is cause to, some fraction of the time, take an action distinct from that which the policy "guesses" is optimal. The particular techniques used in this work will be discussed in the corresponding sections. [23] discusses a wide variety of exploration techniques, with more created regularly.

1.4.1. RL Algorithms

Now that we have covered the framework and notation, a brief discussion of the relevant algorithms is appropriate. More information can be found in [23]. The goal of reinforcement learning is to find policies π which are optimal in that the maximize the expected return $R[\pi]$. While some approaches, such as "dynamic programming", do essentially consider all possible policies and simply select the optimal one, these are of limited utility, as the number of possible policies scales double-exponentially $|\mathcal{S}|^{|\mathcal{A}|^t}$. Instead, the traditional focus of reinforcement learning has been to find methods which achieve as much return as possible, even if these policies do not resemble the optimal policy. For RL to be practically useful, the problem must have a structure which is learnable, often due to simplicity or modularity of a given time-step - schematically, things like mechanical-systems control, which have frequent and informative feedback, are much more amenable to RL than something like guessing the password to a safe (in which RL cannot perform better than brute force).

Here we will discuss only the algorithms used in this work, in hopes of illustrating the reason for their selection.

Tabular Q-Learning

We begin with "tabular Q-learning", among the simplest value-learning algorithms. Tabular Q-learning can be used when the state and action spaces are discrete and finite; that is, one can readily write down a (possibly very large) table indexed along the columns by states, and along the rows by actions. Tabular Q-learning simply records the expected Q-value for each of these table-entries based on its experiences (here we do not consider the importance of the time-variable for clarity's sake).

As an example, let us consider a market with two "states" - cases of "High" and "Low" demand; a single monopolist agent is allowed to bid either 10 or 20 Mega Watts of capacity, at either 10 or 50 USD/MWh (a total of four actions). Perhaps at some point in time, the firm has the following Q-table:

	s="Low"	s="High"
a="10MW at 10USD / MWh"	30	30
a="20MW at 10USD / MWh"	40	70
a="10MW at 50USD / MWh"	5	100
a="20MW at 50USD / MWh"	10	**250**

Table 1.1: Example Q-table prior to update - quantity to be updated bolded

Note that these entries need *not* be correct at any particular moment in time - they are learned gradually. To illustrate the update-rul, supposing the agent sees a state of "High" demand, and in turn determines to bid 20MW at 50USD/MWh, it sees a profit of, say 400. This would cause an update of the form

	s="Low"	s="High"
a="10MW at 10USD / MWh"	30	30
a="20MW at 10USD / MWh"	40	70
a="10MW at 50USD / MWh"	5	100
a="20MW at 50USD / MWh"	10	**350**

Table 1.2: Example Q-table after soft update - updated quantity bolded

Note now that the entry is not overwritten by the new observation, but instead is shifted in that direction - in general, rewards might not always be the same for every state-action pair, while we want to keep track only of the average reward for a given pair.

After an agent has collected data through some form of exploration, the Q-table determines their policy: for any state, play the action which, for that state, yields the maximum reward.

Continuous Case

Why then is it necessary to use something like neural networks or deep learning, if tabular Q-learning, for sufficiently large tables, can handle the problem adequately? First, this tabular approach doesn't take advantage of any patterns which exist in the environment: a tabular approach would treat states like "Extremely Low Demand" and "Very Low Demand" as distinctly as it would either of those and "Extremely High Demand" - each simply has their own set of table entries. Second, as the number of table-entries increases, so too must the number of samples, in order to converge to the correct value in each case. Third, if the environment is truly continuous (e.g., if the values "demand" can take may lie anywhere within some interval), discrete approximations may introduce much error, and improve only at great computational expense. Thus, the switch from tables like Q_{sa} to functions $Q(s, a)$. Typically these functions are parameterized by a family of parameters θ (in our case, these will be neural network weights.

Finally, typical applications of RL to economic problems and market design are focused on determining the incentives of actors within the market - it isn't generally relevant that the RL agent does relatively well, but instead the desideratum is the extent to which it learns game-theoretic equilibrium behavior when other methods of their calculation fail. Because this is most likely to happen in more complex environments, possibly consisting of multiple interconnected markets or multiple interrelated timesteps, tabular methods become inefficient and impractical; thus, neural networks are the go-to solution for discovering complex behavior - not necessarily only in the continuous case. We leave off discussion of these neural-network methods for Section 3.2, that we may discuss them in the context of the non-stationarity problem.

1.4.2. The Connection Between Economics and RL

What relevance has Reinforcement Learning to economics as a whole, let alone to the design of electricity markets? As noted previously, economics is, at least in part, the theory of so-called "rational actors", whose rationality is defined, with reference to, e.g., the von-Neumann-Morgenstern axioms [21], as the maximization of some expected-utility function. While this model is generally regarded only as an approximation to the behavior of individuals with potentially conflicting preferences or other irrationalities, the relation to firm-behavior is, if imperfect, much more direct - the nominal purpose of

a for-profit firm is the maximization of profit. Thus the firm may be modeled as an agent attempting to maximize expected profits in the face of uncertainty.

Regarding market-design more specifically, there are two principle reasons to speak of RL methods. The first is practical: if market-design is to be regarded as some sort of game between agents, this game must at some point be solved; while traditional optimization solvers (e.g., EPEC models [9]) provide exact answers for a limited class of models, as the complexity of the game increases, possibly violating the assumptions of solvers based on, say, convexity, such direct solution becomes intractable. On the other hand RL algorithms are readily adapted to any environment in which there is a well-defined reward function. While RL will not, in general, determine the optimal solution in finite time, it can often readily arrive at reasonable strategies (and thus, hopefully, approximate the actual behavior of firms with sufficient fidelity to draw conclusions about the suitability of a particular market rule).

The second reason is theoretical: firms in reality exhibit behavior which would be irrational were their objective solely the maximization of expected profit; instead, considerations such as risk-aversion [25], and managerial short-termism can be considered rational only with respect to utility functions more complex than the strict expected-profit; RL methods extend to such models merely by changing the reward-function appropriately. Further, in games with multiple equilibrium solutions, actors must eventually co-ordinate to arrive at one particular solution, a process known in economic theory as ”tâtonnement” [26] - modeling agents as reinforcement-learners provides an account of how, subject to appropriate assumptions, rational agents should learn.

The extent to which learning provides a realistic model of behavior in out-of-equilibrium games is discussed in [18] (though with respect to individuals rather than firms). The celebrated Sonnenschein-Mantel-Debreu theorem [26] demonstrates that markets composed of rational actors need not, in general, converge to a unique and stable equilibrium via the tâtonnement; the non-stationarity problem is in many regards analogous to the failure of tâtonnement, except for occurring between learning agents taking multiple actions over time. In this sense, our work with respect to non-stationarity will be an attempt to eliminate all these tâtonnement-failure-analogues which are due solely to the learning process and not the underlying game structure.

2

Theoretical Developments

This chapter discusses the theoretical background for understanding the non-stationarity problem. Section 2.1 introduces non-stationarity in the context of independent-learning. Section 2.2 introduces readers to the TD-error and develops the optimality-deficit, and discusses how these two measurements will be used to evaluate simulations of firm behavior. Section 2.3 proves conditions under which centralized-algorithms like MADDPG will converge, in the process illustrating why decentralized-algorithms cannot, in general, converge to correct behavior. Finally section 2.4 introduces the idea of "adversarial market-design", contextualizing its niche within the literature, and relating it to the other parts of the present work.

2.1. Non-Stationarity and the Failure of Independent Learning

2.1.1. Independent Learning

Traditional RL assumes that the environment within which an agent operates is Markovian - that is, that state-transition and reward probabilities, $\mathbb{P}(S_{t+1}, R_t|S_t, A_t)$ are independent of time [27]. Independent learning (IL) refers to any MARL scheme in which agents are trained using single-agent RL techniques, taking the behavior of other agents to be implicit in the environment. That is, though the state-transitions or rewards may depend on all agents' actions, an IL agent will learn as though its action alone determined these probabilities. In the case where only one agent is learning (others remaining fixed), the Markovian assumption holds, as the other agents' actions may be understood as part of a stochastic environment. On the other hand, when multiple agents are learning simultaneously, the Markovian assumption is, in general, violated. This in turn means that standard guarantees of convergence no longer apply.

Recent work in this vein has focused on applications to cooperative RL, in which agents learn separately but share an overall reward function. In this case, techniques such as decomposition networks [28], coordination graphs [29], and optimality-gap formulations [30] have been used.

By contrast, when agents do not share a reward-function (i.e., in mixed-sum environments), the problem ceases to be one of credit-assignment; in such cases, algorithms such as fictitious play [31], along with techniques like leagues [32] have been used. The algorithm we focus on here, MADDPG, has been used successfully in general MARL environments [16] and in electricity market behavior in particular [20]. At the end of Chapter 2, we discuss general criteria for value-learning convergence.

The problem of non-stationarity in MARL ultimately entails a failure of agents to converge to a Nash-equilibrium, in which case they cannot be used to meaningfully infer the properties of electricity markets with respect to strategic behavior.

2.1.2. Examples

Following [33], consider the co-operative simple matrix game described by the utilities:

$$U = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 6 \\ 0 & 6 & 7 \end{bmatrix} \tag{2.1}$$

(element U_{ij} corresponds to agent 1 taking action i while agent 2 plays action j). Now, if both agents begin by exploring randomly and independently, each outcome will occur with probability $1/9$. E.g., agent 1 learns that action 3 has value $Q^1(3) = (0 + 6 + 7)/9 = 13/9$. Thus, each agent's learned (after training to convergence) Q-values will be $10/9$, $11/9$ and $13/9$ for actions 1, 2, and 3, respectively. Thus supposing the agents implement a greedy policy after training, they will both choose action 3, receiving 7 reward rather than the optimal 10.

This example illustrates the phenomenon of relative overgeneralization for co-operative games. Figure 2.1, reproduced from [33], illustrates the same problem graphically for a different co-operative game. in such cases where there is a shared reward function, there exist a number of techniques which facilitate co-operative learning, as discussed above. These techniques, however, do not readily apply in the case of mixed-sum games such as the economic dispatch problem.

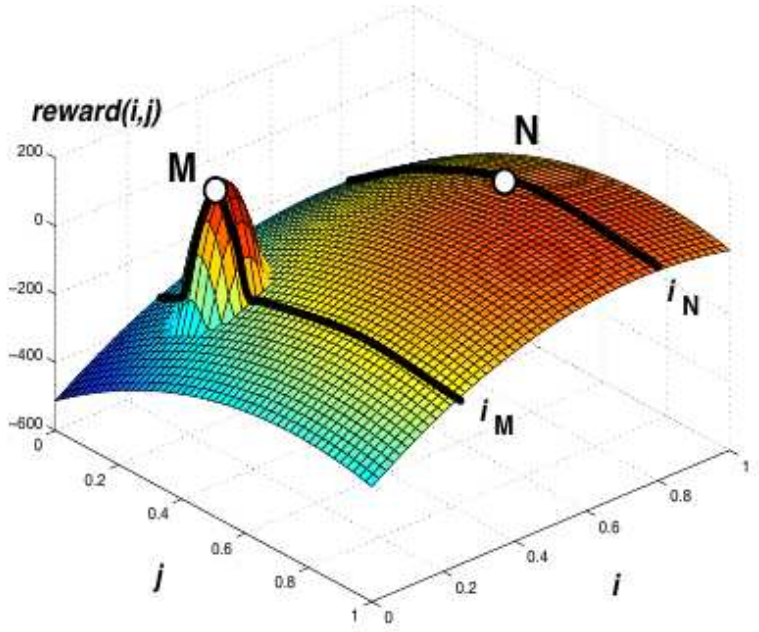


Figure 2.1: Payoff-function for a cooperative game which illustrates the relative overgeneralization pathology. The capital letters "M" and "N" represent joint actions, while the i_M and i_N illustrate the surface along which an agent estimates the average reward for its taking the corresponding action. While M is the optimal action, the agents estimate that i_N is better than i_M, and thus play suboptimally. Reproduced from Figure 1 of [33].

The reason this phenomenon is relevant is that many RL algorithms such as tabular Q-learning and DDPG mistakenly conflate the average reward of an action with the reward of a particular joint action.

2.2. Convergence Criteria and Definition of the Optimality-Deficit

2.2.1. TD-Error and the Optimality-Deficit

We wish to study when RL algorithms converge to an equilibrium solution to the economic dispatch problem; thus we must define some notion which measures the distance of an agent's policy from equilibrium. This section defines such a notion which will be used throughout our experiments, which we call the "optimality-deficit", along with the well-known TD-error; each fulfills a distinct function.

For simplicity, we will begin by discussing the illustrative, simplified case in which the environment itself has no state, and all actions are selected from a discrete set of actions (the same for all agents). We will restrict ourselves also to the case where there are only two bidding agents.

Let us call the policies played by each agent π^1 and π^2, respectively (here, interpreted as vectors describing probability distributions over the discrete action-space). Then the expected return of agent i is given by the simple matrix-vector multiplication

$$\pi^{1,T} U^i \pi^2 \tag{2.2}$$

With U^i the utility matrix of player i (whose elements, U^i_{mn} are the profit obtained by agent i when agent i performs action m and their counterpart action n.

The Nash equilibrium condition asserts that each player is playing a best-response to the other. For agent 1, this takes the form

$$\pi^1 \in \text{argmax}_{\pi^1} \pi^{1,T}(U^1\pi^2) \tag{2.3}$$

(The brackets emphasize that fixing the counterparty's policy entails an effective single-player utility). The Nash-equilibrium condition corresponds to requiring that this condition attains simultaneously for all bidders.

While it is not necessarily easy to calculate Nash equilibria, it is simple enough to verify whether a given set of policies constitutes an equilibrium (at least in this discrete case). A best-response is one in which the reward is the equal to the reward obtained by playing the deterministic policy which always plays the maximum value of (the vector) $U^i\pi^j$ (this strategy may not itself be the equilibrium, as it would not necessarily satisfy the criterion for the remaining agent).

Thus, we can quantify how far a given policy is from a Nash-equilibrium based on it's loss of potential profits:

$$\lambda^i = ||U^i\pi^j||_\infty - \pi^{i,T}U^i\pi^j \tag{2.4}$$

Here $||\cdot||_\infty$ denotes the infinity-norm of a vector (equal to the value of the vector's maximum entry). Intuitively, this deficit is the profit which agent i fails to obtain by fact of playing suboptimally, assuming the behavior of the opponents are fixed. This λ^i we call agent i's "optimality-deficit". When the optimality-deficit of every agent is zero, the outcome played is a Nash-equilibrium (as every agent is playing a best-response given their opponents' strategies). The appropriate generalization to mixed-strategies simply entails taking expectation-values appropriately.

Having developed a criterion by which to gauge whether a true equilibrium has been reached, we now wish to define a measure that describes when a training procedure has converged (not necessarily to a correct equilibrium). A commonly used measure, the TD-error, felicitously fulfills this function. The TD-error is equal to the difference between the agent's expected reward and its actual reward. When these two are approximately equal for every state-action pair, the value-function has converged; in the following section, we consider stateless environments, where the policy is the argmax of the value function. Under our simplifying assumptions the formula for the TD-error δ_t at time t is:

$$\delta_t = Q(a_t) - r_t \tag{2.5}$$

where a_t is the action taken at time t, r_t the reward received, and $Q(a_t)$ the agent's predicted reward. Note that this definition applies only to a single, independent learner - it does not admit introducing the effects of other agents' actions. Intuitively, when the TD-error has gone to zero for all agents over all actions (or at least over all actions which are played), the agent is no longer 'surprised', and will continue to act as it thinks best according to its Q-function. Note also that a zero TD-error will only necessarily correspond to an unchanging final policy in some circumstances; commonly an argmax policy is used in value-learning $\pi(s) = \text{argmax}_a Q(s,a)$ - in which case a zero TD-error does indeed entail a static policy.

2.2.2. Formulation of the Optimality-Deficit for General MDPs

For reference, we formulate the above-defined optimality-deficit without the simplifying assumptions of statelessness, observability, discrete action space, or having only two agents. Here we let τ^i_t denote the observation-action history of agent i at time t, as opposed to a full trajectory.

The Bellman-equation defining the optimal state-value function is

$$V^{*,i}(\tau^i_t) = \max_a \mathbb{E}\left[R^i(S_t, A^{-i}_t, a) + \gamma V^{*,i}(\tau^i_{t+1})\right] \tag{2.6}$$

As noted before, all capital letters are understood to be random variables over which expectations are taken (albeit with τ being inferred from context in the absence of an easily identifiable capital). As much as possible, we will suppress distributions which are obvious from context. Likewise, the observations of other agents are suppressed for notational simplicity, but must also be integrated over.

The expected profit of an agent obeying strategy π^i is similarly

$$V^{\pi^i,i}(\tau^i_t) = \mathbb{E}\left[R^i(S_t, A^{-i}_t, A^i_t) + \gamma V^{\pi^i,i}(\tau^i_{t+1})|\pi^i\right] \tag{2.7}$$

Different generalizations of the optimality-deficit can be obtained by choosing whether the agent subsequently obeys some original policy or its optimal policy. As we do not, in general, have access to the optimal policy or value function, the version for which we will have the most use is the deficit with respect to an agent subsequently playing its original policy; this corresponds to the expected profit suboptimality in a single timestep.

$$\lambda^i(\tau_t^i) = Q^{\pi^i,i}(\tau_t^i, a^{i,*}) - V^{\pi^i,i}(\tau_t^i) \tag{2.8}$$

In the upcoming section, we will have occasion to refer to the case in which agents bid for a single clearing of a market, and in which each has hidden information about its generation capacity and marginal costs; for this case, the idea is the same but for taking the expectation over the counterparties' policies (once again specializing to a single-timestep environment):

$$\lambda^i(o_t^i) = \max_a \left\{\mathbb{E}\left[R^i(S_t, A^{-i}, a)|A^{-i} \sim \pi^{-i}(O^{-i})\right]\right\} - \mathbb{E}\left[R^i(S_t, A^{-i}, A^i)|A^{-i} \sim \pi^{-i}(O^{-i}), A^i \sim \pi^i(o^i)\right] \tag{2.9}$$

Intuitively, the deficit represents the expected loss in profit - compared to what it would achieve playing optimally - of an agent with generation parameters τ_t^i, supposing that its opponents bid according to fixed policies, with their own hidden generation parameters drawn from some known distribution.

And of course, if one considers the case not for one timestep, but for the overall optimality-deficit of a policy, one finds a natural multi-agent generalization of the concept of "regret" from traditional learning theory.

One important feature of the discrete environment is the ability to directly compute the optimality-deficit: fixing the agent whose optimality-deficit we wish to calculate, the remaining agents may be assumed (once exploration has been disabled) to play a greedy policy which always selects their guessed-optimal action - thus, the given agent's optimal action may be obtained by brute-force search, requiring only $|\mathcal{A}^i|$ (the number of actions available to the agent under consideration) market clearings.

2.2.3. The Optimality-Deficit in the Continuous Case

In comparison to the tabular case, calculating the optimality-deficit in continuous spaces can be quite difficult. In particular, one must calculate, for each possible hidden-state, the expected return of an agent's optimal action. This entails, for each candidate for the optimal action, sampling over hidden-states and opponents' possibly stochastic policies - a scaling that quickly becomes intractable. In this section, we describe one approach to approximately compute the optimality-deficit for DDPG using an MADDPG-agent trained in parallel.

Our goal is to know, for a given agent, and fixing the other agents' policies, the expected profit of acting optimally. Supposing that the given agent is trained using DDPG, and thus is associated with both a policy $\mu^i(o^i)$ and a critic function, $Q^i(o^i, a^i)$. We then introduce a "supervisor" counterpart of this agent (trained on the same rounds as the acting agents with appropriately augmented data, and used only for deficit estimation - never acting) with associated $\mu_S^i(o^i)$ and $Q_S^i(o^i, a^i, o^{-i}, a^{-i})$ (the "S" subscript denoting the supervisor). Further, as the supervisor policy is meant to determine the best-response, it may be made more accurate by giving it access to other agents' actions and observations, i.e. $\mu_S^i(o^i, o^{-i}, a^{-i})$ (this generalization may or may not be the most sensible depending on context - one may wish to measure deficit with respect to optimal-outcome behavior or with respect to optimal-in-expectation behavior).

In the limit of "perfectly effective" training, we anticipate that the supervisor should yield the action which maximizes the expected centralized Q-value; that is, $\mu_S^i(o^i) \rightarrow \max_{a^i}\mathbb{E}\left[Q_S^i(o^i, a^i, O^{-i}, A^{-i})\right]$. This is precisely the 'optimal action' we would like to use in formulating the deficit (indeed, there is no particular need to use reinforcement-learning to find this action - other stochastic optimization methods may be used to determine the optimal action, if desired).

Then, the approximate value of the deficit is,

$$\lambda^i(o^i, \mu_S^i(o^i), o^{-i}, a^{-i}) \approx \max_{a^i} Q_S^i(o^i, a^i o^{-i}, a^{-i}) - Q^i(o^i, \mu^i(o^i), o^{-i}, a^{-i}) \tag{2.10}$$

Unfortunately, the primary algorithms we discuss here, DDPG and MADDPG are on-policy algorithms - that is, training the supervisor policy based on the actor's actions will not work. where possible, instead made to analytical solutions. Future work might examine alternative extensions of the optimality-deficit and other ways to compute it.

2.3. Convergence Proofs

We are now in a position to discuss mathematically the manifestation of the non-stationarity problem as a convergence to incorrect equilibria.

In what follows, we seek to prove sufficient conditions for convergence to a correct equilibrium; to make this simpler, we consider the discrete case with only two agents; the generalization to many agents is straightforward, while convergence theorems are impractical to prove for the continuous case in which Q-function updates rely heavily on the complicated neural network parameterization and specifics of the optimizer used for gradient-descent.

First we consider the centralized policy: Let Q_t^i be a matrix whose entry i, j represents the agent's Q-value estimate of the reward obtained for itself and its opponent taking actions a^i and a^j, respectively. Upon the action-pair (a^i, a^j) occurring, an update rule dictates that agent k updates as

$$Q_{t+1}^k = f(Q_t^k, R_{ij}^k \delta_{ij}) \tag{2.11}$$

(where δ_{ij} is the matrix which is 1 in entry i, j and 0 in all others) For instance, the soft-update rule has the form $f(A, B) = A(1 - \delta_{ij}) + \delta_{ij}(\rho A + (1 - \rho))B$. This is simply to say that the update rule updates only the entry for the action-pair which is observed, which updates to a weighted average of the estimated and observed value.

The following condition suffices to show that an update rule for Q is non-increasing: if we have for some matrix-norm $|| \cdot ||$ that for any observation (a^i, a^j)

$$||Q_{t+1}^k - R^k|| \leq ||Q_t^k - R^k|| \tag{2.12}$$

then the update-rule is non-increasing and bounded above by $||Q_0 - R||$, and bounded below by 0 - thus the sequence converges (not necessarily to 0).

Then, requiring that, for each possible observation-pair, (a^i, a^j), $Q_{t+1,ij}^k = R_{ij}^k$ or that the observation-pair occur with non-zero probability, and upon occurring, cause $|Q_{t+1,ij}^k - R_{ij}^k| \leq \rho |Q_{t,ij}^k - R_{ij}^k|$ for some $0 \leq \rho < 1$. That this criterion is sufficient for convergence follows from considering any observation-pair (a^i, a^j) where $|Q_{t,ij}^k - R_{ij}^k| = c_t > 0$ for some value c_t. Let the stopping-time $\tau_1 = t$ denote the event that t is the first time such that $c_t < c_{t-1}$. By hypothesis, τ_1 is finite with probability one. Then let τ_2 be the stopping-time associated with the second such decrease - conditioning on $t \geq \tau_1$ (a set which is nonempty as τ_1 is finite), τ_2 is likewise finite, etc. Thus, for any integer N, there exists, with probability one, a time t_N such that (a^i, a^j) has been drawn N times; in each case, the mismatch $|Q_{t,ij}^k - R_{ij}^k|$ decreases by at least a factor of $\rho < 1$; as this is true for each pair of indices and for all N, Brouwer's fixed-point theorem [34] implies that the Q-estimate converges element-wise and almost-surely to the true reward. Thus we have:

Theorem: Let Q_t^k denote agent k's estimated Q-function as a matrix whose entries correspond to possible action pairs of all agents. Similarly, let R^k be a matrix denoting the actual reward obtained, and $f(Q^k, R_{ij}^k \delta_{ij})$ be an update rule which alters the Q-estimate upon observing a reward R_{ij}^k corresponding to the action pair i, j. Then, if there exists a matrix-norm $||\cdot||$ such that always $||Q_{t+1}^k - R^k|| \leq ||Q_t^k - R^k||$, then the error $||Q_t^k - R^k||$ converges to a constant c as $t \to \infty$.

Theorem Suppose the following three conditions on are met:

- Seeing the i, j-action pair, the update rule alters only Q's i, jth entry
- For each i, j there exists a ρ, $0 \leq \rho < 1$, $|Q_{t+1,ij}^k - R_{ij}^k| \leq \rho |Q_{t,ij}^k - R_{ij}^k|$
- All action-pairs (a^i, a^j) for which $Q_{ij}^k \neq R_{ij}^k$ occur with non-zero probability

Then Q_t^k converges element-wise and almost-surely to R^k as $t \to \infty$.

A stateful multi-agent MDP with multiple timesteps can always be transformed into a multi-agent, stateless, timeless MDP such as the one discussed above; each agents' transformed action is a policy over all partial-trajectories. Thus,

Corollary The convergence theorem stated above applies to stateful MDPs with multiple timesteps if a nonstandard Q-function over trajectories is used.

Let us now turn to the independent case, analogous to DDPG. In this case, the Q-estimate can be written as a vector q_t^k (or as a matrix whose columns are all q_t^k - one for each of the opponents' actions). The analogous update rule is, for an observation of the i, j action-pair,

$$\begin{aligned} q_{t+1}^k &= f(q_t^k, R_{ij}^k \delta^i) \\ [q_{t+1}^k, q_{t+1}^k, \ldots] &= [f(q_t^k, R_{ij}^k \delta^i), f(q_t^k, R_{ij}^k \delta^i), \ldots] \end{aligned} \tag{2.13}$$

(the second-line is the matrix-representation, for more direct comparison)

It is immediately clear that the theorems above will not apply in general. The contraction property cannot be assumed without further assumptions, and further, such an update rule necessarily entails altering entries of the Q matrix which have nothing to do with the observation. Intuitively, provided that the second agent's actions effect the first's reward, it can cause the first agent's Q-function to oscillate merely by consistently playing one action for a long time, and then playing a different action for a long time, and so on.

Whether or not this mis-generalization occurs in practice thus clearly depends on the update rule and the behavior of the other agent.

2.4. Adversarial Market-Design

In general, market-design is the discipline of translating objectives into market-design parameters. If these objectives can be translated into some generalization of the "social-welfare" function from economics, and these parameters can be quantified, it is possible to phrase the problem of market design itself in terms of reinforcement learning.

As an example, consider the case of an oligopolistic market with known supply and demand curves; a market-designer might take their goal to be the maximization of the social-welfare function, and aim to set bidding rules accordingly. Note that, many market-design questions entail deciding between two discrete alternatives (e.g., a nodal vs. a zonal pricing-scheme) in such cases, it is possible to employ tabular methods, though this is likely not an effective method for performing such analyses.

In the language of (single-parameter) mechanism-design, a market rule consists of an outcome-rule $x(b)$ mapping a set of bids b to a set of good allocations, and a payment-rule $p(b)$ mapping a set of bids to payments exacted upon each bidder. One must further specify the bidders' utility functions, $u(b)$ [35]. Each bidder values a good differently, modeled as corresponding to a vector of random values V (realized values are known only to their bidder). Now, for a given market, it is probably the case that bidders' utility functions are not subject to market-design; on the other hand, the allocation and pricing rule likely are. The revelation principle [36] implies that for any mechanism, there exists a mechanism such that it is a weakly dominant strategy for all agents to submit their bids as truthful reports of their own values; such mechanisms are known as "dominant strategy incentive compatible" (DSIC).

In the case of an electricity market, then, bidders are power-producing firms while market-design parameters might include things such as a price-cap and/or floor, or more complicated tiered price-restrictions which set a dynamic price-cap according to demand. Meanwhile, the bids correspond to agents' submitted (Q, P) pairs. A market with fixed demand-function $P_D(Q)$, according to the revelation principle, has some clearing-rule such that truthfully reporting $(Q_{\max}, MC)$ is a dominant strategy for the producing firms - this cannot be the standard merit-order clearing-rule, as this rule in general allows agents to profit by e.g., strategically withholding capacity.

Here we introduce the concept of adversarial market-design. In this method, an "adversary" agent is trained to select amongst the possible market-designs, while firm agents are trained to act strategically within the selected design. Unlike the standard procedure of mechanism design, this method does not focus on producing DSIC mechanisms, but instead on choosing the optimal mechanism when presented with a parameterized family of possible market-designs. In adversarial market-design, we introduce an agent whose purpose is to select the optimal market-design - i.e., that which maximizes some social-welfare function. No longer assuming honest bids, firms are simultaneously trained to behave strategically under this market-design. Such an approach is geared towards cases where, e.g., a regulator has decided to implement a particular rule, say, a price-cap, and wishes to know what value it should be set at.

Traditional mechanism design approaches cannot solve this problem, as the mechanism guaranteed by the revelation-principle will, in general, not be a member of the family of parameterized market-

designs. For example, one might determine the optimal price-cap given honest bids, only to find that the outer mechanism necessitated by the revelation-principle cannot be implemented using a price-cap alone.

We illustrate the difference in procedure in Figures 2.2 and 2.3. Figure 2.2 illustrates a procedure which a market-designer might use to evaluate different market-designs via simulation - the designer begins by selecting some number N of designs to compare; these might be, say, markets with one-price and two-price imbalance settlement rules. Firm behavior is simulated in each case, and the results compared by hand; of course, all steps are moderated by theoretical analysis and the designer's good judgement, ultimately leading to a discussion of results, and if appropriate, a recommendation as to which design is ”better”.

By contrast, adversarial market-design approaches the problem differently; instead of starting with a discrete number of conceptual designs, the market-designer lays out a parameterized family of market-designs - then the adversarial agent chooses amongst this family while firms simultaneously learn to adjust their strategies in the new environment.

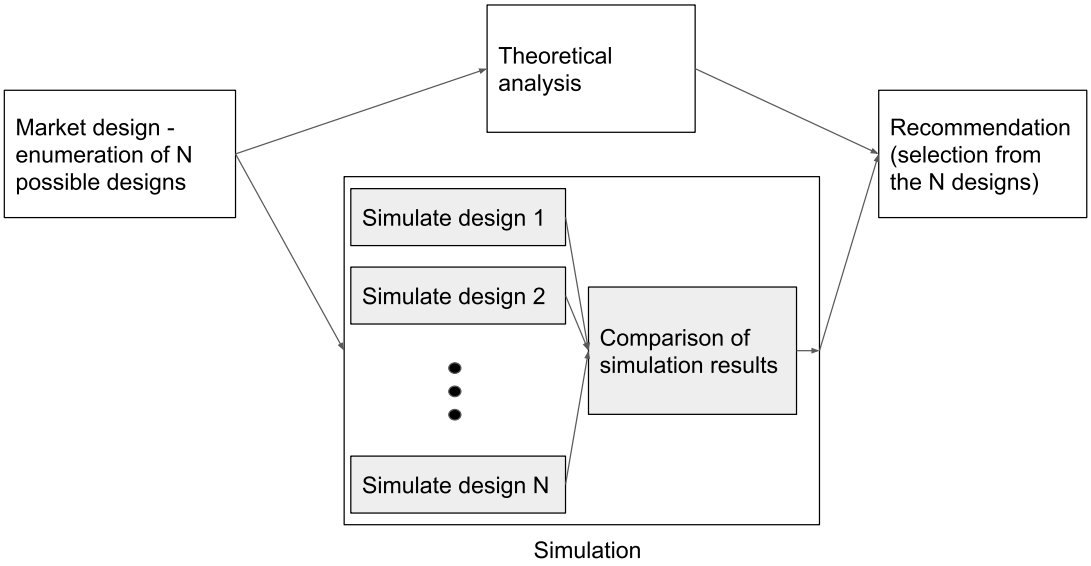


Figure 2.2: Schematic illustration of the traditional use of simulation in market-design; starting with N discrete candidate designs, a market designer simulates each in order to gather information about strategic behavior, and compares the results by hand. Using theoretical analysis to inform their judgement at each stage, the designer evaluates each of the N designs according to some criteria, and, often, makes a recommendation as to which is ”best”.

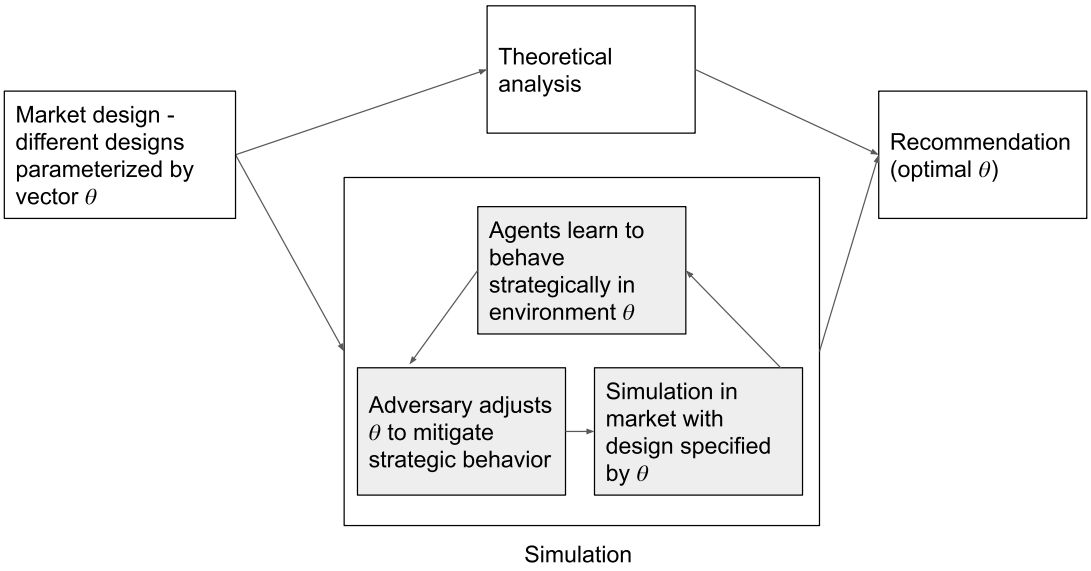


Figure 2.3: Schematic illustration of the role of adversarial market-design in evaluating markets. In contrast to the traditional case, the markets under consideration must be a parameterized family. Instead of running many simulations to evaluate the incidence and impact of strategic behavior, the adversary agent is tasked with mitigating strategic behavior from within the simulation-loop. The output is a selection of one particular member of the parameterized family, which hopefully optimizes the adversary's selected social-welfare function.

3

Simulation Setup and Methodology

In this chapter, we outline the three classes of experiment we shall run. Section 3.1 discusses the setup of the market and the training procedure pertaining to the tabular experiments. Section 3.2 introduces the DDPG and MADDPG algorithms at length. Section 3.3 discusses the relevance of analyzing an environment with a continuous bid-space, and introduces the two cases we will examine within this environment, additionally providing a description of the training procedure and hyperparamters. Finally, section 3.4 explicates the the form of the experiments illustrating the use of adversarial market-design.

3.1. Tabular Day-Ahead Market

In light of the theory presented previously, we wish to investigate the circumstances under which independent learners will converge to an incorrect bidding strategy. To this end, we simulate a simple day-ahead market, described in detail below, with agents trained implementing simple tabular Q-learning. The optimality-deficit is calculated, and examples are shown in which convergence occurs while agents maintain a non-zero optimality deficit.

3.1.1. Description of the Environment

For the sake of clarity, we model a simple day-ahead electricity market. Consumer demand is taken to be linear as a function of the electricity (in MWh) demanded. A single market-clearing is simulated - technical operation constraints are not considered. Agent observations are given by four parameters, each selected from a discrete list of possible candidate values: two parameters describe the demand function (i.e. a slope and an intercept), which are visible to all agents. The remaining two parameters describe an agent's maximum generation capacity (in MWh), and the marginal operating cost of their generator (in USD / MWh) - these generation parameters are visible only to the generator's owner (e.g. agent 1 is not permitted to see agent 2's marginal operating cost). Of course, this is not especially realistic given that firms can reasonably infer such information about their opponents - the point is to provide the grounds to illustrate the optimality-deficit in a game with hidden information.

In each round, agents select a pair of actions (a, b) corresponding to the fraction of their total generation capacity to bid, and the fraction of their marginal cost to bid - i.e., the action selection (a, b) corresponds to a bid to sell aQ^i_{max} units of electricity at a price bP^i_{MC}. Allowed values of a and b are the same for all agents at all times, and are each drawn from a distinct discrete set. This set is a linear spacing between a minimum and maximum value with a chosen number of sample points. Unless otherwise noted, Q_{max} fractions were permitted to go from 0.5 to 1.0, while P_{MC} fractions were permitted to range from 0.5 to 10; there 3 choices for quantity bidding, and 10 for price-bidding, resulting in 30 total pairs.

The market clears in as-bid merit-order, with a uniform clearing price set to be the minimum price at which sufficient bids are accepted to satisfy demand at that same price level. In case of shortage, the clearing price is the willingness-to-pay of demand for the total offered quantity. Ties are resolved in no particular order, and the acceptance of a fraction of a bid's quantity is permitted.

3.1.2. Training Procedure

Agents were initialized with an empty Q-table. After every round, an agent learns from an observation. This learning takes one of two forms; one in which all reward-observations are stored, and their average value used as the estimated Q, and one in which Q was updated such that $Q_{\text{new}} = \rho Q_{\text{old}} + (1-\rho)r$ (with ρ a "learning-rate" parameter). We call the former the "average reward" scheme, and the latter the "soft-update" scheme.

Different types of exploration were examined - ϵ-exploration, in which agents select the action with the highest Q value with probability $1-\epsilon$, and act randomly with probability ϵ. Thus, the exploration method discussed in the example in the previous section is ϵ-exploration, with $\epsilon = 1$. Alternatively, Boltzmann exploration was also used in some experiments; in this method an action is selected as the softmax probability distribution over all Q-values [37]. Boltzmann exploration employs a parameter β (the "inverse temperature") governing the amount of exploration - $\beta \to \infty$ corresponds to choosing the guessed-optimal action with probability 1, while $\beta = 0$ corresponds to acting uniformly randomly.

For the sake of preventing a profusion of graphs and test-cases, in the experiments below, we employ the soft-update rule with $\rho = 0.99$, and epsilon exploration with $\epsilon = 1$ In the plots shown, exploration was, unless otherwise noted, employed for half of all time-steps. Unless otherwise noted, agents continued to learn (i.e. to update their Q-tables) after exploration was disabled.

3.2. Overview of DDPG and MADDPG

The purpose of this section is to briefly introduce the DDPG and MADDPG algorithms as approaches to Q-learning in continuous spaces - these will be used for the experiments described subsequently.

3.2.1. DDPG

The DDPG algorithm is an algorithm for doing deep reinforcement learning that permits one to make use of a deterministic policy - that is, an actor's policy is associated with a function (traditionally denoted μ rather than π) $\mu : \mathcal{S} \to \mathcal{A}$, as compared to probabilistic policies which have the type signature $\pi : \mathcal{S} \to \Delta\mathcal{A}$. We focus our attention on the DDPG algorithm for two principle reasons: first, the fact that the policy is deterministic makes theoretical computations much simpler, as no distribution-sampling is required; second, it forms the basis of the popular centralized-training decentralized-execution algorithm, MADDPG, and thus provides an especially appropriate basis for comparison. It is worth noting that deterministic policies are "universal" in the sense that any parameterized probabilistic policy may be modeled as a deterministic policy over the distributions' parameters (e.g., a policy which deterministically chooses the mean and variance of a normal distribution); on the other hand, the standard version of DDPG can only be deployed on problems with a discrete state-space and continuous action-space.

We will describe the DDPG algorithm for the case of a single-agent, as, without the modifications introduced in MADDPG, the simplest extension to the multi-agent case is merely to instantiate an independent DDPG learner for each agent.

The agent is represented by a policy $\mu_\theta : \mathcal{S} \to \mathcal{A}$, a function parameterized by parameters represented by θ. Unlike probabilistic policies, gradient-descent on deterministic policies requires modelling a reward function (sometimes this reward model is called the "critic", in which case the policy is known as the "actor", and the algorithm is said to be one amongst a class of "actor-critic" algorithms). The simplest target to model is the Q-function discussed previously, which we associate with parameters ϕ. Q is a function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. In this algorithm, gradient-descent on Q seeks to minimize the difference between the predicted Q for a given state-action pair and the observed reward; the policy μ is optimized to maximize this model-reward function. That is, ideally we would have:

$$\begin{aligned} Q^\mu_\phi(s_t, a_t) &\to \mathbb{E}\left[R(s_t, a_t) + \Sigma_{t' \geq t+1} \gamma^{t'} R(S_{t'}, \mu(S_{t'}))\right] \\ \mu_\theta(s_t) &\to \text{argmax}_\mu \mathbb{E}\left[R(s_t, \mu(s_t)) + \Sigma_{t' \geq t+1} \gamma^{t'} R(S_{t'}, \mu(S_{t'}))\right] \end{aligned} \tag{3.1}$$

Accordingly, we can consider the Bellman equations, and take the approximations:

$$\begin{aligned} Q^\mu_\phi(s_t, a_t) &\approx \mathbb{E}\left[R(s_t, a_t) + \gamma Q^\mu_\phi(S_{t+1}, \mu(S_{t+1}))\right] \\ \mu(s_t) &\approx \text{argmax}_a Q^\mu_\phi(s_t, a) \end{aligned} \tag{3.2}$$

(these are approximations because changes to one function are only gradually propagated to the other) Which in turn allows us to define the appropriate loss functions:

$$\begin{aligned}\mathcal{L}^{Q}\left[\phi\right] &= \mathbb{E}\left[\left(R_t + \gamma Q^{\mu}_{\phi}(S_{t+1}, A_{t+1}) - Q^{\mu}_{\phi_0}(S_t, A_t)\right)^2\right] \\ \mathcal{L}^{\mu}\left[\theta\right] &= \mathbb{E}\left[Q^{\mu}_{\phi}(S_t, \mu_\theta(S_t))|\right]\end{aligned} \tag{3.3}$$

The subscript ϕ_0 indicates that the function-parameters used are those corresponding to ϕ, but gradients are not taken with respect to them (otherwise gradients flow in an acausal manner which gives rise to instabilities). The quantities shown are all differentiable and may be estimated from environmental data, thus the gradients may be estimated for gradient-descent.

3.2.2. MADDPG

MADDPG, introduced in [16], is an extension of the DDPG algorithm to the multi-agent case. The core problem addressed is that independent learning does not always converge correctly. MADDPG is different from DDPG in that the ”critic” functions are permitted access to the data from the whole environment (including data which is hidden from the agent); thus the actor, which we wish to have access only to data available to it at execution time, is still able to learn to behave intelligently in response to other agents' changing policies.

The reason for giving each part of the algorithm access to different data is that we ultimately wish the actor's to behave without knowing any hidden information from their opponent - however, as we have demonstrated, it is necessary to take some account of opponents' behavior in order to achieve an equilibrium. The compromises solution is to allow the critic function Q to act as a ”computational middleman” - that is, it is allowed access to data in training which it's agent should not actually know. However, the policy's gradient-step takes an expectation over this hidden information anyways - in the ideal case, the gradient-step would be the same as in DDPG. Because all agents are learning at once, however, the DDPG Q-function will fluctuate as agents change their policies - the MADDPG Q-function will assign disparate actions to different ”bins” instead of lumping them together (which bins do *not* depend on agents' policies). Thus, when opponents change their policy, MADDPG must account for this in the policy loss - however it does *not* need to simultaneously re-learn the Q-function.

Mathematically, each agent is defined, as before, by an actor μ_{θ^i} and a critic Q_{ϕ^i} the only change is that the Q_{ϕ^i} now have the type signature $Q_{\phi^i} : \mathcal{S} \times \bigtimes_{j=1}^{N} \mathcal{O}^j \times \mathcal{A}^j$. The appropriately altered loss function for the Q is

$$\mathcal{L}^{Q^i}\left[\phi_i\right] = \mathbb{E}\left[\left(R^i_t + \gamma Q_{\phi^i}(S_{t+1}, O^i_{t+1}, A^i_{t+1}, O^{-i}_{t+1}, A^{-i}_{t+1}) - Q_{\phi_0}(S_t, O^i_t, A^i_t, O^{-i}_t, A^{-i}_t)\right)^2\right] \tag{3.4}$$

(It is of note that this Q-function makes use of all the agents' observations o and the full hidden state)

Of particular interest to the cases we discuss below, when the environment is stateless, the Q function is then totally independent of all agents' policies (because in the general case, Q depends only on agents' policies through the state-transition distribution). That is, for the simple day-ahead clearing market described previously and below, the Q-function has a global minimum independent of all agents' policies - training agents with respect to it, then, if it converges, must converge to a valid Nash-Equilibrium. For a formal statement of the conditions and proof, see Section 2.3.

We note that the puzzle of non-stationarity is that both DDPG (subscript D) and MADDPG (subscript M) policies, should ideally be the same; if everything converges perfectly, we have:

$$\begin{aligned}\pi^i_M = \text{argmax}_{a^i} Q_M(a^i, \pi^{-i}_M) = \text{argmax}_{a^i} r(a^i, \pi^{-i}_M) \\ \pi^i_D = \text{argmax}_{a^i} Q_D(a^i) = \text{argmax}_{a^i} r(a^i, \pi^{-i}_D)\end{aligned} \tag{3.5}$$

That is, the policies are, in theory, the same. The problem is a combination of the estimation of the Q-functions with the exploration policies.

3.3. Extension to the Continuous Day-Ahead Market

3.3.1. Neural Network Technicalia

For the sake of transparency we note here some of the parameter values used in training. All neural networks used in this work have two hidden layers of 300 nodes, with Layer-Norm and ReLU activations.

The Adam optimizer was used. For exploration, the policy-generated act was added to zero-mean Gaussian noise with standard-deviation 0.3. All acts were scaled such that 1.0 corresponded to Q_{max} or MC. Rewards used for learning were scaled down by a factor of 1000. In all cases in this work, exploration occurs for the first half of the run only, while learning occurs throughout. A replay buffer large enough to store one quarter of the total episodes for learning was used (storing in FIFO order); batches of size 128 were drawn at random from this buffer; learning updates occurred every 128 timesteps.

3.3.2. Price-Only (Bertrand), Hypercompetitive

The first set of continuous experiments consists of testing these DeepRL algorithms in a continuous environment under Bertrand competition. The reason for this is to highlight the connection between the tabular experiments - restricting to Bertrand competition allows the calculation of the best-response function, and thus the optimality-deficit; this is not readily done when agents are allowed to submit both quantity- and price-bids.

We are interested in verifying the adequate implementation of the DeepRL algorithms, and in illustrating the continued failure of independent learning due to non-stationarity. As in the tabular case„ we examine a simple day-ahead clearing market; for this case, we restrict ourselves to price-only bidding in the hypercompetitive regime (in which each agent can unilaterally meet all demand), as this allows comparison of agent performance to the expected analytical solution of the Bertrand market discussed above. All agents' marginal costs are $c = 40$USD/MWh, while demand (in USD) is given by $P_D(Q) = 500 - 2Q$. Clearing occurs in as-bid merit-order until the price at the quantity contracted equals the demand at that price. Negative quantities and negative prices are forbidden, though agents are expected to learn to bid above their marginal costs rather than being hard-coded to do so. Shortage is impossible as the Bertrand model assumes non-bindingness of capacity constraints.

We examine four cases in which agents compete only with respect to price (a la Bertrand competition): in the first, the DDPG agent is a monopolist (and so is expected to converge to the monopoly price); this provides evidence as to whether the learners have been implemented correctly. In the second, a DDPG agent competes with a naive marginal-cost bidder (the maximum profit is thus zero); this should illustrate that the non-stationarity problem does not occur when the other agents are fixed. In the third, two DDPG agents are pitted against one another in hopes of illustrating the failure to converge to a Nash equilibrium, analogous to the discrete case. Finally these two DDPG learners are replaced with MADDPG learners, which it is hypothesized will suffice to ensure that a correct equilibrium is learned.

3.3.3. Aside: Bertrand Analytical Solution

We begin our experiments by seeking to replicate the well-known analytical solution for Bertrand competition; Bertrand competition models firms which produce at the same, constant marginal cost subject to no capacity constraint for a fixed, downward sloping demand-curve. This solution is useful because it is possible to derive both the Nash equilibria and the best-response functions of the firms involved.

Suppose we have a demand function $Q_D(P)$ (with corresponding inverse $P_D(Q)$), along with N firms selecting selling-prices p_i and producing at constant marginal cost c. We assume that all production is awarded to the firm with the lowest bid, with ties being broken evenly. Fix a particular agent i; when any other agent bids at or below the common marginal cost c, the agent's best-response is to likewise bid c. If the minimum of other agents' bids is between c and the monopoly price, the best response is to bid this same minimum price less an arbitrarily small positive number ε (in order to ensure it is awarded the full amount rather than tie). Finally, when all other firms have bid above the monopoly price, the best-response is to bid the monopoly price. Formally, if not illuminatingly, we may write this

$$p^i_{\text{BR}}(p^{-i}) = \min(p_{\text{monopoly}}, \max(c, \min_i(p^{-i}) - \varepsilon)) \tag{3.6}$$

Clearly, for any number of agents $N > 1$, the only Nash-equilibrium is for all players to bid the marginal cost. The principal reason for considering Bertrand competition in this section of the work is that it admits a simple reformulation of the deficit formulated in the previous chapter. The deficit is simply the agent's best-response profit less its realized profit.

$$\lambda^i(p^{-i}) = Q^i_D(p^i_{BR}(p^{-i}), p^{-i})(p^i_{BR}(p^{-i}) - c) - Q^i_D(p^i, p^{-i})(p^i - c) \tag{3.7}$$

Where the contracted quantity $Q^i_D(p^i, p^{-i})$ is the amount awarded to agent i given other agents' actions - 0 if another agent undercuts it, and otherwise the whole demand $Q_D(p^i)$.

As before, this profit deficit may be considered a measure of how well agents converge to Nash equilibria.

3.3.4. Q,P Bidding, Competitive

The purpose of the second set of continuous-environment experiments is to illustrate that the same problems of non-stationarity occur as previously when agents participate in a more complex market. In this case, the action-space of the agents is extended to permit an agent to choose both the price and quantity of its bid. Further, the capacity of each agent is lowered to $Q_{\text{max}} = 200$MW so that no individual agent can meet all demand alone. Because both quantity and price are set independently, the analytical solution is fairly complicated compared to the Bertrand case; instead we report profits as a proxy for relative performance. This increase in complexity is meant to illustrate that DDPG learners are capable of operating in (slightly) more complex environments, while still failing when pitted against one another. In such a case, it is hypothesized that the centralized MADDPG will be capable of converging where independent-learning algorithms like DDPG fail to do so.

As in the previous cases, agents have a uniform marginal cost $c = 40$USD$/$MWh, while demand (in USD) is given by $P_D(Q) = 500 - 2Q$. Clearing is again in as-bid merit-order, with payment set at a uniform clearing price. In case of shortage, the clearing price is the willingness-to-pay of demand for the total offered quantity.

3.4. Adversarial Market-Design

3.4.1. Description of the Environment

Having examined the behaviour of various algorithms in the simple day-ahead market, experiment three to analyzing more complex models of the electricity market which allow for the fruitful application of MARL methods. This third set of experiments is conducted according to Cournot competition - firm's submit quantity-bids only, with the priced determined according to the total quantity offered. The key difference is the introduction of an adversary-agent who sets a market price-cap. The reason for selecting Cournot competition is that it allows for ready computation of the best-response function, and requires fewer modifications to accomodate the adversarial price-cap than would the Bertrand solution. The goal of this experiment is to examine the performance of an adversary tasked with optimizing the market to maximize a certain social welfare-function, while learning agents *simultaneously* learn to behave strategically as the adversary adjusts the market-rules.

For ease of comparison between our experiments, we maintain as many of the same parameters as possible; the notes in Subsection 3.3.1 pertain to the neural networks used here as well, for both the adversary and the firms. The adversary's action-space is a single continuous scalar, the price-cap; firms' actions are quantity-only bids of the same form as before. As previously, no agents observe any state-information - instead the channel for information to flow is the agents' critic functions. All agents use MADDPG to learn.

As in the previous cases, firms have a uniform marginal cost $c = 40$USD$/$MWh, while demand (in USD) is given by $P_D(Q) = 500 - 2Q$. The clearing price is set according to the total quantity bid (again, agents submit only quantity bids) - the price cap is implemented as noted in Section 2.4 as, in effect, an alteration to the demand-curve.

The purpose of this experiment is not, of course, to present policy-advice based on such a simplistic model; instead, it is hoped that the simplicity of the model will render the application and its limitations more legibly than a fully realistic environment. A real application of this technique would entail careful selection of market-designs such that they can be usefully represented as the output of a neural network, as well as a more thoroughgoing consideration of the appropriate state- and action-spaces.

3.4.2. Price-Cap Analytical Solution

In what follows, we will consider a market-designer attempting to optimize the social-welfare function

$$\text{SW} = 2\int_0^{Q^*} dq(\alpha P_D(q) - (1-\alpha)P_S(q)) + (1-2\alpha)P^*(Q^*) \tag{3.8}$$

(a social welfare-function which (dis-)privileges consumer welfare relative to producer welfare according to a factor α ($0 \leq \alpha \leq 1$)).

For ease of reference, we call α the "adversariality", as it dictates how strongly "adversarial" the market agent is to the firm agents.

In contrast to the typical mechanism-design approach which seeks DSIC payment rules from truthful bids, one may wish to find the optimal parameter-value for a given family of market-designs. This might be of use in, say, setting an optimal price-cap, as in the example we will consider; it again bears noting that the present example is meant as an illustration of the technique rather than an attempt to solve any particular market-design problem.

The fact which makes this example interesting is that, rather than requiring truthful bidding, MARL methods allow us to train a market-designer seeking to maximize social-welfare while *simultaneously* training agents to simulate strategic behavior under a given market rule! The work in previous sections allows us to be comfortable that the centralized-decentralized training (a la MADDPG) regimen should allow the correct derivation of equilibria, at least in principle.

The setup is as follows: N producer firms are in posession of identical generators with $MC = 40$USD. For the sake of crisply highlighting the effect on strategic behavior, we consider a Cournot model in which agents compete solely on quantity without capacity constraints, with the corresponding price determined by the market. The demand is modelled as a simple linear function $P_D(Q) = b - mQ = 500 - 2Q$. When demand exceeds supply, the price is set to the demand's price offer for the supplied quantity.

The price-cap to be set by the market-designer agent, $\bar{p}$, implemented by replacing the consumer-demand with an effective demand-curve $P_{D,\text{eff}}$

$$P_{D,\text{eff}}(Q) = \begin{cases} \bar{p} & P_D(Q) \geq \bar{p} \\ P_D(Q) & P_D(Q) \leq \bar{p} \end{cases} \tag{3.9}$$

This said, we may fix the price-cap in order to find the oligopoly bids (as a function of $\bar{p}$); then, we may derive an analytical expression for the optimal price cap from the social-welfare function.

Denote the total quantity bid by Q, and individual agents' bids by q^i. Now for a given agent, the profit obtained is

$$\Pi^i = (P_D(Q) - c)q^i \tag{3.10}$$

In the absence of any price-cap, the agent's best response to a set of other agents' bids Q^{-i} would be to bid

$$q^i_{\text{BR}}(Q^{-i}) = \frac{1}{2m}(b - c - mQ^{-i}) \tag{3.11}$$

Which, applied to all agents, yields the equilibrium bid:

$$q^i_{\text{olig}} = \frac{b-c}{m(N+1)} \tag{3.12}$$

with corresponding price and individual firm profit

$$\begin{aligned} p_{\text{olig}} &= \frac{b + Nc}{N+1} \\ \Pi^i_{\text{olig}} &= \frac{1}{m}\left(\frac{b-c}{N+1}\right)^2 \end{aligned} \tag{3.13}$$

Once the price cap is in place (assuming it is above the marginal cost, as the optimal bid is otherwise trivially zero), the above solution is no longer necessarily an equilibrium - where a decreasing demand would have offset the gains from increasing quantity, now this demand is, for some range of quantities, fixed. If the price-cap is above the oligopoly-price, the oligopoly solution maintains, as it remains the point at which any unilateral change in a quantity is penalized.

On the other hand, if the price-cap is below the oligopoly-price, the "crossover-point" (the quantity demanded by consumers at the price-level set by the price-cap, $P_D(Q) = \bar{p}$) is the equilibrium solution. To see this, note that, while below both the oligopoly price the price-cap, any unilateral restriction in quantity increases a firm's profit. This profit increases until the price-cap is hit, at which point a decrease in quantity no longer changes demand, so that decreasing quantity is now penalized.

The best-response bid can be calculated by examining the sign of the cap-modified profit (again considering only the case where the cap is above the marginal cost). The derivative of the profit in the

price-capped region is positive by assumption; at the crossover point, this either becomes negative, so that the crossover point is optimal, or remains positive, so that the capless best-response maintains. The post-crossover profit derivative is positive as long as the capless best-response is greater than the crossover quantity.

Then

$$q^i_{\text{BR}}(Q^{-i}, \bar{p}) = \begin{cases} \frac{b-\bar{p}}{m} - Q^{-i}, Q^{-i} < \frac{b+c-2\bar{p}}{m} \\ \frac{b-c}{2m} - \frac{1}{2}Q^{-i}, \text{else} \end{cases} \tag{3.14}$$

The equilibria are now slightly more diverse - when the oligopoly price is realizable, this is the sole equilibrium. However, if the price-cap prevents this, then *any* set of bids which sum to $\frac{b-\bar{p}}{m}$ is an equilibrium. The producer surplus is

$$\text{PS} = \begin{cases} \frac{N}{m}\left(\frac{b-c}{N+1}\right)^2, \frac{b+Nc}{N+1} \leq \bar{p} \\ \frac{1}{m}(b-\bar{p})(\bar{p}-c), \text{else} \end{cases} \tag{3.15}$$

The consumer surplus is then

$$\text{CS} = \begin{cases} \left(\frac{N}{N+1}\right)^2 \frac{(b-c)^2}{m}, \frac{b+Nc}{N+1} \leq \bar{p} \\ \frac{1}{2m}(b-\bar{p})^2, \text{else} \end{cases} \tag{3.16}$$

Thus, for our market designer (who weighs consumer surplus by a factor 2α and producer surplus by a factor $2(1-\alpha)$), the social welfare function is

$$\text{SW} = \begin{cases} 2\frac{(b-c)^2}{m(N+1)^2}(\alpha N + (1-\alpha)N^2), \frac{b+Nc}{N+1} \leq \bar{p} \\ \frac{b-\bar{p}}{m}(\bar{p}(3\alpha-1) + (b(1-\alpha) - 2c\alpha)), \text{else} \end{cases} \tag{3.17}$$

Because of the many overlapping constraints, the inversion to obtain the optimal price cap is not readily attainable or simply expressed; instead, we note a few points to guide the reader's intuition in interpreting the experimental results below.

First, in general, to the extent that they are free to choose their prices, the strategically acting firms should cause a deadweight loss. Clearly, when $\alpha = 1$, the optimal price cap is the marginal cost (plus an arbitrarily small real number) - this maximizes not only the consumer welfare, but the total social welfare for any $\alpha > 1/2$ - since lowering the cap in this regime converts producer welfare into a strictly greater amount of consumer welfare (which is also valued more). When $\alpha < 1/2$, the price cap can force the oligopolists to produce a quantity greater than the oligopoly quantity, but not less - thus the optimal cap in this region involves trading off (relatively more valuable) producer welfare to consumer welfare in an attempt to recoup by decreasing the deadweight loss.

4

Results

In this chapter, we present the results of the experiments carried out as described in Chapter 3. Each section corresponds to one of the three experiment classes discussed previously, presenting two examples of the full results plots, followed by a summary figure (the remaining results plots are to be found in the appendix). Section 4.1 deals with the tabular experiments, Section 4.2 with the continuous bid-space analogue, and Section 4.3 with the adversarial market-design experiments. While abbreviated remarks are offered to highlight the relevant features of the results, all interpretative discussion is carried out in Chapter 5; the reader interested primarily in the interpretation of the results is encouraged to go to this section directly after familiarizing themselves with the summary figures.

4.1. Tabular Q-learning Results

The four experiments performed in the tabular environment examine the behavior of either one or three agents, which either do or do not have access to hidden information (i.e., each combination of these conditions is tested and shown in the summary figure).

4.1.1. Interpreting the Tabular Experiments

(**Note** that these interpretative guides hold also for the continuous-space experiments which use the Bertrand solution upon interchanging ”TD-error” for ”Critic-Loss”)

Here we present two schematic illustrations of the type of plots which are to follow. In this experiment, the primary question we wish the data to answer is ”Does the agent arrive at a persistent final strategy, and if so, is it a best-response given other players’ behavior?”.

The first of these conjuncts is answered with reference to the TD-error. When the agent-averaged TD-error goes to zero, the agents are no longer surprised by the outcome of their actions, and, because they play the argmax-policy, their policy ceases to change. This is illustrated schematically in figure 4.1. The final policies these agents play need not be correct Nash-equilibria.

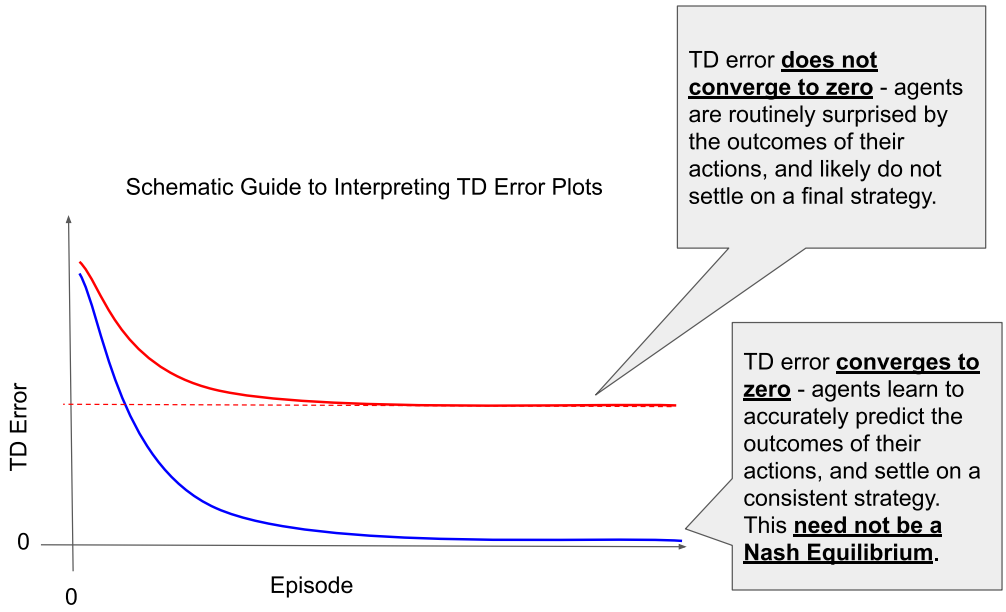


Figure 4.1: Schematic guide to the interpretation of TD-error plots shown for Experiment 1. Firms arrive at a consistent policy only insofar as their TD-error tends towards zero. These policies need not be best-responses.

The second set of data we are concerned with is the optimality-deficit. This quantity measures how far an firm's policy is from being a best-response; when it is zero, all agents are playing a best-response, and thus have arrived at a correct Nash-equilibrium. This is illustrated in Figure 4.2. This quantity makes sense only if firms' policies have ceased to change substantially, as indicated by the TD-error; if the TD-error remains quite high despite training, this measurement is not especially meaningful.

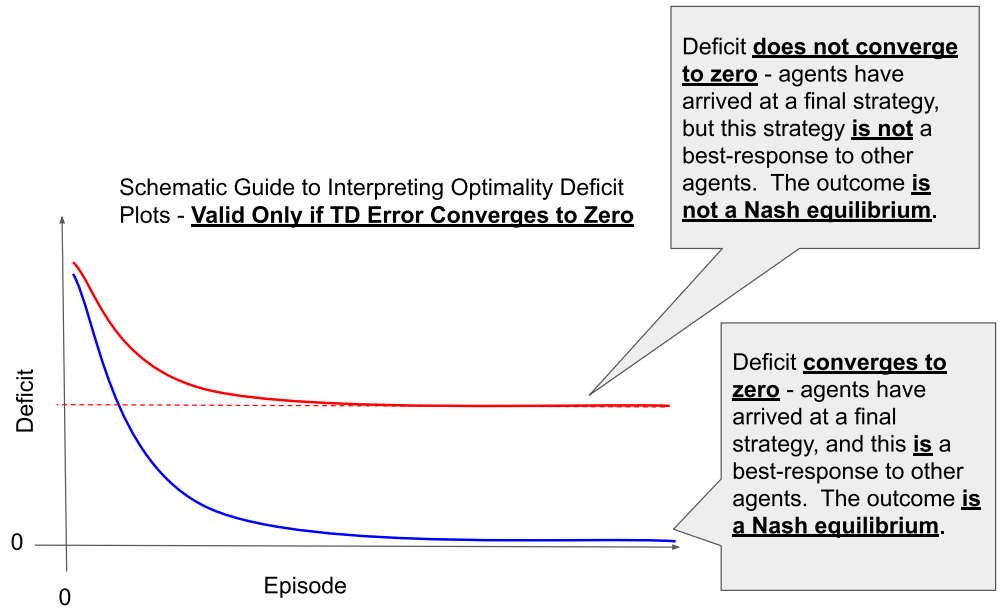


Figure 4.2: Schematic guide to the interpretation of optimality-deficit plots shown for Experiment 1. Provided that the TD-error tends to zero (so that firms play a persistent final policy), the optimality-deficit will tend to zero if and only if these final policies are a Nash equilibrium.

4.1.2. Example 1: Single Agent Monopolist, Stateless

We now present two example cases examining the convergence properties of the TD-error and the optimality-deficit, in order to illustrate the learning dynamics; data for those tests which are not used as examples may be found in part in the summary figure, Figure 4.5, and in full in the appendix.

The first case examines the simplest possible configuration, in part to ensure the functionality of the environment; there is a single agent and no state-information. The results are shown in 4.3. The plots show a rapidly decreasing TD-error and optimality deficit, becoming near-zero within about 10,000 episodes.

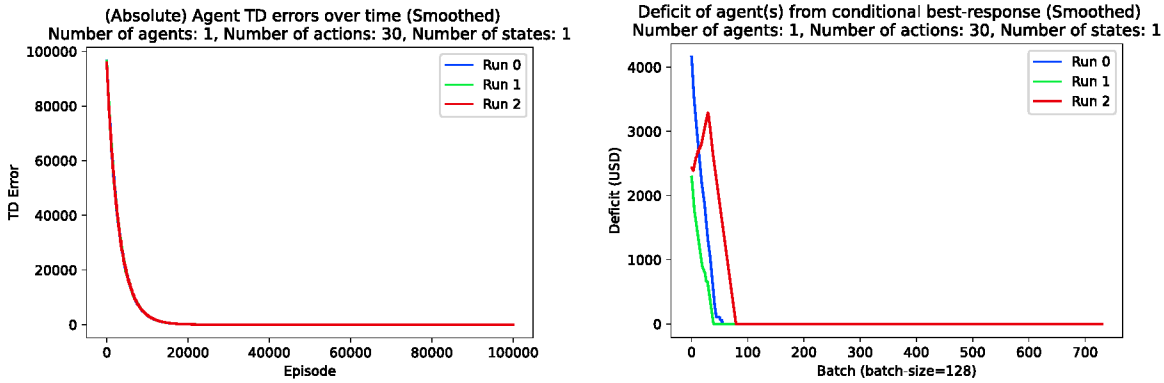


Figure 4.3: Plot of the (smoothed) agent TD-error (left) and optimality-deficit (right) for the case with one agent and no state. In the testing stage, the conditional-deficit goes to zero. Left: Absolute TD-error. Right: Optimality-deficit

4.1.3. Example 2: Three Agents, Stateful

The next example examines a cases with three agents in a case with four states (Q_{max} and MC each being allowed to take two possible values, observable only to the owning-agent). Here we observe that the TD-error does not converge to zero, while the optimality-deficit remains materially non-zero.

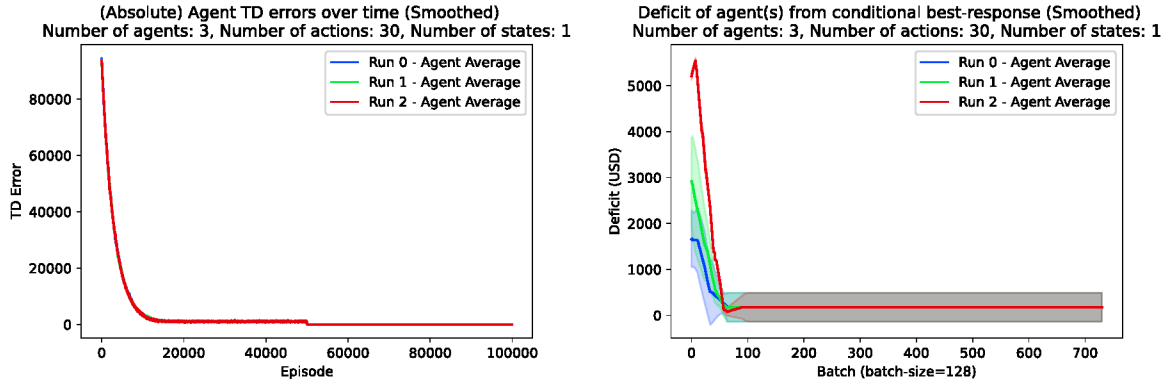


Figure 4.4: Plot of the (smoothed) agent TD-error (left) and conditional deficit (right) for the case with three agents and no state. The shaded region denotes standard-deviation over agents. The optimality-deficit is materially non-zero. Left: Absolute TD-error. Right: Optimality-deficit

Tabular Experiments Summary

Having examined these two examples, we present the data for the remainder of the tests; discussion of these results is given in 5.1.1, while the relevant data is given in Appendix A. In the summary figure, the final values used are the average of the relevant quantity over the final 1000 episodes; the variation indicated is between-runs, while the data-points are all averaged over the learning agents where appropriate.

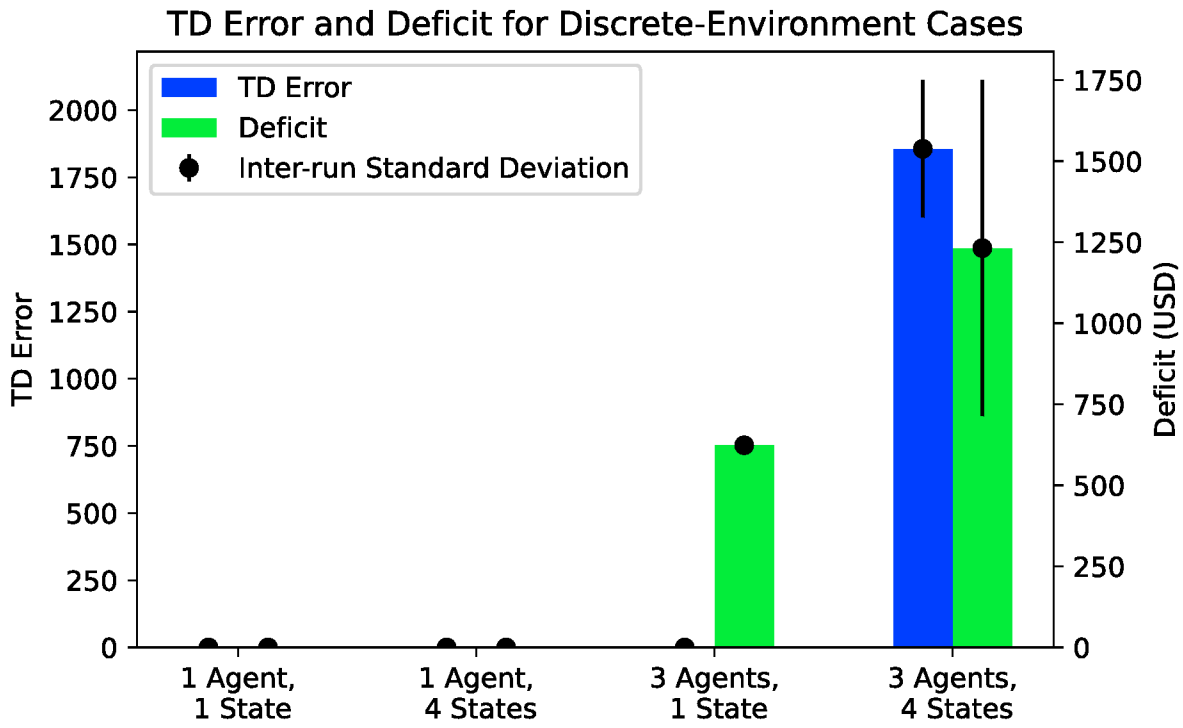


Figure 4.5: Summary of the tabular experiment results. For each case, the final TD-error and optimality-deficit are shown; uncertainty bars represent the standard deviation over all three runs. Left vertical axis corresponds to the scale of the TD-error, and the right to the deficit. The "final" value used is, in each case, the average over the most recent 1000 timesteps, or the coterminous batches; within each run, the values are averaged over all learning agents.

4.2. Continuous Environment Results

Note that, per the parameters described in 3.3, the monopoly solution is to sell at a price of $P = 270\text{USD} = 6.75c$, resulting in a quantity $Q = 115\text{MWh}$ being sold for a total profit of $\Pi = 26450\text{USD}$.

We perform two types of experiments in the continuous environment. In the first, firms submit price-only bids in the hypercompetitive regime, competing à la Bertrand; here we examine four cases, the first a DDPG monopoly, the second a DDPG agent competing with a naive, marginal-cost bidding generator, the third competition between two DDPG agents, and the fourth competition between two MADDPG agents.

In the second set of experiments, agents submit (Q,P) bid pairs (instead of price-only), and no longer compete in the hypercompetitive regime. The cases examined in these experiments are the same as those described immediately above, save for the omission of the monopoly scenario.

4.2.1. Continuous Environment, Bertrand Competition

Example 1: DDPG vs DDPG, Bertrand Competition

In this first example, we go directly to examining the behavior of two DDPG agents when placed in competition with one another. Figure 4.6 shows data pertaining to the agents' actions, while Figure 4.7 illustrates the learning performance. We see that the optimality deficit does not immediately converge to zero, while the TD-error, despite some fluctuation, does.

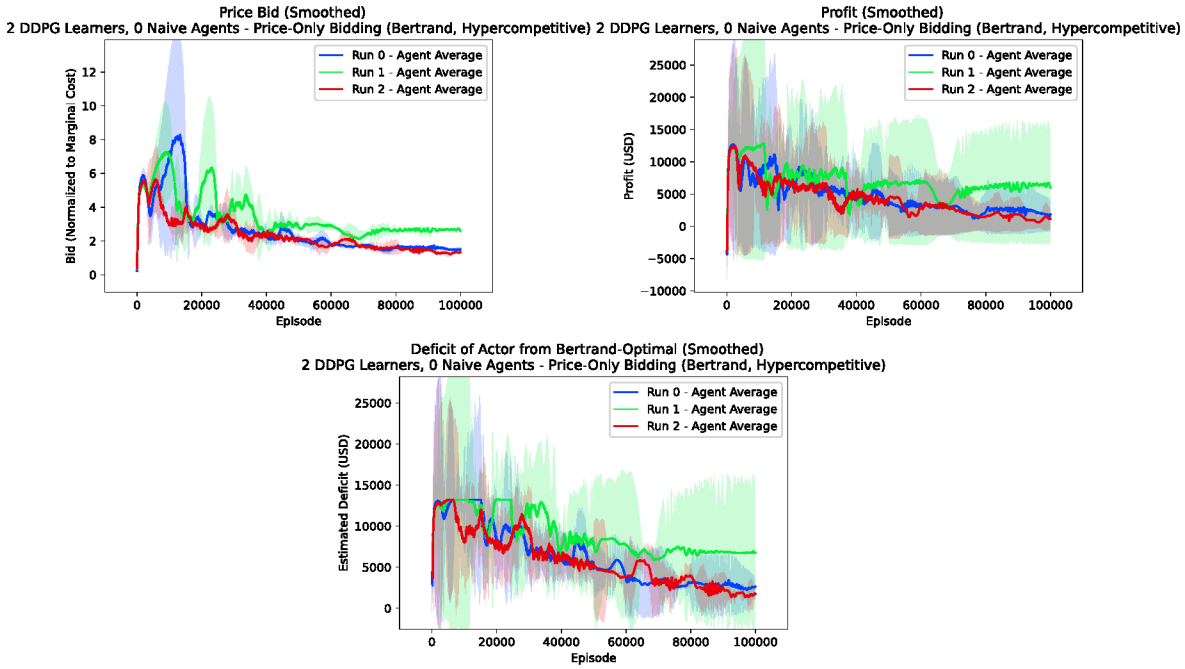


Figure 4.6: Bertrand competition: DDPG vs DDPG. Plots illustrating the performance of two competing DDPG algorithms for a simple, price-only market. All plots are shown for each of three independent runs of the simulation; plotted curves are averaged over all learning-agents (i.e. excluding marginal-cost bidders). Top-left: Price bids over training. Top-right: Profit earned. Bottom: Deficit with respect to analytical Bertrand solution

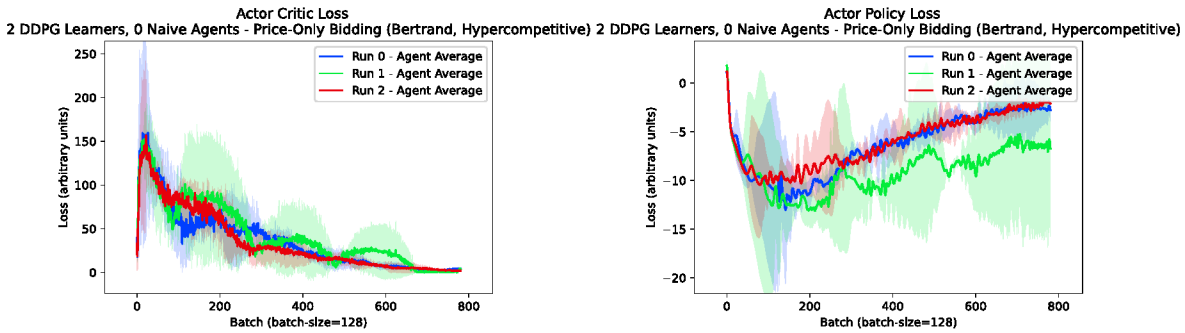


Figure 4.7: Bertrand competition: DDPG vs DDPG. Plots illustrating the learning performance of two DDPG learners pitted against one another in a simple, price-only market. All plots are shown for each of three independent runs of the simulation; plotted curves are averaged over all learning-agents. Left: Learner critic loss. Right: Learner actor loss.

Example 2: MADDPG vs. MADDPG, Bertrand Competition

In this second example, we examine the behavior of two firms trained under MADDPG when placed in competition with one another. Figure 4.8 shows data pertaining to the agents' actions, while Figure 4.9 illustrates the learning performance. We see that the optimality deficit does not immediately converge to zero, while the TD-error, despite some fluctuation, does.

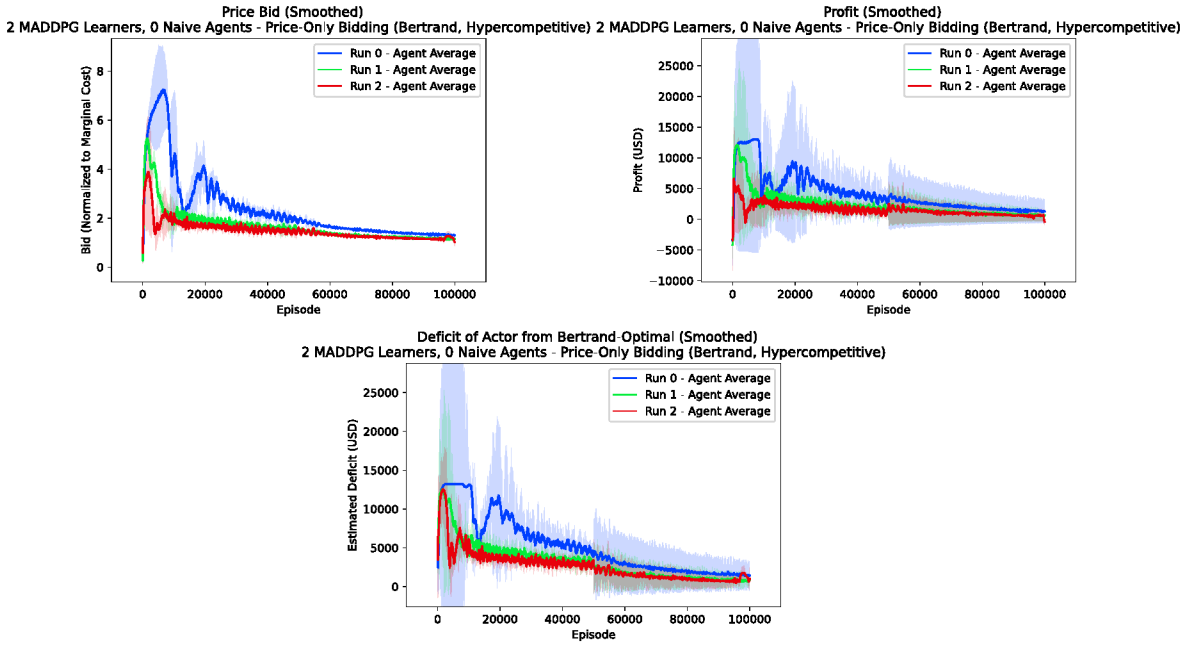


Figure 4.8: Bertrand Competition: MADDPG vs MADDPG. Plots illustrating the performance of two competing MADDPG algorithms for a simple, price-only market. All plots are shown for each of three independent runs of the simulation; plotted curves are averaged over the two learning-agents. Top-left: Price bids over training. Top-right: Profit earned. Bottom: Deficit with respect to analytical Bertrand solution

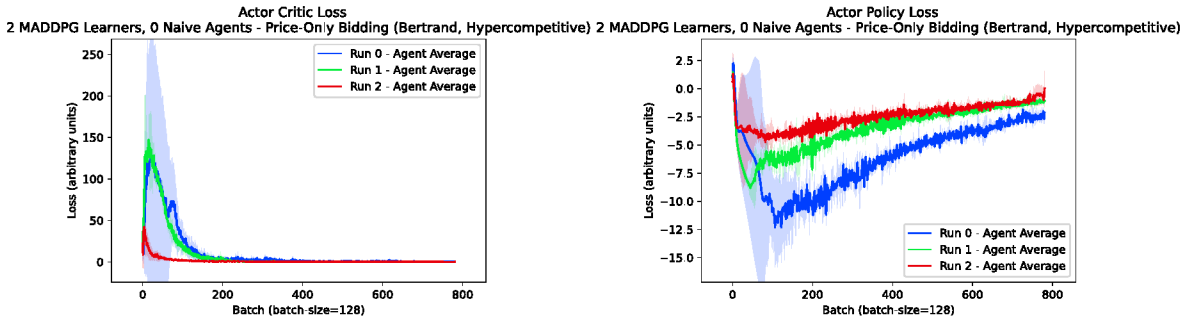


Figure 4.9: Bertrand Competition: MADDPG vs MADDPG. Plots illustrating the learning performance of two MADDPG learners pitted against one another in a simple, price-only market. All plots are shown for each of three independent runs of the simulation; plotted curves are averaged over all learning-agents. Left: Learner critic loss. Right: Learner actor loss.

Continuous Environment, Bertrand Competition Summary

Figure 4.10 summarizes the performance of the agents in each of the cases. These results are discussed in Chapter 5.

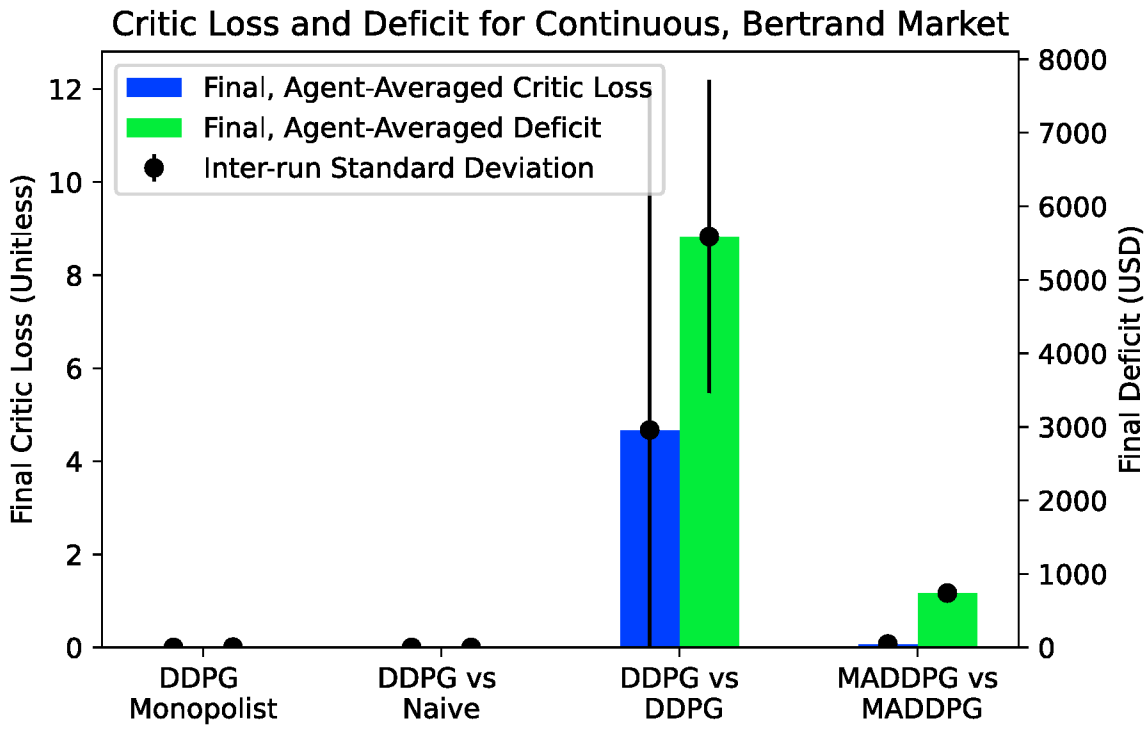


Figure 4.10: Summary of the continuous environment experiment results for Case 1 (hypercompetitive, Bertrand competition). For each case, the final critic-loss and optimality-deficit are shown; uncertainty bars represent the standard deviation over all three runs. Left vertical axis corresponds to the scale of the TD-error, and the right to the deficit. The "final" value used is, in each case, the average over the most recent thousand timesteps, or the coterminous batches; within each run, the values are averaged over all learning agents.

4.2.2. Continuous Environment, Q,P-Bidding

In this section we summarize the experiments in the continuous environment in which agents submit quantity-price pairs as actions, and can no longer unilaterally meet demand. We observe in Figure 4.11 that in all cases, the learners converge to achieving an appreciable profit, if not without some subtantial fluctuations. In Figure 4.12, we see the critics converge relatively rapidly, while the actors exhibit some fluctuation.

Example 1: DDPG vs. DDPG, Q,P-Bidding

Here we examine the case in which two independent DDPG learners compete with one another.

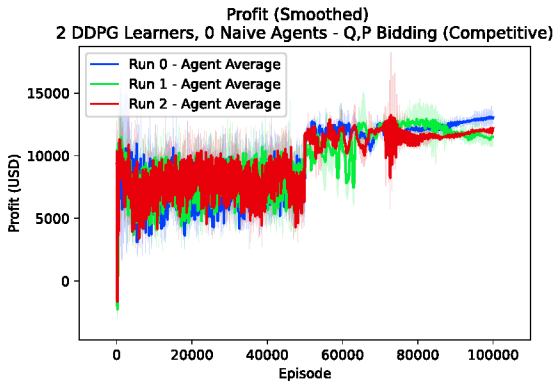


Figure 4.11: Q,P-Bidding: DDPG vs DDPG. Plot illustrating the profits of two independent DDPG learners for Q,P-bidding. Plots are shown for each of three independent runs of the simulation; curves are averaged over the two learners.

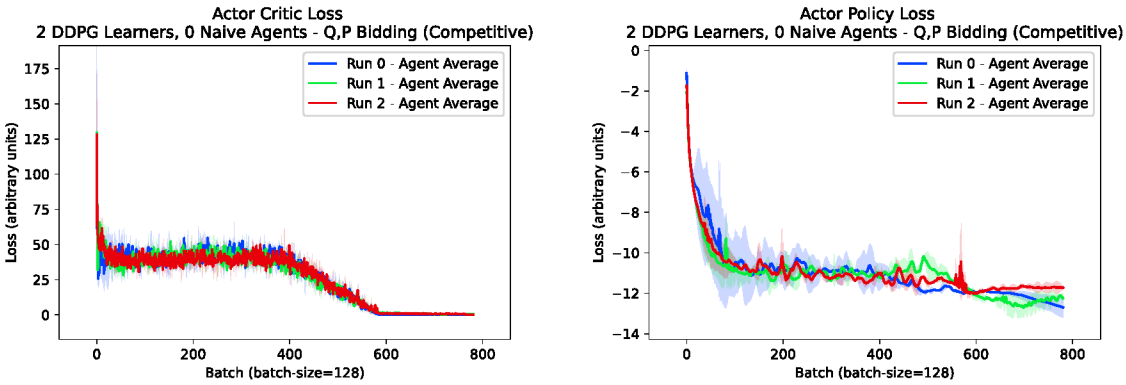


Figure 4.12: Q,P-Bidding: DDPG vs DDPG. Plots illustrating the performance of two independent DDPG learners for Q,P-bidding. All plots are shown for each of three independent runs of the simulation; plotted curves are averaged over the two learners.

Example 2: MADDPG vs. MADDPG, Q,P-Bidding

Here we examine the case in which two independent MADDPG learners compete with one another. We observe in Figure 4.13 that in all cases, the learners converge to achieving an appreciable profit, if not without some subtantial fluctuations. In Figure 4.14, we see the critics converge quite rapidly, the actors seem to fluctuate quite dramatically. Surprisingly, the MADDPG learners seem to flucuate much more dramatically than their DDPG counterparts.

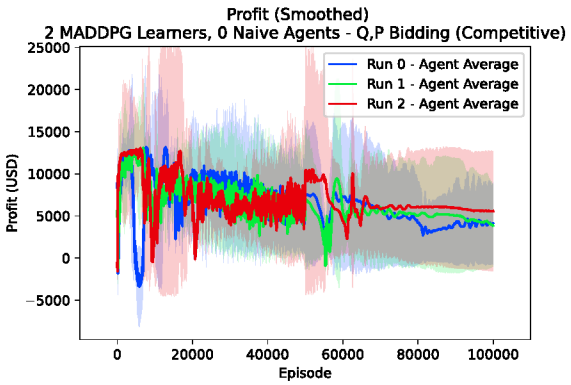


Figure 4.13: Q,P-Bidding: MADDPG vs MADDPG. Plots illustrating the performance of two MADDPG learners for Q,P-bidding. All plots are shown for each of three independent runs of the simulation; curves are averaged over the two learners. Left: price bids over training.

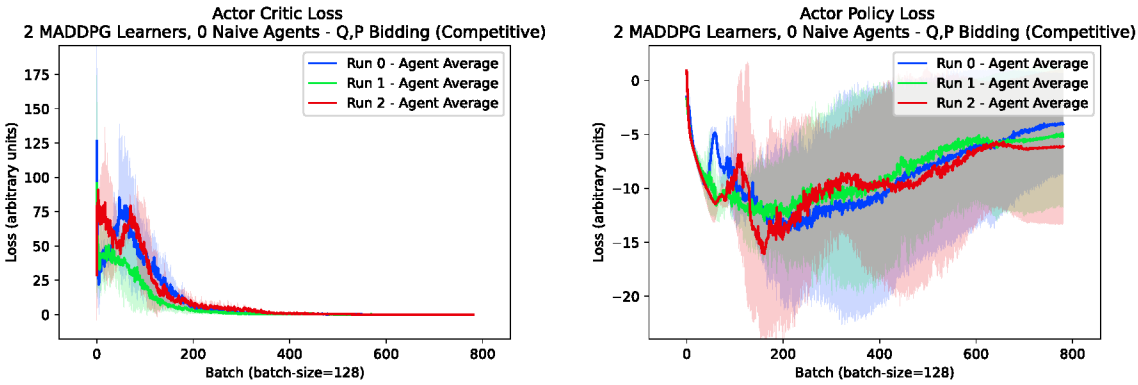


Figure 4.14: Q,P-Bidding: MADDPG vs MADDPG Plots illustrating the performance of two MADDPG learners for Q,P-bidding. All plots are shown for each of three independent runs of the simulation; plotted curves are averaged over the two learners. Left: actor critic loss. Right: actor policy loss.

Continuous Environment, Q,P-Bidding Summary

Figure 4.10 summarizes the performance of the agents in each of the Q,P-Bidding cases. These results are discussed in Chapter 5.

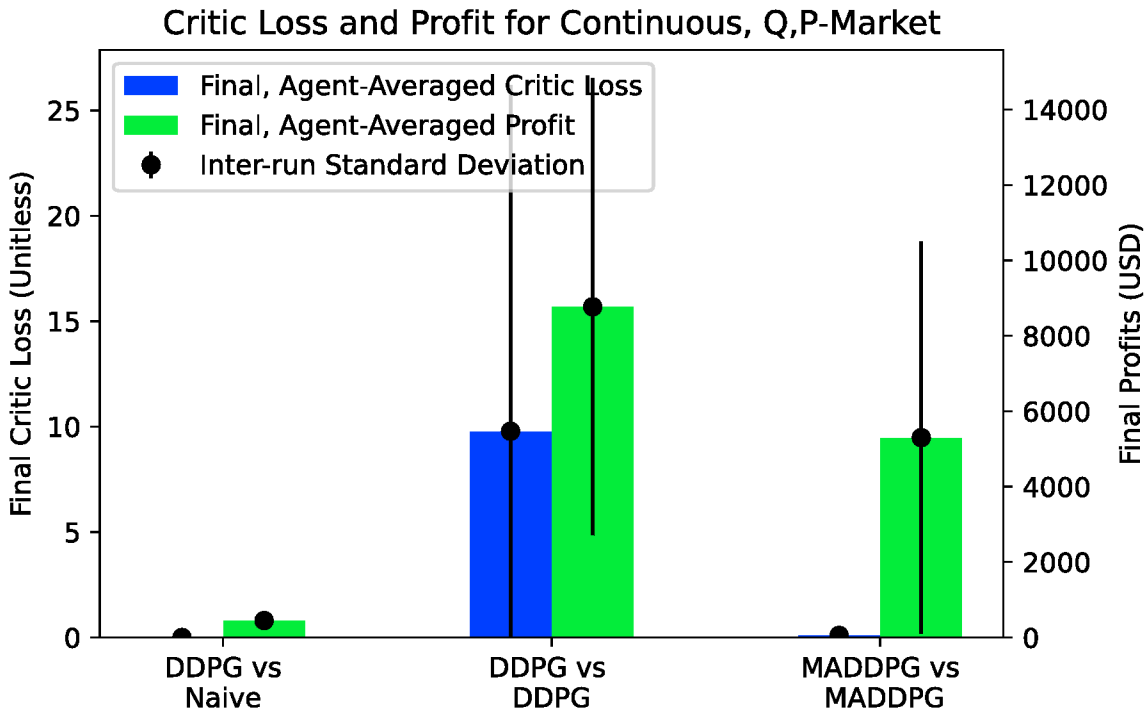


Figure 4.15: Summary of the continuous environment experiment results for case 2 (competitive, Q,P-competition). For each case, the final critic-loss and profit are shown; uncertainty bars represent the standard deviation over all three runs. Left vertical axis corresponds to the scale of the TD-error, and the right to the deficit. The "final" value used is, in each case, the average over the most recent thousand timesteps, or the coterminous batches; within each run, the values are averaged over all learning agents.

4.3. Adversarial Market-Design

Here we present the results of the experiment detailed in Section 2.4. We begin by presenting some example plots showing the training-progress over time. After these example plots have been shown, we present a "sweep" of parameter-space, in which we attempt to determine the optimal price-cap as a function of α and of N.

4.3.1. Example 1: One Firm, $\alpha = 0.25$

For our first example case, we run the experiment with a single producing firm, and a market-agent with adversariality $\alpha = 0.25$ - meaning that it should prefer to improve producer welfare rather than consumer welfare. Based on our earlier reasoning, we should expect to see little effect from the price-cap; the most producer-surplus is to be had at the oligopoly point.

And indeed - in spite of some erratic behavior (attributable to the fact that very high price-caps are equi-welfare and thus leave the neural-network optimizer with some 'momentum', causing outputs to continue to change), we observe in Figure 4.16 that in all three runs, the price cap tends to be very high - almost always above the monopoly price; further, the social-welfare function illustrates that the market-agent's welfare is practically the same as the agent's surplus.

With respect to the learning shown in Figure 4.17, it is noteworthy that even in this fairly simple case, both the actor and the critic exhibit fairly erratic behavior, though they do ultimately converge to a reasonable solution.

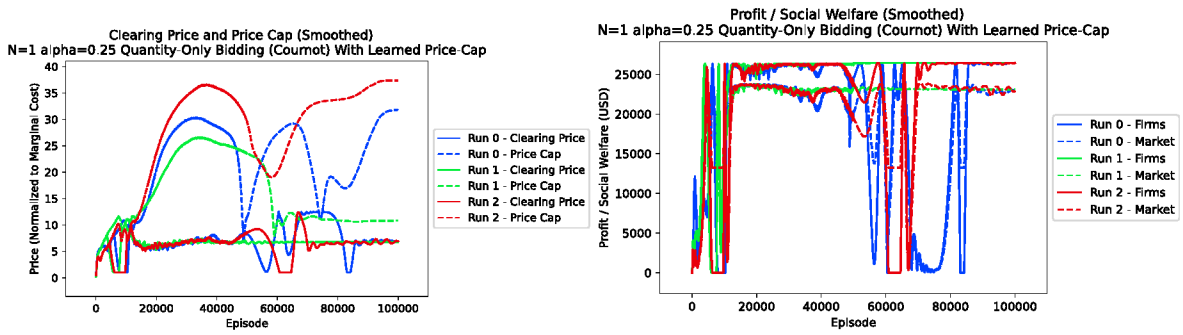


Figure 4.16: Adversarial market-design: One firm, $\alpha = 0.25$ Plots illustrating the learning progress of one firm agent and the market agent, with adversariality $\alpha = 0.25$. Left: clearing price and price cap. Right: profit and social welfare

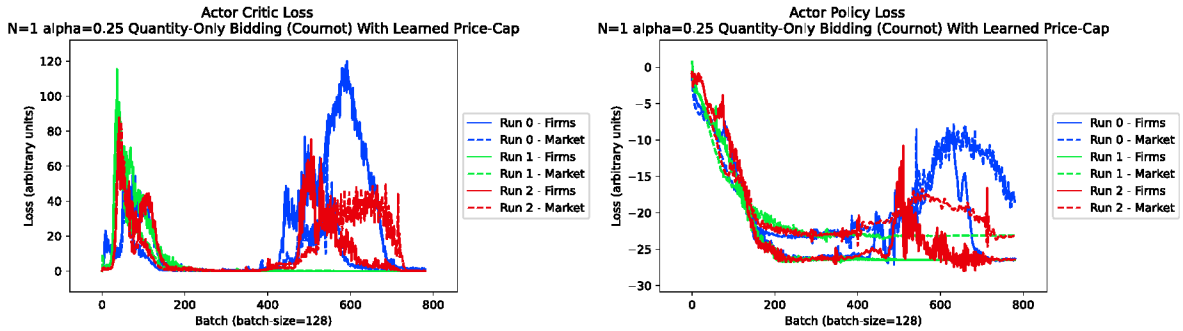


Figure 4.17: Adversarial market-design: One firm, $\alpha = 0.25$. Plots illustrating the learning progress of one firm agent and the market agent, with adversariality $\alpha = 0.25$. Left: critic loss. Right: policy loss

4.3.2. Example 2: Five Agents, $\alpha = 0.75$

We now perform the analogous experiment, though in this case, we have five power producing firms, and an adversariality of $\alpha = 0.75$ - generally favoring consumers. Since consumers can be favored without creating any deadweight loss, we should expect the price-cap to approach the generators' marginal costs.

Again, in spite of some erratic learning behavior, we see in Figure A.33 that the clearing prices tend to be low, while the market-agent's welfare tends to be high. It seems that much of the odd fluctuations of the price-cap coincide with periods of low clearing price - which is often, given the number of agents; as noted above, the market-agent when the price-cap is inactive, may learn erratically, though this is not a problem provided this does not interfere with the reward.

Perhaps surprisingly, we observe in Figure A.34 that the learning proceeds with relatively few hiccups - though there is a marked difference in the scales of the producers' loss compared to the market, this is not especially important provided convergence is achieved; this scale difference is due in part to the fact that the producers' rewards must be split among the five agents, while the market-agent keeps its own reward undivided.

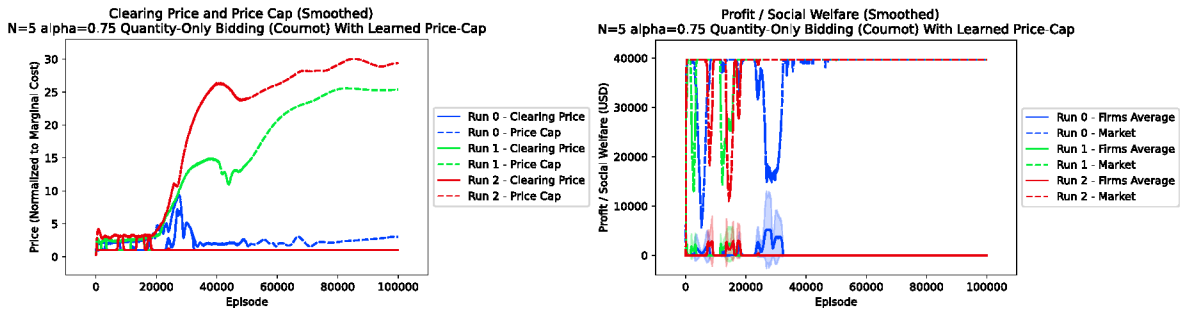


Figure 4.18: Adversarial market-design: Five firms, $\alpha = 0.75$. Plots illustrating the learning progress of five firm agents and the market agent, with adversariality $\alpha = 0.75$. Quantities for the firm-agents are averaged over all five agents, with the shaded region indicating the standard deviation over agents. Left: clearing price and price cap. Right: profit and social welfare

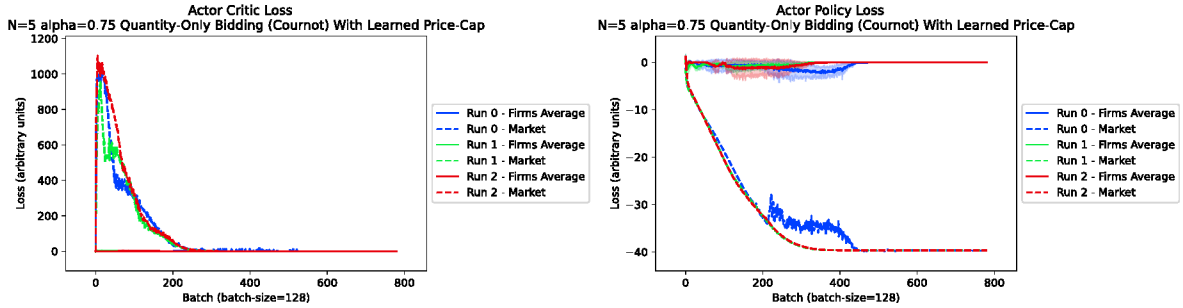


Figure 4.19: Adversarial market-design: Five firms, $\alpha = 0.75$. Plots illustrating the learning progress of one firm agent and the market agent, with adversariality $\alpha = 0.75$. Quantities for the firm-agents are averaged over all five agents, with the shaded region indicating the standard deviation over agents. Left: critic loss. Right: policy loss

4.3.3. Varying the Number of Firms and the Adversariality

Now that we have presented these two example cases, we may move on to the final part of this section. Here we present results of a "parameter sweep", showing what the final clearing prices and price caps were found to be after running the experiment above for each possible combination of (the number of agents) $N = 1, 3, 5$ and $\alpha = 0.0, 0.25, 0.50, 0.75, 1.0$ with three runs per data-point. The "final" values used are the average over the previous 1000 timesteps. We plot for reference the oligopoly prices for each N. Figure 4.20 summarizes this data; discussion is in Section 5.1.5.

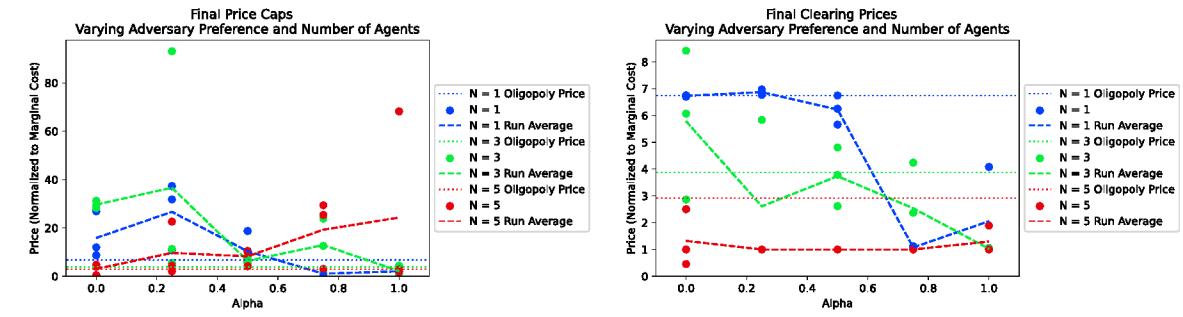


Figure 4.20: Adversarial market-design: parameter-sweep summary. Plots illustrating the behavior of the final clearing price and price cap. Left: final learned price-cap. Right: clearing-price

5

Discussion and Conclusion

This chapter discusses the results of the experimental results shown in Chapter 4, summarizes the argument of this work, and highlights limitations of the methodology used, pointing out potential directions for future work. The discussion of results is in Sections 5.1. The summary is in Section 5.2. The final remarks on applicability and directions for future work are in 5.3

5.1. Discussion of Results

5.1.1. Tabular Experiment Results

The relevant data are summarized in figure 4.5.

This data reveals that when there is only a single agent, they are entirely capable of learning to play optimally (monopolistically), both when there is no relevant state-information, and when it must adapt different behaviors to different market scenarios. This should not be surprising - the whole point being to examine non-stationarity, a phenomenon unique to situations with multiple agents. This does however, increase our credence that the environment model and learning algorithms have been implemented correctly.

By contrast, as expected, the case with three competing firms shows a deficit from optimal-play in both cases. When these agents compete with one another, the learning dynamics prevent them from accurately assessing and countering opponents' strategies. The case with three firms and no hidden state-information illustrates this most clearly - the agent-averaged TD-error is extremely low. This means that the agents have all learned to guess perfectly the profit they can expect on performing a particular action, and will continue to bid however they estimate will maximize this value. Yet these agents are playing suboptimally - a market-designer attempting to use this simulation to assess the possibility for strategic behavior would draw incorrect conclusions.

We note also that, as expected, the introduction of hidden-information in the form of differing generator-capacities increases the TD-error - firms will learn the average reward they get for bidding a certain way upon seeing they have a certain generator-capacity and routinely be surprised to find that the outcome is not as they expected. The point is, again, not that this is especially representative of a real-world scenario (as estimating an opponent's generator capacity and costs is hardly difficult, especially given access to bidding-histories), but instead that the limited expressiveness of typical Q-learning algorithms will cause them to fail when applied in complex situations.

The key to understanding how this can occur is to note that the agents have learned a ”self-fulfilling prophecy”; in the exploration-phase, each agent learns the value of a bidding strategy *averaged over the other agents' possible bidding strategies* - after exploration has ended, it will play the bid with the highest average value, and other agents will do the same. Perhaps this then results in some more learning and some switching of strategies, but eventually the agents will arrive at a point where they are all playing actions which they estimate to be the best, they will cease to play other actions (whose estimated values will not be updated to account for the *current* opponents' policies!). Thus once exploration has ended, the agents have trapped themselves into ”leaving money on the table”.

Now in 2.3 we discuss conditions under which this is *guaranteed* not to happen. The reason these guarantees do not hold is that the agents' Q-tables are not ”expressive” enough - they simply don't have

enough entries or dimensions to satisfy the conditions of the theorems; indeed these conditions might be referrered to as ”no self-fulfilling prophecy” conditions. We shall discuss below the relevance of different methods of exploration, but for now note only that a sufficiently expressive Q-table and *some* exploration is *sufficient* to avoid non-stationarity - it isn't clearly the case that lengthier or more complex exploration-rules will guarantee convergence if the Q-table is still restricted.

While we have shown that ”expressiveness” of the agents' learned Q-tables must be sufficient in order to learn correctly, it remains unclear what this would look like in this discrete setting. We have seen that the agents must learn based on action tuples, not just their own actions; while a tabular learner can learn a Q-table this way with no difficulty, this table *must be indexed by information to which a firm does not have access* (i.e., the other firms' actions) - thus there is no natural analogue of the argmax-policy (modelling other agents is necessary in this case, but the difficulties discussed in 1.2 maintain). While experiment two extends this experiment into a more complicated, continuous bid-space, this is essentially a technical achievement; on the other hand, the separation of the learning algorithm into ”actor” and ”critic” allows an effective compromise-position on what information is available, as discussed in 3.2. We will elaborate on the consequences of this below.

In sum, the discrete-environment with tabular learners clearly exhibits pathology due to non-stationarity, preventing the realistic simulation of strategic behavior; further, there is not a natural way to alter the tabular-learning algorithm which resolves this problem. Other algorithms exist which *do* address this problem, but these are beyond our scope here and to the extent they succeed, they do so for the same reasons which we will outline below.

5.1.2. Continuous Environment Experiment Results: Bertrand Competition

The relevant data are summarized in figure 4.10.

The purpose of this first case is to perform experiments as closely analogous as possible to those performed in the tabular environment. We reiterate that the critic-loss in the continuous-domain is a close analogue to the TD-error in the tabular case - they may be interpreted as measures of a firm's degree of ”surprise” upon receiving some outcome.

As expected, we see that the DDPG agent is capable of learning monopolist behavior, and of competing to equilibrium with another *non-learning* agent - as illustrated by the low critic-loss and deficit. This provides evidence that the failure to find a best-response in the multi-learner cases indicates a failure due to non-stationarity.

We then examine the case of competition between two agents - in the first case, using DDPG, and in the second, using MADDPG. As DDPG is a close analogue of the tabular Q-learning discussed above, we see, as expected, that it does not learn to play a best-response against itself. The deficit remains large, while the critic-loss is also large; in the tabular case, we would have expected this to go to zero for a ”self-fulfilling prophecy”, but in the case of the neural network, this critic-loss decays more slowly due to the usage of a memory-buffer - i.e., the agent learns from the last handful of episodes, rather than the last merely. Nonetheless, MADDPG exhibits a lower critic-loss and agent-averaged deficit (these are not zero, though a look at the appropriate plots in the appendix will show them to be making a perhaps more convincing approach than those for DDPG).

As discussed above, the reason that MADDPG succeeds where DDPG fails is that the Q-function is more expressive - it has enough dimensionality to satisfy the theorems in 2.3. The reason this solution is acceptable here where it was not in the discrete-case is that, while the Q-function is given access to hidden-information, *the policy trained from it is not*. For the practical purposes of market-design, this means that firms will not behave as if they had access to hidden information, but will still be able to learn responses to opponents' actions reliably.

5.1.3. Continuous Environment Experiment Results: Q,P-Bidding

The relevant data are summarized in Figure 4.15.

The purpose of this second case is to extend the previous environment slightly. Agents being allowed to bid in (Q, P) pairs, the range of strategic behavior becomes considerably more complex, especially as the firms no longer maintain sufficient capacity to unilaterally meet demand. Further, note that, as the analytical solution to the problem becomes considerably more complex, we provide results in terms of agent-averaged profits, rather than deficit from the optimal profit.

We observe that in the first case, the DDPG agent learns to make a small profit - this is to be expected, in that the naive opponent is happy to meet most of demand at-cost, leaving only a few

tens of MW of demand, which, however, the learner is free to set the price for (within demand-constraints). Likewise, the critic-loss goes to zero, as there are no other learning agents, nor is there hidden-information.

By contrast, we compare again the DDPG and MADDPG learners competing. Clearly, the MADDPG learners do much better with respect to critic-loss - as expected, they are much less regularly surprised than their DDPG counterparts. Interestingly, the MADDPG agents attain less average-profits than the DDPG learners. That is, were a market-designer evaluating this market for strategic behavior using DDPG or other independent-learners, *they would be substantially misestmating a key design metric*. Instead, when the MADDPG algorithm is used, we see that firms make much less overall profit (with high variation between runs; note that the quantities shown are averaged over agents in all cases, and the uncertainty bars in Figure 4.15 correspond to the standard deviation between runs). Correspondingly, the market-designer obtains more accurate information about the design.

5.1.4. Notes on the Tabular and the Continuous Environment Results

We offer here a few remarks pertaining to the results obtained in the first two experiments, as they are of a similar character.

First we note that the results obtained all used a particular exploration-rule, which could readily be substituted for others. While these rules were not exactly analogous between the two experiments (uniform-random in the tabular case, gaussian input-noise in the continuous), the essential non-stationarity problem remains the same. The convergence theorems proved do permit a relatively large role for the exploration rule, and no doubt, some might cause learners to be less prone to the non-stationarity problems illustrated above than others, especially if they taper gradually in effect rather than turniong off after a set time. We do not claim that the simple exploration protocol used was the most effective for mitigating non-stationarity, merely that the switch to a sufficiently expressive/informed Q-function is sufficient to solve the problem.

A similar remark pertains to the neural network technicalia as specified in 3.3.1, and indeed for the use of DDPG and MADDPG in general, of which the latter is more interesting to discuss, the former being the subject of engineering that every project must complete for itself regardless. DDPG and MADDPG were selected as algorithms because of their close relationship to the tabular Q-learning method, which allows for a conceptually clearer comparison of the two experiments' outcomes. The key difference between DDPG and MADDPG is the expressiveness of the Q-function, and in turn the access (during training only) to normally hidden-information - DDPG does not satisfy the convergence theorems proved, while MADDPG does. We expect a similar distinction to hold between any pair of algorithms which are differentiated by the information given to the critic. On the other hand, it is worth noting that the independence of the policy of hidden-information is a design-desideratum, not a theoretical constraint - we do not think agents will arrive at realistic strategic behavior if they are given access to hidden-information such as an opponent's bidding strategy.

5.1.5. Adversarial Market Design Experiment Results

The relevant data are summarized in Figure 4.20.

The qualitative analysis of the price-cap setting problem presented in 2.4 will guide our discussion here. Overall, we can see that the final price-cap set can be unreasonably high, while the actually achieved clearing-prices are more reasonable. This may be explained by noting that the high price-cap values are observed for very low adversarialities, and these tend not to be reflected in the clearing price - that is, when the market is design to favor producing-firms, the market designer simply steps out of the way and allows the oligopoly outcome to occur. This being the case, most of our discussion will focus on observations relating to the price-cap, which more clearly illustrates the outcomes, even as it is not the design-variable.

We also observe a marked spread in the final clearing-prices, even between runs with the same parameters. A look at the related figures in the appendix will confirm that the training-process is quite erratic, and did not always clearly arrive at an unambiguous final result. This would seem to reflect that the adversarial market-design process is much harder to train for than simple strategic behavior - though this is reasonable to expect if one compares to the close alternative of setting the adversary as an outer-loop optimizer, whose single-timestep would correspond to a full retraining of the strategic agents. Nonetheless, this seems to be a limitation of the method as applied in this simple form (c.f. remarks below on limitations and future work).

That said, the parameter-sweep does yield some information: recall that we predicted from the analytical solution that the price-cap should be equal to the marginal cost for adversariality $\alpha > 1/2$, and should interpolate to being the oligopoly price (or higher, as the outcomes are the same) as $alpha$ goes to zero (as producers are more favored). Indeed, we find that, when the adversariality is one - i.e. when the market-designer is totally unconcerned with producer welfare, the price cap is consistently set to be approximately the producers' marginal costs, as expected. By comparison, as the level of adversariality decreases, we see that the clearing prices tend to increase - though, compared to predictions, these cases do not all show a clear trend of convergence to the oligopoly outcome.

In sum, we see that adversarial market design has a moderate ability to find the optimal outcome as intended, though this is strongly restrained by the reliability of the training method. We will discuss this problem and concomitant recommendations below.

5.2. Recapitulation

We reiterate the core research question we wish to answer in this work: "**under what conditions can MARL methods be productively used to simulate rational agents' behaviour for the purposes of electricity-market design?**"

We are now in a position to outline how our theoretical work and experiments answer this question. In particular, the argument may be summarized as follows:

- **Claim**: Provided the TD-error / Critic-loss (and concomitant change in policy) vanish, an agent is playing a best-response given its opponents' policies if and only if the optimality-deficit is zero. When agents' TD-errors / critic-losses vanish and the optimality-deficit does not, the agents have arrived at an incorrect equilibrium due to non-stationarity.
  Evidence: The first sentence is true by design, as given in 2.2. The second sentence is implied by contradiction of convergence theorems for single-agent Q-learning (c.f., e.g., [23]).
- **Claim**: Independent-learning MARL methods are not, in general, capable of simulating realistic strategic behavior in markets (in the sense of arriving at Nash-equilibria).
  Evidence: Such failure is implied by non-zero optimality deficits in the tabular and in the continuous, Bertrand competition experiments, in those situations with multiple competing agents (except with MADDPG, which is not an independent-learning algorithm); equivalently, the firms do not learn to play best-responses given each others' policies.
- **Claim**: For value-learning methods, the non-stationarity problem may be solved if the algorithm meets the constraints imposed by the theorems in 2.3.
  Evidence: see proofs in 2.3; the superior performance (i.e. lower optimality-deficit) of MADDPG over DDPG outcomes in the continuous environments is also evidence for this claim.
- **Claim**: Provided that learners are known to converge to correct Nash equilibria, they may be used to infer the possible severity of strategic-behavior in a given market-design, even when the Nash equilibria are not known in advance.
  Evidence: the continuous Q,P-Bidding experiments show MADDPG agents to earn less profit (a proxy for market vulnerability to strategic-behavior) than DDPG agents; previous claims support the assertion that the MADDPG outcome is more likely to be correct than the DDPG one.
- **Claim**: Insofar as learners reliably learn correct strategic-behavior within a market-design, this permits the use of adversarial market-design - the optimization of market-design constrained by the optimization of agents' strategic behavior.
  Evidence: see the adversarial market design experiments discussed above; this claim is also supported by the theorems proven in 2.3 when the adversary is taken to be one of the agents.

The first three of these claims tell us when agents may be taken as approximating rational behavior. The final two relate these constraints to the use of RL simulations for practical market-design.

5.3. Limitations and Directions for Future Work

We close with remarks highlighting the range of applicability of the present work, places where it is limited in application, and how it might be extended in future work.

The present work has several shortcomings regarding the simplicity of the models used. First, the simple day-ahead market simulated is clearly not of meaningful interest to market-designers. The key

class of problems for which RL simulation is useful is when complexity of the market is high enough that optimization and analytical methods become intractable. Future work might apply the methodology used here to much more complex cases of coupled markets, markets with multiple timesteps, or cases where actors control multiple assets or have the ability to submit bid-curves rather than bid-pairs - the key is that while optimization methods may become impractical, RL methods remain applicable provided the problem can still be framed as a muti-agent MDP. It is hoped that the present work provides the methodological foundation for such extensions.

Further, it is unrealistic to assume that agents are totally unaware of each others' actions and of the outcome of a bidding-round; likewise it is unreasonable to assume that agents' generator capacities are really hidden-information. In the former case, access to more information pertaining to the outcome of previous bids would clearly be more realistic; thus future work might focus on ways to productively use such data as input to the agents (though it should be noted that this extension is only justified for the case of problems with multiple timesteps - it is not in the case of the single-timestep environments considered here, insofar as the problem by the Markovian assumption cannot depend at all on such data). Similarly, if one is interested in strategic behavior, it would perhaps behoove a market-designer to investigate the role of possible communication among agents - i.e. the explicit formation of cartels; market models might be extended to include such considerations.

There are likewise a number of constraints faced by the simplicity of the training protocols of the current work; for one, the epsilon-exploration used in the tabular-learning section is certainly among the more apt to cause non-stationarity problems, so future work might investigate if any exploration protocols might provably save independent Q-learning or otherwise render the process more efficient. Another glaring constraint of the section on adversarial market design is the clear difficulty of obtaining converged results - here work might investigate different RL setups (learning algorithms, exploration rules, etc.) which might stabilize this procedure and thus render it more practically useful.

Similarly, the adversarial market-design procedure is fundamentally limited by its need to handle a parameterized family of market designs (an extension to discrete choices is in principle possible, but would seem to not play to DeepRL's strengths except perhaps as an ad hoc method of combinatorial optimization over many discrete design choices). Though this is a serious limitation, future work might productively apply it to situations complex enough to be of practical utility to designers e.g., a demand-schedule for carbon-credit reserves.

References

[1] Yujian Ye et al. “Deep Reinforcement Learning for Strategic Bidding in Electricity Markets”. en. In: *IEEE Trans. Smart Grid* 11.2 (Mar. 2020), pp. 1343–1355. ISSN: 1949-3053, 1949-3061. DOI: 10.1109/TSG.2019.2936142. URL: https://ieeexplore.ieee.org/document/8805177/ (visited on 02/07/2023).

[2] Thomas Wolgast, Eric MSP Veith, and Astrid Nieße. “Towards reinforcement learning for vulnerability analysis in power-economic systems”. en. In: *Energy Inform* 4.S3 (Sept. 2021), p. 21. ISSN: 2520-8942. DOI: 10.1186/s42162-021-00181-5. URL: https://energyinformatics.springeropen.com/articles/10.1186/s42162-021-00181-5 (visited on 12/02/2022).

[3] Christoph Graf et al. “Computational Performance of Deep Reinforcement Learning to Find Nash Equilibria”. en. In: *Comput Econ* (Jan. 2023). ISSN: 1572-9974. DOI: 10.1007/s10614-022-10351-6. URL: https://doi.org/10.1007/s10614-022-10351-6 (visited on 04/21/2023).

[4] Ksenia Poplavskaya, Jesus Lago, and Laurens de Vries. “Effect of market design on strategic bidding behavior: Model-based analysis of European electricity balancing markets”. en. In: *Applied Energy* 270 (July 2020), p. 115130. ISSN: 03062619. DOI: 10.1016/j.apenergy.2020.115130. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306261920306425 (visited on 11/23/2022).

[5] Daniel S. Kirschen and Goran Strbac. *Fundamentals of Power System Economics*. New York, UNITED KINGDOM: John Wiley & Sons, Incorporated, 2004. ISBN: 978-0-470-02058-6. URL: http://ebookcentral.proquest.com/lib/delft/detail.action?docID=219775 (visited on 02/14/2023).

[6] Peter Cramton. “Electricity market design”. en. In: *Oxford Review of Economic Policy* 33.4 (Nov. 2017), pp. 589–612. ISSN: 0266-903X, 1460-2121. DOI: 10.1093/oxrep/grx041. URL: http://academic.oup.com/oxrep/article/33/4/589/4587939 (visited on 02/07/2023).

[7] Lion Hirth. “The market value of variable renewables”. en. In: *Energy Economics* 38 (July 2013), pp. 218–236. ISSN: 01409883. DOI: 10.1016/j.eneco.2013.02.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0140988313000285 (visited on 02/14/2023).

[8] Lawrence M. Ausubel and Peter Cramton. “Using forward markets to improve electricity market design”. en. In: *Utilities Policy* 18.4 (Dec. 2010), pp. 195–200. ISSN: 09571787. DOI: 10.1016/j.jup.2010.05.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957178710000366 (visited on 02/07/2023).

[9] Steven A. Gabriel et al. *Complementarity Modeling in Energy Markets*. en. Google-Books-ID: Lu1L5wUea8IC. Springer Science & Business Media, July 2012. ISBN: 978-1-4419-6123-5.

[10] Marc Lanctot et al. “A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/hash/3323fe11e9595c09af38fe67567a9394-Abstract.html (visited on 06/03/2023).

[11] Lucian Busoniu, Robert Babuska, and Bart De Schutter. “A Comprehensive Survey of Multiagent Reinforcement Learning”. en. In: *IEEE Trans. Syst., Man, Cybern. C* 38.2 (Mar. 2008), pp. 156–172. ISSN: 1094-6977, 1558-2442. DOI: 10.1109/TSMCC.2007.913919. URL: https://ieeexplore.ieee.org/document/4445757/ (visited on 12/22/2022).

[12] Georgios Papoudakis et al. *Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning*. en. arXiv:1906.04737 [cs, stat]. June 2019. URL: http://arxiv.org/abs/1906.04737 (visited on 11/23/2022).

[13] Paul Weirich. “Causal Decision Theory”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2020. Metaphysics Research Lab, Stanford University, 2020. URL: https://plato.stanford.edu/archives/win2020/entries/decision-causal/ (visited on 07/18/2023).

[14] Richard Everett. “Learning Against Non-Stationary Agents with Opponent Modelling & Deep Reinforcement Learning”. en. In: (), p. 8.

[15] *Infra-Bayesianism - AI Alignment Forum*. en. URL: https://www.alignmentforum.org/s/CmrW8fCmSLK7E25sa (visited on 07/18/2023).

[16] Ryan Lowe et al. *Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments*. en. arXiv:1706.02275 [cs]. Mar. 2020. URL: http://arxiv.org/abs/1706.02275 (visited on 11/23/2022).

[17] Julien Perolat et al. *Mastering the Game of Stratego with Model-Free Multiagent Reinforcement Learning*. en. arXiv:2206.15378 [cs]. June 2022. URL: http://arxiv.org/abs/2206.15378 (visited on 11/29/2022).

[18] Ido Erev et al. “Learning and equilibrium as useful approximations: Accuracy of prediction on randomly selected constant sum games”. en. In: *Economic Theory* 33.1 (July 2007), pp. 29–51. ISSN: 0938-2259, 1432-0479. DOI: 10.1007/s00199-007-0214-y. URL: http://link.springer.com/10.1007/s00199-007-0214-y (visited on 02/14/2023).

[19] Ziqing Zhu et al. “Reinforcement learning in deregulated energy market: A comprehensive review”. en. In: *Applied Energy* 329 (Jan. 2023), p. 120212. ISSN: 03062619. DOI: 10.1016/j.apenergy.2022.120212. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306261922014696 (visited on 12/01/2022).

[20] Yan Du et al. “Approximating Nash Equilibrium in Day-ahead Electricity Market Bidding with Multi-agent Deep Reinforcement Learning”. en. In: *Journal of Modern Power Systems and Clean Energy* 9.3 (2021), pp. 534–544. ISSN: 2196-5625. DOI: 10.35833/MPCE.2020.000502. URL: https://ieeexplore.ieee.org/document/9406572 (visited on 02/07/2023).

[21] Katie Steele and H. Orri Stefánsson. “Decision Theory”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2020. Metaphysics Research Lab, Stanford University, 2020. URL: https://plato.stanford.edu/archives/win2020/entries/decision-theory/ (visited on 07/18/2023).

[22] Yoav Shoham. “Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations”. en. In: ().

[23] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. en. Second edition. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018. ISBN: 978-0-262-03924-6.

[24] Richard S Sutton et al. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. en. In: (), p. 7.

[25] Sven Ove Hansson. “Risk”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta and Uri Nodelman. Summer 2023. Metaphysics Research Lab, Stanford University, 2023. URL: https://plato.stanford.edu/archives/sum2023/entries/risk/ (visited on 07/18/2023).

[26] Hugo Sonnenschein. “Market Excess Demand Functions”. In: *Econometrica* 40.3 (May 1972), p. 549. ISSN: 00129682. DOI: 10.2307/1913184. URL: https://www.jstor.org/stable/1913184?origin=crossref (visited on 07/18/2023).

[27] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. en. SpringerBriefs in Intelligent Systems. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-28927-4 978-3-319-28929-8. DOI: 10.1007/978-3-319-28929-8. URL: http://link.springer.com/10.1007/978-3-319-28929-8 (visited on 11/29/2022).

[28] Tabish Rashid et al. “Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 10199–10210. URL: https://proceedings.neurips.cc/paper/2020/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html (visited on 07/18/2023).

[29] Wendelin Boehmer, Vitaly Kurin, and Shimon Whiteson. “Deep Coordination Graphs”. en. In: *Proceedings of the 37th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, Nov. 2020, pp. 980–991. URL: https://proceedings.mlr.press/v119/boehmer20a.html (visited on 07/18/2023).

[30] Kyunghwan Son et al. “QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning”. en. In: *Proceedings of the 36th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, May 2019, pp. 5887–5896. URL: https://proceedings.mlr.press/v97/son19a.html (visited on 07/18/2023).

[31] Drew Fudenberg and David K. Levine. “Learning and Equilibrium”. en. In: *Annu. Rev. Econ.* 1.1 (Sept. 2009), pp. 385–420. ISSN: 1941-1383, 1941-1391. DOI: 10.1146/annurev.economics.050708.142930. URL: https://www.annualreviews.org/doi/10.1146/annurev.economics.050708.142930 (visited on 12/22/2022).

[32] Oriol Vinyals et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. en. In: *Nature* 575.7782 (Nov. 2019). Number: 7782 Publisher: Nature Publishing Group, pp. 350–354. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1724-z. URL: https://www.nature.com/articles/s41586-019-1724-z (visited on 07/18/2023).

[33] Ermo Wei and Sean Luke. “Lenient Learning in Independent-Learner Stochastic Cooperative Games”. en. In: ().

[34] *Brouwer theorem - Encyclopedia of Mathematics*. URL: https://encyclopediaofmath.org/index.php?title=Brouwer_theorem (visited on 07/18/2023).

[35] Tim Roughgarden. “CS364A: Algorithmic Game Theory Lecture #2: Mechanism Design Basics”. en. In: ().

[36] “Mechanism Design and the Revelation Principle”. en. In: ().

[37] Nicolò Cesa-Bianchi et al. “Boltzmann Exploration Done Right”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/b299ad862b6f12cb57679f0538eca514-Abstract.html (visited on 07/18/2023).

A

Supplementary Data for Numerical Experiments

This appendix contains the plots of all experiments performed which were not discussed as examples in Chapter 4. We organize them according to the environment in which they were performed, and label the relevant parameters which were varied, but otherwise offer no comments.

A.1. Tabular Experiments

Single Agent, Multiple States:

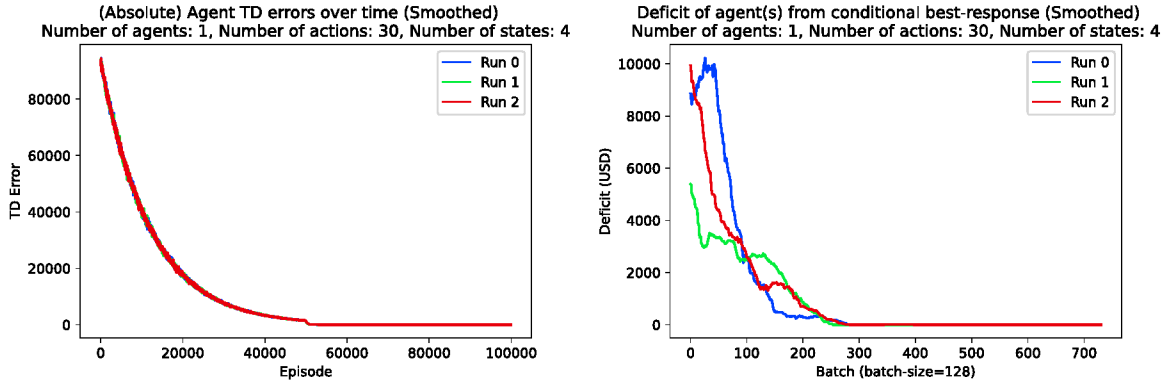


Figure A.1: Plot of the (smoothed) agent TD-error (left) and conditional deficit (right) for the case with one agent - this time with 4 states (corresponding to different generator sizes and marginal costs); as there is only one agent, there is no hidden information.

Three Agents, Stateless: Plotted quantities are agent-averaged; the shaded region represents the standard-deviation amongst agents.

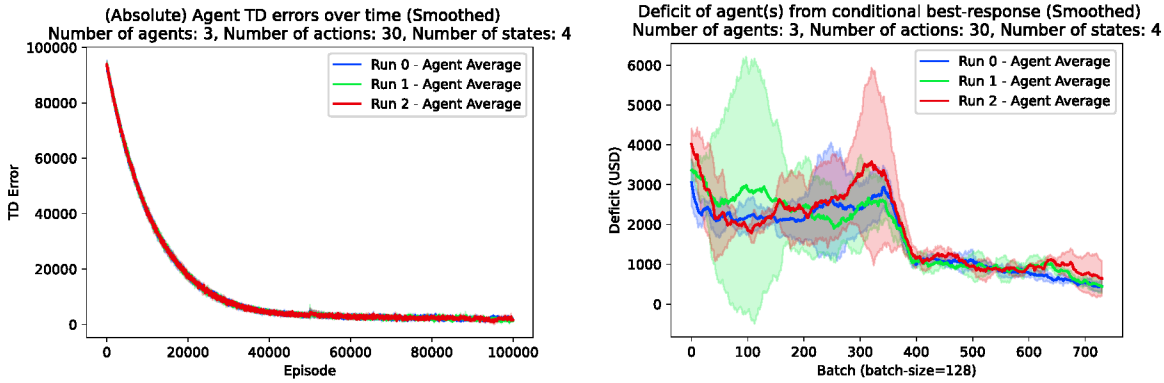


Figure A.2: Plot of the (smoothed) agent TD-error (left) and conditional deficit (right) for the case with three agents and four states (per-agent, corresponding to generator size).

A.2. Continuous Environment

A.2.1. Continuous Environment, Bertrand Competition

The following remarks apply to all plots in this subsection: 'Hypercompetitive' indicates that the agent's generator capacity is greater than the zero-price demand. All plots are shown for each of three independent runs of the simulation. Where multiple learning agents were present, the central curve is the agent-average quantity, while the shaded region is the standard deviation. Exploration was active for half of the total time, while learning occurred throughout.

DDPG Monopolist

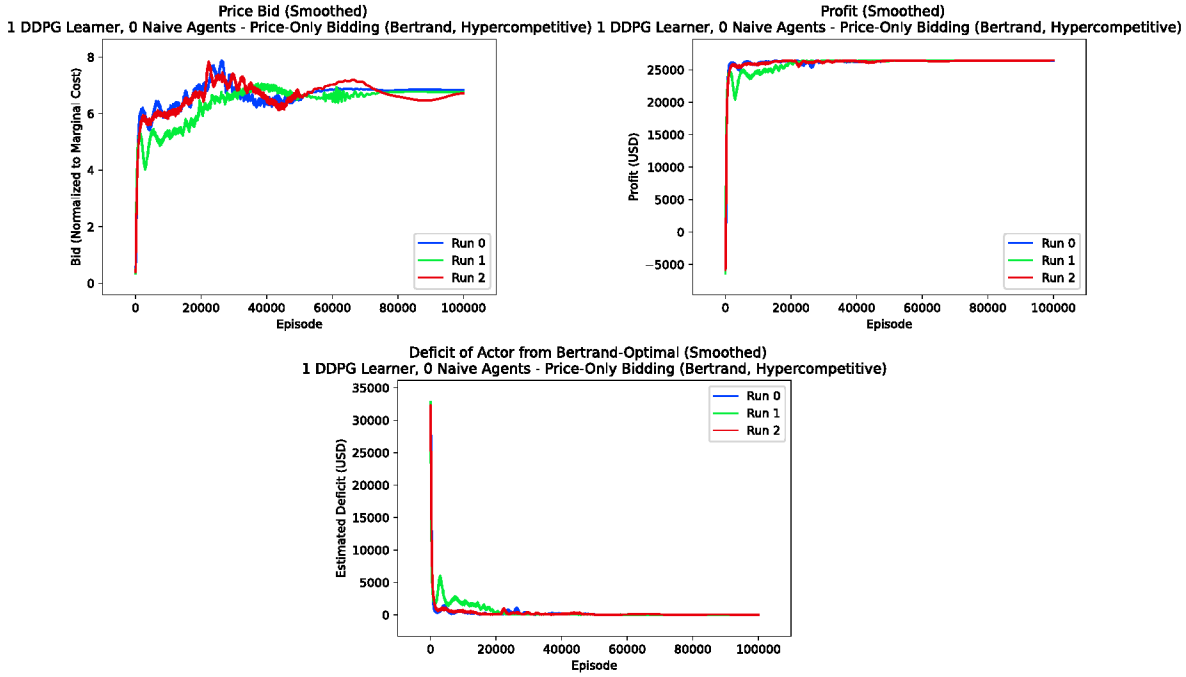


Figure A.3: Bertrand competition: DDPG Monopolist. Plots illustrating the performance of the DDPG algorithm for a simple, price-only market; Top-left: Price bids over training. Top-right: Profit earned. Bottom: Deficit with respect to analytical Bertrand solution.

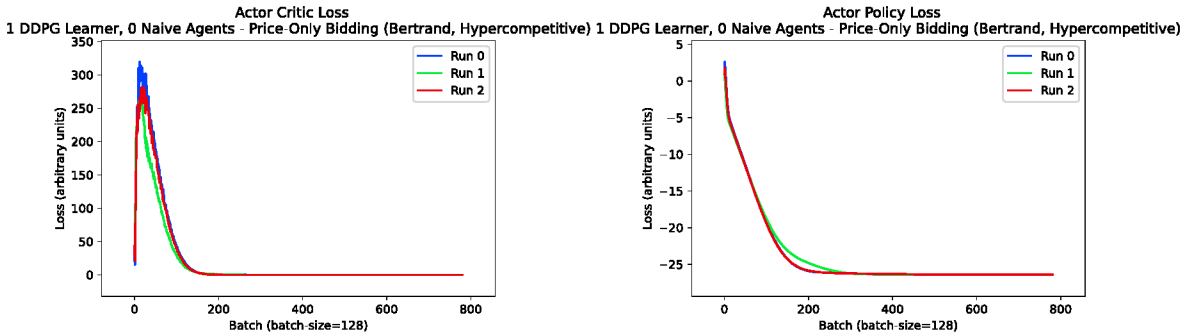


Figure A.4: Plots illustrating the performance of the DDPG algorithm for a simple, price-only market; Left: actor critic loss. Right: actor policy loss.

DDPG vs Marginal Cost Agent

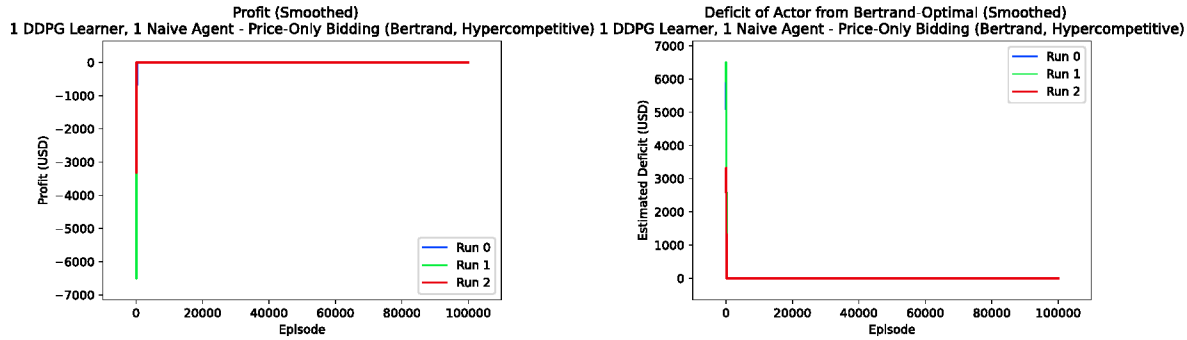


Figure A.5: Plots illustrating the performance of the DDPG algorithm for a simple, price-only market, competing against a single other agent which always bids its marginal cost. Top-left: Price bids over training. Top-right: Profit earned. Bottom: Deficit with respect to analytical Bertrand solution

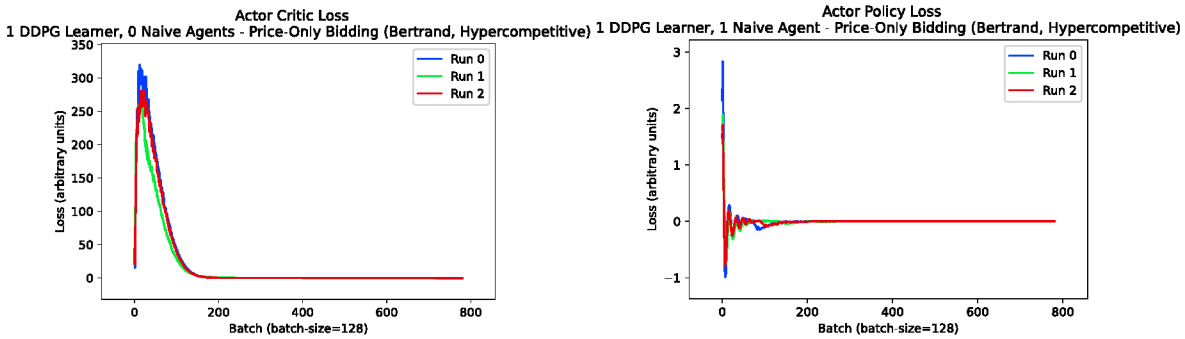


Figure A.6: Plots illustrating the performance of the DDPG algorithm for a simple, price-only market, competing against a single other agent which always bids its marginal cost. Left: Learner critic loss. Right: Learner actor loss.

A.2.2. Continuous Environment, Q,P-bidding

DDPG vs. Marginal Cost Agent

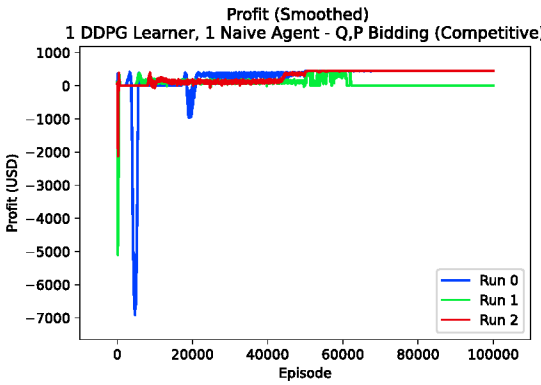


Figure A.7: Plot illustrating the profit of the DDPG algorithm for Q,P-bidding against a single marginal-cost agent.

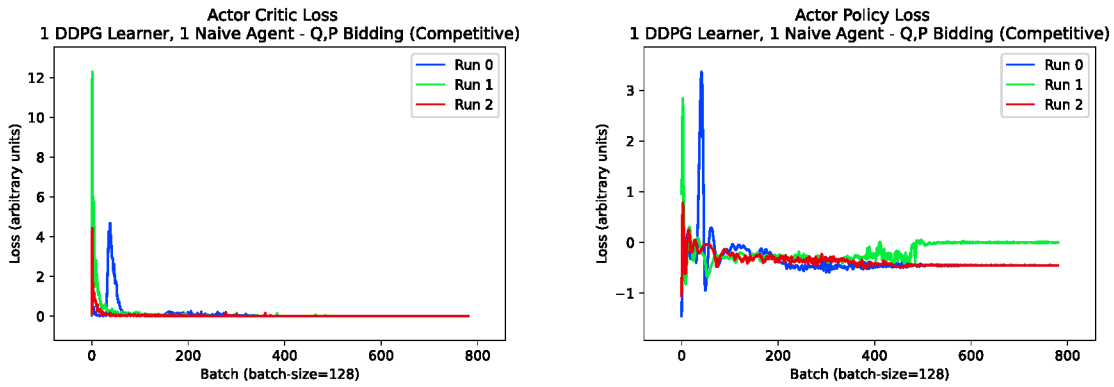


Figure A.8: Plots illustrating the performance of the DDPG algorithm for Q,P-bidding against a single marginal-cost agent. Left: actor critic loss. Right: actor policy loss.

A.3. Adversarial Market Design

The following remarks apply to all plots in this section: Quantities for the firm-agents are averaged over all firm-agents, with the shaded region indicating the standard deviation over agents. Each experiment is plotted for each of three independent runs. The plots for the price and price cap, as well as the social-welfare are smoothed (as noted in the title), while the learning-progress variables are not. In designating the plots, N refers to the number of firm agents, and α to the adversariality.

$N = 1$, $\alpha = 0.00$

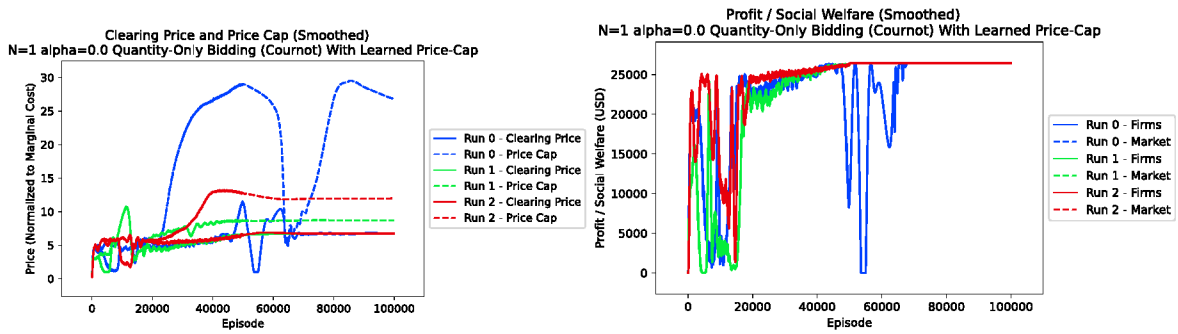


Figure A.9: Plots illustrating the clearing prices and profits for 1 firm agent and the market agent, with adversariality $\alpha = 0.00$. Left: clearing price and price cap. Right: profit and social welfare.

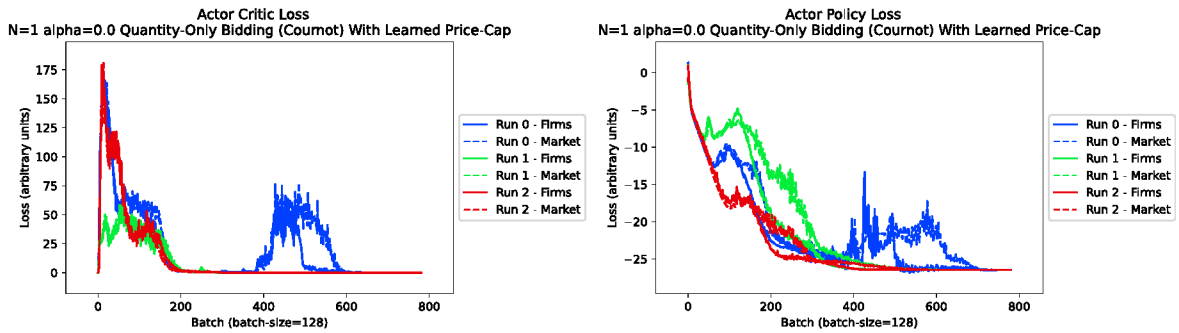


Figure A.10: Plots illustrating the learning progress of 1 firm agent and the market agent, with adversariality $\alpha = 0.00$. Left: critic loss. Right: policy loss

$N = 1$, $\alpha = 0.25$ was included as "example 1" in Section 4.3.
$N = 1$, $\alpha = 0.50$

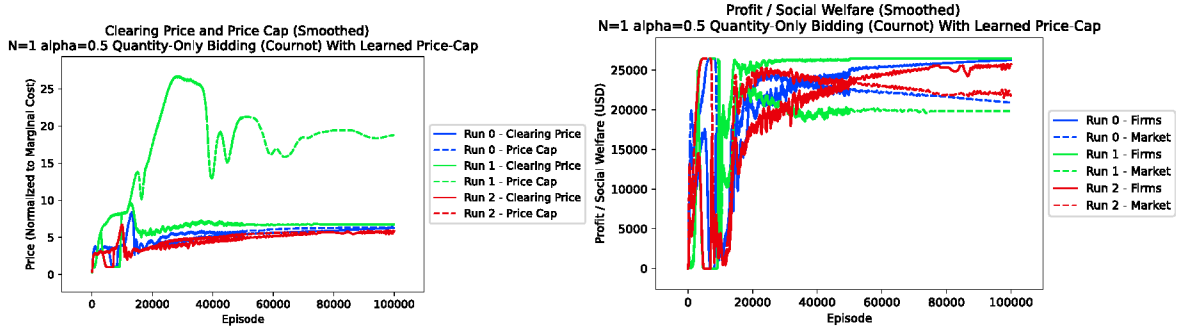


Figure A.11: Plots illustrating the clearing prices and profits for 1 firm agent and the market agent, with adversariality $\alpha = 0.50$. Left: clearing price and price cap. Right: profit and social welfare.

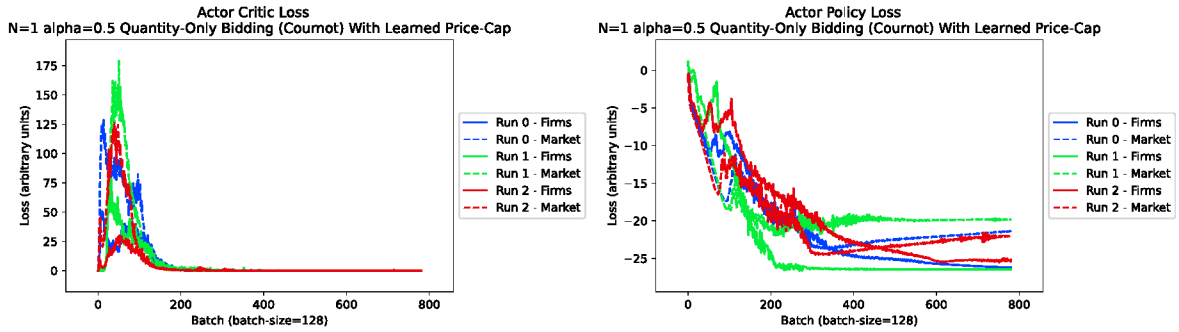


Figure A.12: Plots illustrating the learning progress of 1 firm agent and the market agent, with adversariality $\alpha = 0.50$. Left: critic loss. Right: policy loss

$N = 1$, $\alpha = 0.75$

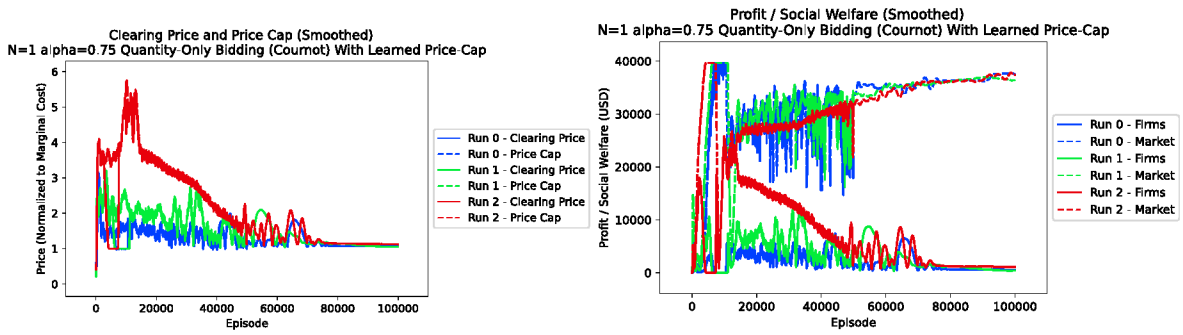


Figure A.13: Plots illustrating the clearing prices and profits for 1 firm agent and the market agent, with adversariality $\alpha = 0.75$. Left: clearing price and price cap. Right: profit and social welfare.

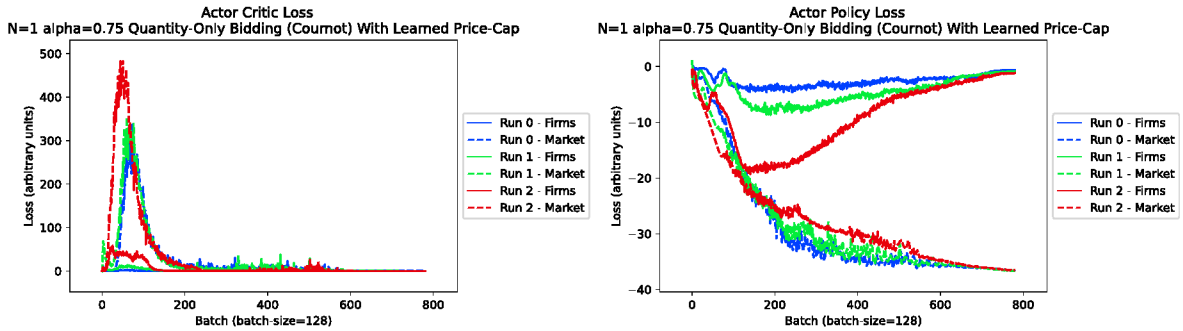


Figure A.14: Plots illustrating the learning progress of 1 firm agent and the market agent, with adversariality $\alpha = 0.75$. Left: critic loss. Right: policy loss

$N = 1, \alpha = 1.00$

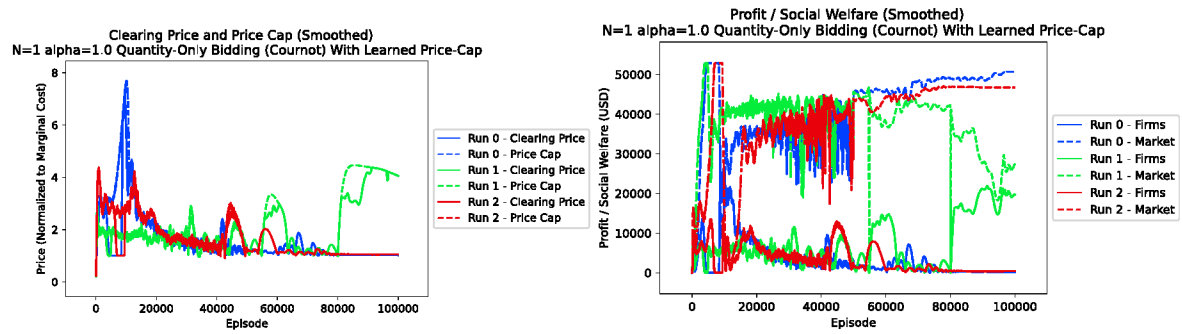


Figure A.15: Plots illustrating the clearing prices and profits for 1 firm agent and the market agent, with adversariality $\alpha = 1.00$. Left: clearing price and price cap. Right: profit and social welfare.

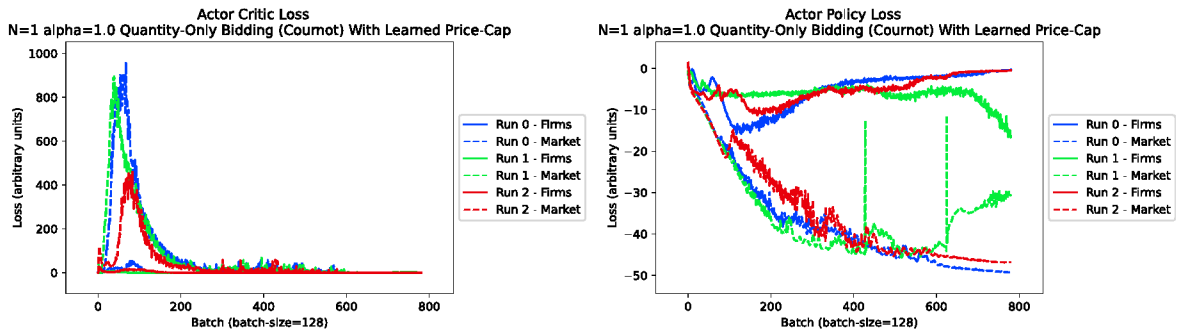


Figure A.16: Plots illustrating the learning progress of 1 firm agent and the market agent, with adversariality $\alpha = 1.00$. Left: critic loss. Right: policy loss

$N = 3, \alpha = 0.00$

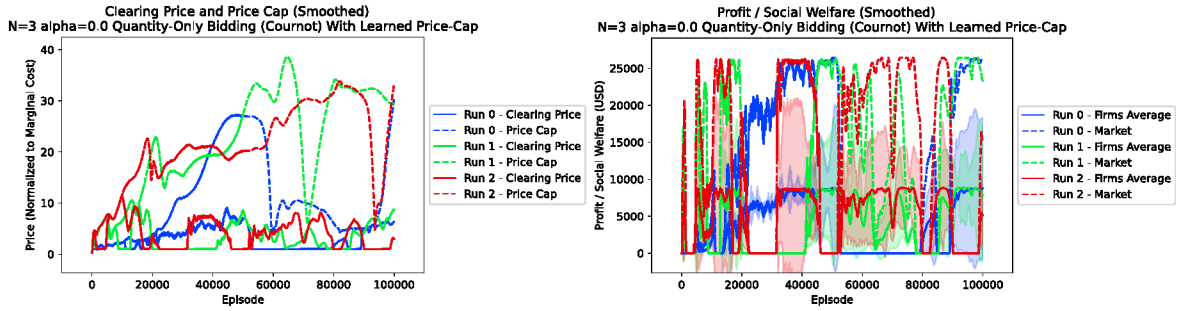


Figure A.17: Plots illustrating the clearing prices and profits for 3 firm agents and the market agent, with adversariality $\alpha = 0.00$. Left: clearing price and price cap. Right: profit and social welfare.

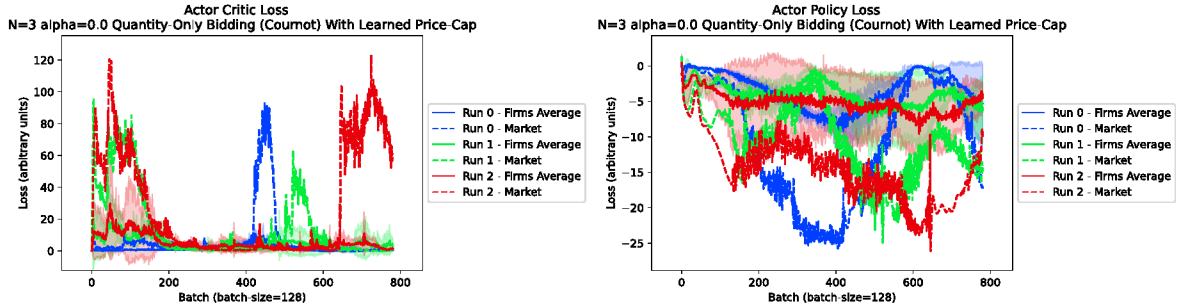


Figure A.18: Plots illustrating the learning progress of 3 firm agents and the market agent, with adversariality $\alpha = 0.00$. Left: critic loss. Right: policy loss

$N = 3, \alpha = 0.25$

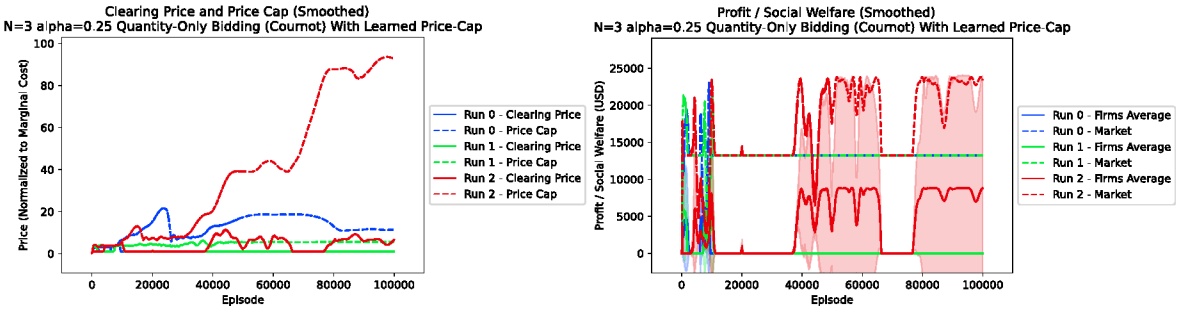


Figure A.19: Plots illustrating the clearing prices and profits for 3 firm agents and the market agent, with adversariality $\alpha = 0.25$. Left: clearing price and price cap. Right: profit and social welfare.

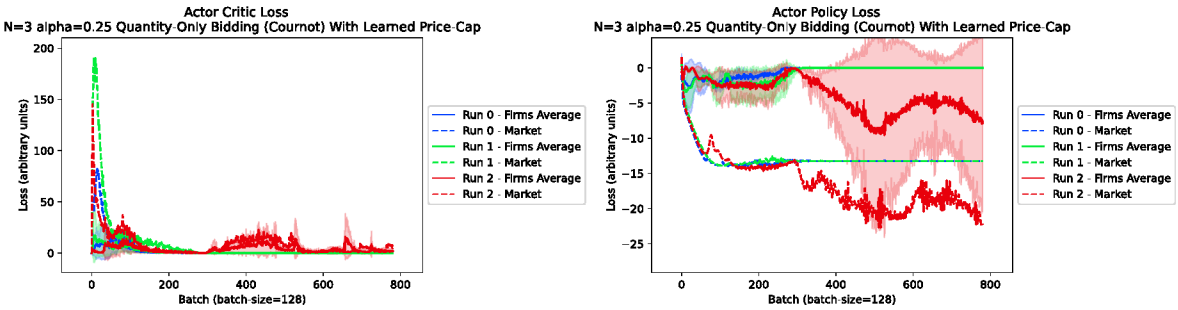


Figure A.20: Plots illustrating the learning progress of 3 firm agents and the market agent, with adversariality $\alpha = 0.25$. Left: critic loss. Right: policy loss

$N = 3, \alpha = 0.50$

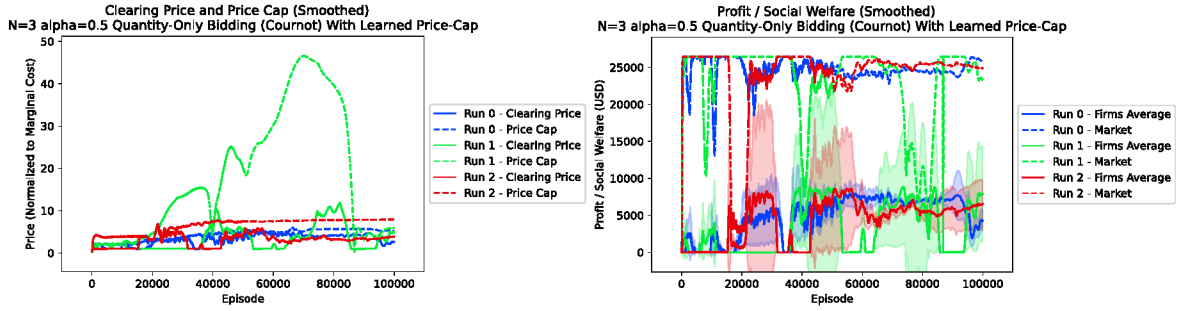


Figure A.21: Plots illustrating the clearing prices and profits for 3 firm agents and the market agent, with adversariality $\alpha = 0.50$. Left: clearing price and price cap. Right: profit and social welfare.

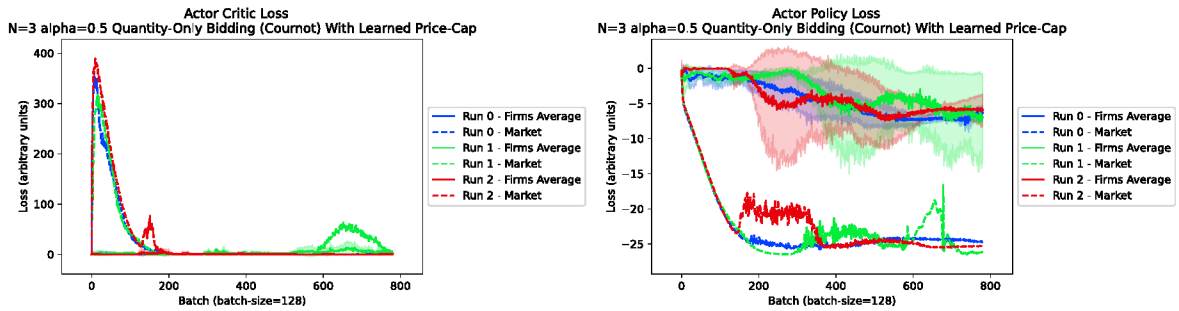


Figure A.22: Plots illustrating the learning progress of 3 firm agents and the market agent, with adversariality $\alpha = 0.50$. Left: critic loss. Right: policy loss

$N = 3$, $\alpha = 0.75$

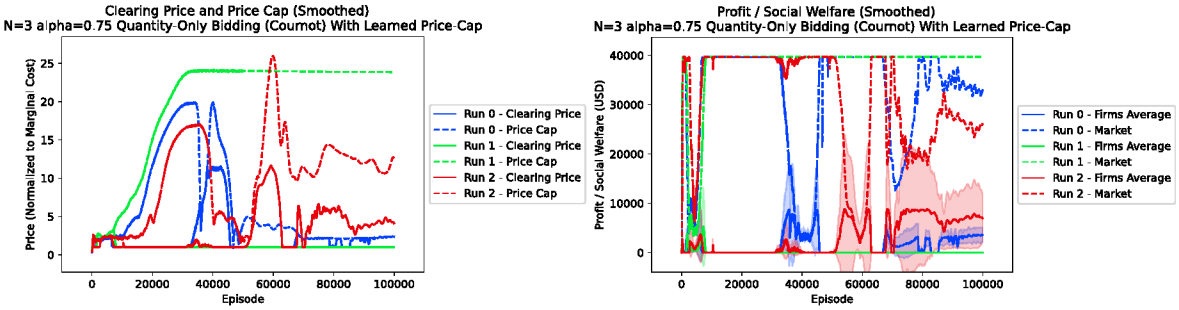


Figure A.23: Plots illustrating the clearing prices and profits for 3 firm agents and the market agent, with adversariality $\alpha = 0.75$. Left: clearing price and price cap. Right: profit and social welfare.

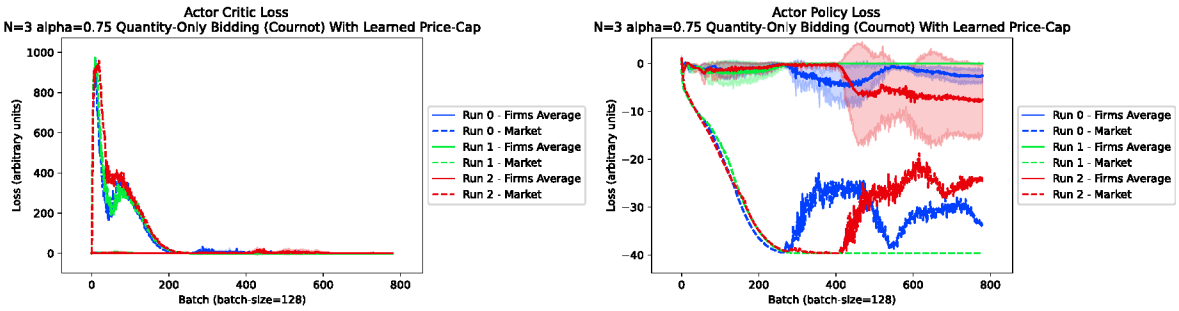


Figure A.24: Plots illustrating the learning progress of 3 firm agents and the market agent, with adversariality $\alpha = 0.75$. Left: critic loss. Right: policy loss

$N = 3$, $\alpha = 1.00$

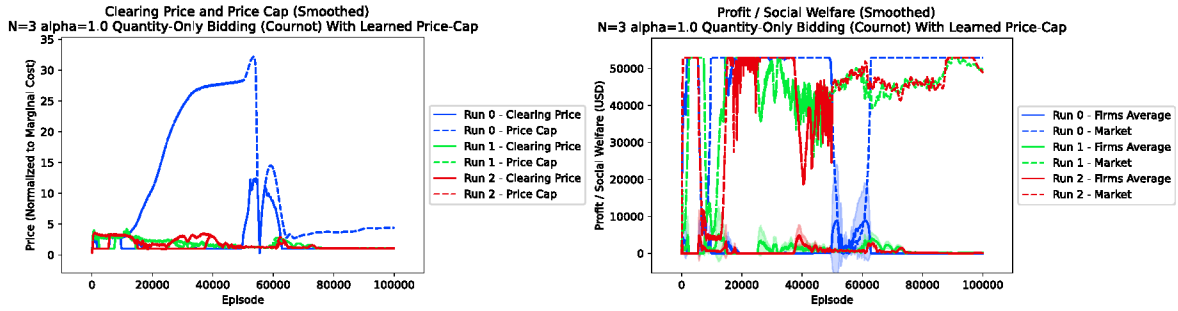


Figure A.25: Plots illustrating the clearing prices and profits for 3 firm agents and the market agent, with adversariality $\alpha = 1.00$. Left: clearing price and price cap. Right: profit and social welfare.

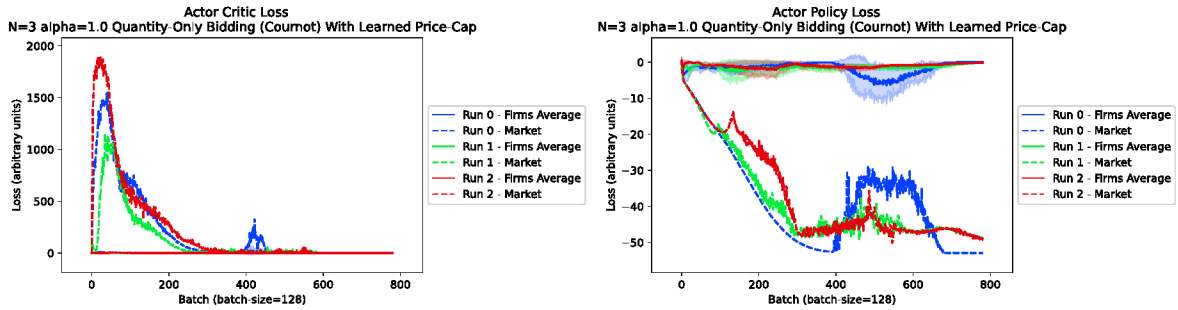


Figure A.26: Plots illustrating the learning progress of 3 firm agents and the market agent, with adversariality $\alpha = 1.00$. Left: critic loss. Right: policy loss

$N = 5, \alpha = 0.00$

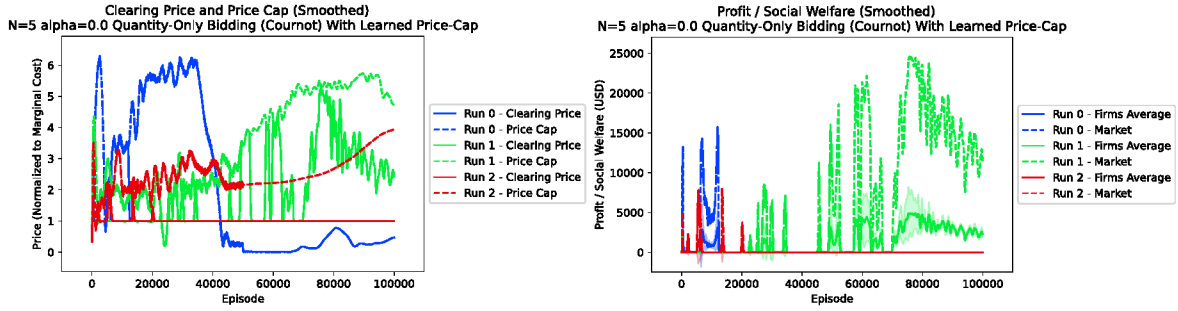


Figure A.27: Plots illustrating the clearing prices and profits for 5 firm agents and the market agent, with adversariality $\alpha = 0.00$. Left: clearing price and price cap. Right: profit and social welfare.

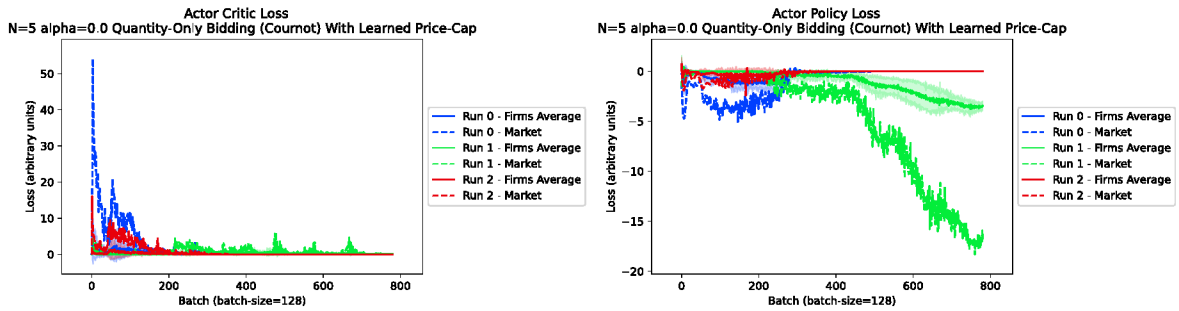


Figure A.28: Plots illustrating the learning progress of 5 firm agents and the market agent, with adversariality $\alpha = 0.00$. Left: critic loss. Right: policy loss

$N = 5$, $\alpha = 0.25$

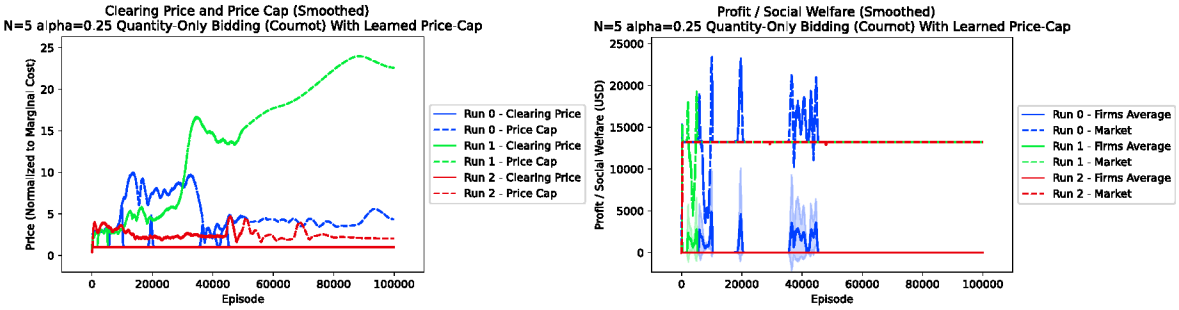


Figure A.29: Plots illustrating the clearing prices and profits for 5 firm agents and the market agent, with adversariality $\alpha = 0.25$. Left: clearing price and price cap. Right: profit and social welfare.

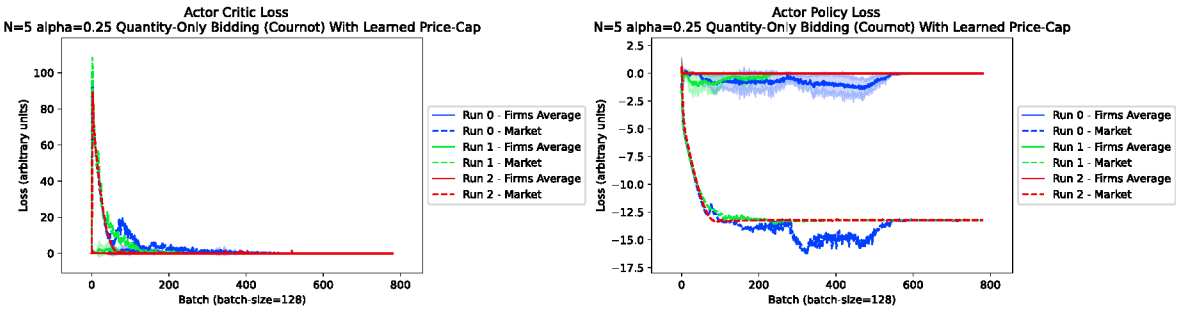


Figure A.30: Plots illustrating the learning progress of 5 firm agents and the market agent, with adversariality $\alpha = 0.25$. Left: critic loss. Right: policy loss

$N = 5$, $\alpha = 0.50$

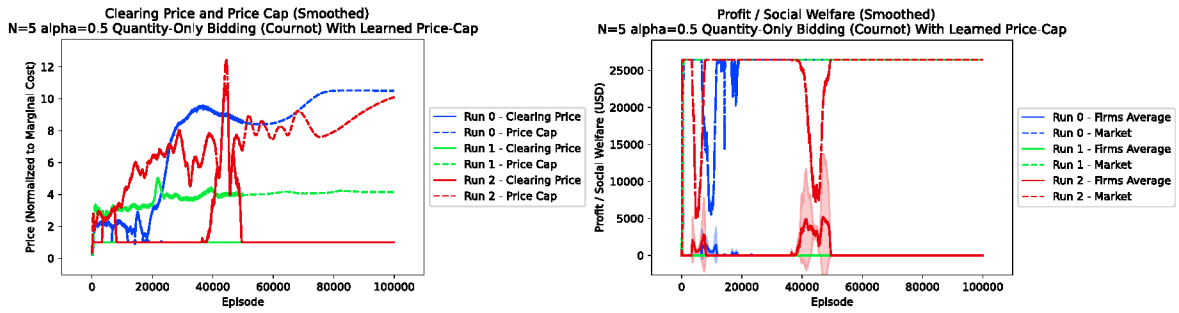


Figure A.31: Plots illustrating the clearing prices and profits for 5 firm agents and the market agent, with adversariality $\alpha = 0.50$. Left: clearing price and price cap. Right: profit and social welfare.

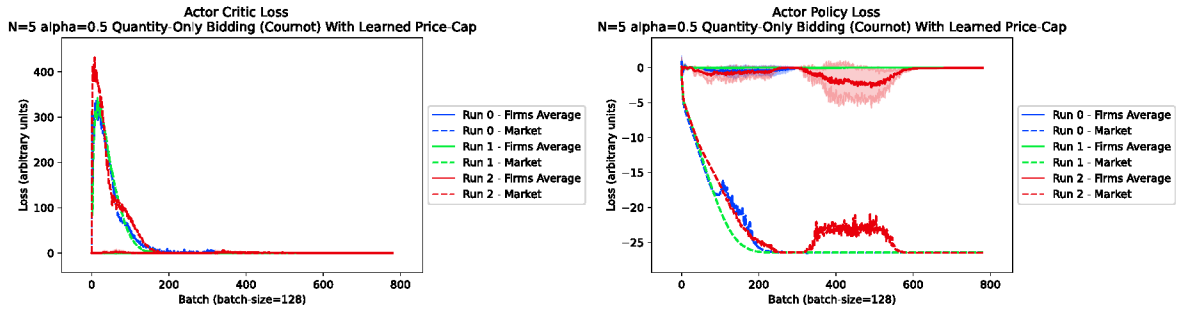


Figure A.32: Plots illustrating the learning progress of 5 firm agents and the market agent, with adversariality $\alpha = 0.50$. Left: critic loss. Right: policy loss

$N = 5$, $\alpha = 0.75$ was included as "example 2" in Section 4.3
$N = 5$, $\alpha = 1.00$

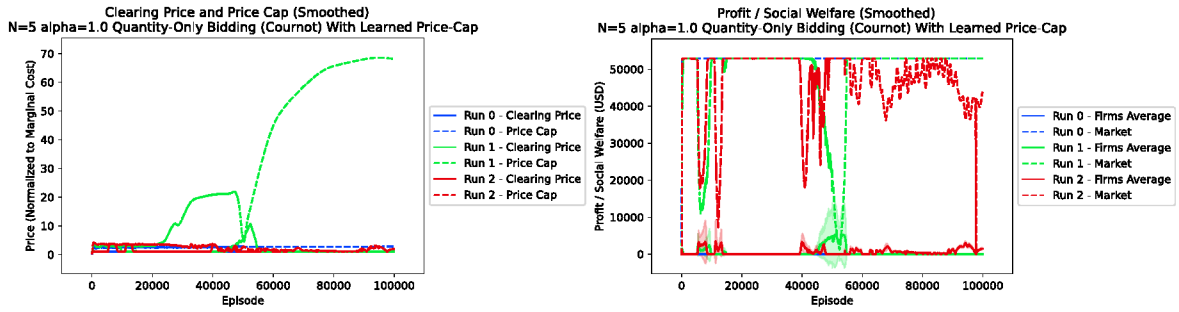


Figure A.33: Plots illustrating the clearing prices and profits for 5 firm agents and the market agent, with adversariality $\alpha = 1.00$. Left: clearing price and price cap. Right: profit and social welfare..

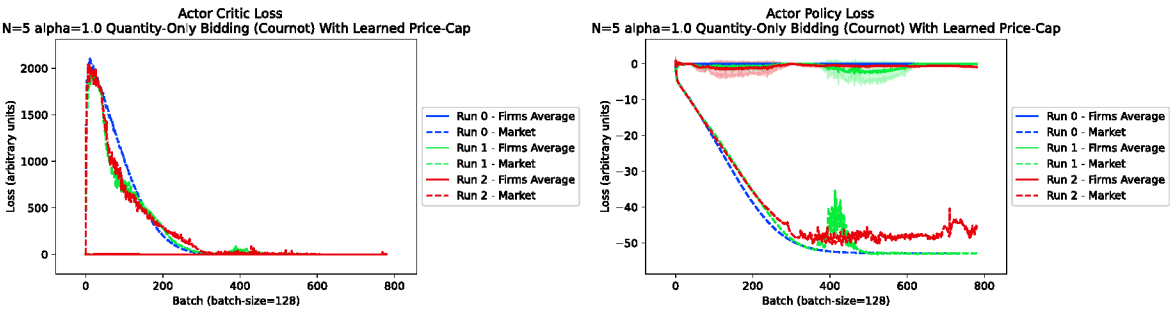


Figure A.34: Plots illustrating the learning progress of 5 firm agents and the market agent, with adversariality $\alpha = 1.00$. Left: critic loss. Right: policy loss