Load Frequency Control: A Deep Multi-Agent Reinforcement Learning Approach

Sergio Rozada, Dimitra Apostolopoulou, and Eduardo Alonso
City, University of London
London, UK EC1V 0HB
Email: {Sergio.Rozada, Dimitra.Apostolopoulou, E.Alonso}@city.ac.uk

***Abstract*—The paradigm shift in energy generation towards microgrid-based architectures is changing the landscape of the energy control structure heavily in distribution systems. More specifically, distributed generation is deployed in the network demanding decentralised control mechanisms to ensure reliable power system operations. In this work, a Multi-Agent Reinforcement Learning approach is proposed to deliver an agent-based solution to implement load frequency control without the need of a centralised authority. Multi-Agent Deep Deterministic Policy Gradient is used to approximate the frequency control at the primary and the secondary levels. Each generation unit is represented as an agent that is modelled by a Recurrent Neural Network. Agents learn the optimal way of acting and interacting with the environment to maximise their long term performance and to balance generation and load, thus restoring frequency. In this paper we prove using three test systems, with two, four and eight generators, that our Multi-Agent Reinforcement Learning approach can efficiently be used to perform frequency control in a decentralised way.**

***Index Terms*—Reinforcement Learning, MADDPG, Droop Control, ACG, Load Frequency Control.**

I. Introduction

The power systems paradigm is shifting from large electromechanical generators driven by heat engines towards distributed generation systems. In this landscape, new challenges should be addressed and tackled, crucially how generators coordinate to maintain generation and load balanced [2]. This objective is met through a hierarchical control system: i.e., primary, secondary and tertiary frequency control. The last two control systems are usually based on centralised methodologies. However, the dynamics of power electronics systems are far too different from traditional systems. When dealing with power electronics systems, the problem of solving this frequency control in a central way is infeasible [3]. In addition, the deployment of distributed renewable generation, e.g., photovoltaic (PV) rooftops, requires new decentralised control designs that ensure frequency restoration in an efficient way.

Several decentralised control strategies have been proposed to efficiently decide on the generation set points to meet the load [4]. The most common approach is to attempt to implement the traditional hierarchical control in a decentralised manner (see, e.g., [5], [6]).

In this paper we investigate Multi-Agent Reinforcement Learning (MARL) as computational models for decentralised load frequency control, and show that they offer an implementable solution to this problem. MARL is an area of Machine Learning that plays with the idea of having different software agents interacting with the world and each other in order to learn optimal policies by negotiating, cooperating, and/or competing [7]. Following this philosophy, the engineering problem has been formulated as a framework where the agents learn to modify the energy supply to keep generation and load balanced only using local information.

In this work: i) we formalise the frequency control problem as a MARL problem; ii) we use Multi-Agent Deep Deterministic Policy Gradient (MADDPG) as an actor-critic architecture that is able to deal with continuous states and actions in a multi-agent problem; and iii) we demonstrate that the proposed framework can solve the control problem in a decentralised way through numerical studies.

The rest of the paper is structured as follows: in Section II the hierarchical frequency control system is formulated. In Section III we define our approach to solve the problem. In Section IV results are presented. We finish with some conclusions in Section V.

II. Preliminairies

The balance of supply and demand can be judged through the system frequency. If the frequency of the system is higher than the nominal set point, the total amount of power generated exceeds the total load. On the other hand, if the frequency of the system is below the nominal set point, more power is needed to supply the demand. Thus, a common approach to keep generation and load balanced is to control the frequency of the system [8]. This control is implemented hierarchically and consists of three different layers: primary, secondary and tertiary control.

The idea behind primary control is to rapidly balance generation and demand. Generators share the load and move the power output in the direction that stabilises the frequency around the nominal set point. This is accomplished through a decentralised proportional control mechanism called droop control [9].

The secondary control layer acts over the primary control layer to compensate the steady state error that cannot be eliminated by a proportional controller. The main solution used to implement the secondary control is an integral control called

Automatic Generation Control (AGC) [10]. In order to work, AGC needs to gather information from the generation units of the system in a central way.

Following the classical model of representation we consider a power system with n generators. We denote by $\Delta\omega$ the deviation of the center of inertia speed from the synchronous speed. We denote the synchronous speed or nominal frequency ω_s; the set of n generators $\mathscr{G} = \{G_1, \ldots, G_n\}$; the total electrical power produced $P_G = \sum_{i \in \mathscr{G}} P_i$, where P_i is the output of generator i; the total load P_L; and the total AGC command $Z_G = \sum_{i \in \mathscr{G}} z_i$, where z_i is the participation of each generator i in AGC. Let σ_i denote the normalized participation factor of bus load changes ΔP_{L_i} with respect to total system load change ΔP_L, and P_{losses} the system losses, then ρ, which denotes the sensitivity of the losses with respect to the system load is defined as [5]:

$$\rho = \sum_{i \in \mathscr{G}} \sigma_i \frac{\partial P_{\text{losses}}}{\partial P_i}. \quad (1)$$

The dynamic behaviour of the system can be expressed with the following equations:

$$M\frac{d\Delta\omega}{dt} = P_G - (1+\rho)P_L - D\Delta\omega, \quad (2)$$

$$T_G\frac{dP_G}{dt} = -P_G + Z_G - \frac{1}{R_D}\Delta\omega, \quad (3)$$

where $M = \frac{2H}{\omega_s}$, with H being the system inertia constant; D is the load damping; R_D is the droop; and T_G is some time constant (see, e.g., [9]).

The tertiary control layer has to do with the economic aspect of power system operations. This layer establishes the load sharing between the sources so that the operational costs are minimised [11]. Tertiary control is implemented through the economic dispatch, which calculates the optimal operating point in an offline process.

III. Reinforcement Learning For Load Frequency Control

In this section we formulate the load frequency control as a MARL problem, determine the reward function and choose the appropriate architecture that deals with continuous state and action spaces and involves the interaction of multiple agents.

A. Formalizing the problem as a Markov Decision Process

MARL is used to propose a fully decentralised solution to implement the frequency control system. Its main focus is how an agent or a collection of agents interact with a given environment in order to optimise their performance. In our approach, each generation unit has been designed as one agent, and the frequency control problem is mathematically formalised as a Markov Decision Process (MDP) [13], which is the framework to parse any context into a structure that MARL techniques can operate on. A MDP is defined as the tuple $\langle S, A, P, R\rangle$, where each term is:

- *S or state space*: the set of all possible states. It defines where the agent is in the world. In the frequency control problem, there are two continuous states: the deviation of the center of inertia speed from the synchronous speed denoted by $\Delta\omega$ and the current control action z_i of each generator i. The first state informs the agent about how large the gap between demand and supply is. The second one gives the agent a notion of how much they are contributing with.
- *A or action space*: the set of all possible actions to take in each state. The actions may be the same for all the states in the state space or each state can have its own actions to take. In the context studied here, the action that each agent can perform is to increase or decrease the control action z_i. This will directly affect the state of the world.
- *P or transition function*: it defines how the environment transits between states. This is a set of equations that are derived from the dynamics of the environment stated in (2) and (3), namely:

$$M\frac{d\Delta\omega_{\text{new}}}{dt} = P_{G_{\text{old}}} - (1+\rho)P_L - D\Delta\omega_{\text{old}}, \quad (4)$$

$$T_G\frac{dP_{G_{\text{new}}}}{dt} = -P_{G_{\text{old}}} + Z_{G_{\text{new}}} - \frac{1}{R_D}\Delta\omega_{\text{old}}, \quad (5)$$

$$Z_{G_{\text{new}}} = \sum_{i \in \mathscr{G}} z_{i_{\text{new}}}, \quad (6)$$

$$z_{i_{\text{new}}} = z_{i_{\text{old}}} + \Delta z_i, \quad (7)$$

$$\Delta\omega_{\text{new}} = \Delta\omega_{\text{old}} + \frac{d\Delta\omega_{\text{new}}}{dt}\Delta t, \quad (8)$$

$$P_{G_{\text{new}}} = P_{G_{\text{old}}} + \frac{dP_{G_{\text{new}}}}{dt}\Delta t, \quad (9)$$

  where Δt is the time ellapsed between steps, the subscripts new and old denote the value in previous and current state and Δz_i is the increase or decrease in power generation by each unit i in $\mathscr{G}$ estimated by each agent. Δz_i is determined by the Multi-Agent Deep Deterministic Policy Gradient, as described in Section III-B. When an agent decides to increase or decrease z_i, this set of equations expresses in which way the environment will change.
- *R or reward function*: it defines a numerical signal or reward expressing how "good" being in a state and performing an action is. The reward function tells the agent to maintain the deviation of the center of inertia speed from synchronous $\Delta\omega$ as close to zero as possible.

The goal of MARL is to find an optimal policy π that maximises the cumulative reward obtained in the long run. However, the reward does not say anything about the global performance of the agent. Thus, there is a need for a mechanism to measure how good in the long run being in a state can be. The concept of an action-value function Q^π describes the expected long-term reward we can obtain from being in a given state, taking an action and following a policy π. Using Bellman's equation the action-value function can be formalised as:

$$Q^\pi(s_t, a_t) = \mathbb{E}\left[R_t | s_t, a_t\right] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t, a_t\right], \quad (10)$$

where s_t and a_t are the state and action at time t respectively, $\mathbb{E}[\cdot]$ is the expectation operator, γ is the discount factor, a parameter that weights how much we trust long term estimations of the future rewards achievable from a given state, R_t is the cumulative reward achievable in the long run, and r_t is the reward at time t.

However, there are two distinctive characteristics of the problem here analysed that will directly affect what MARL algorithms we can use: first, the state and action spaces are continuous – this is critical since most Reinforcement Learning (RL) algorithms can only deal with discrete state and action spaces; second, the system involves the interaction of multiple agents at the same time, which makes learning the optimal policy more complicated than in a single agent scenario.

Several approaches have been proposed in the literature to extend RL to multi-agent scenarios, most based on the application of game theory to the solution of the Q-learning problem, such as Nash Q-learning for mixed cooperative/competitive tasks or minimax Q-learning for fully competitive scenarios [16]. For the load frequency control problem, we have adopted Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [17], since it applies to multi-agent systems and does so in continuous state and action spaces.

B. Multi-Agent Deep Deterministic Policy Gradient

MADDPG is a MARL actor-critic algorithm that deals with continuous state and action spaces. MADDPG proposes centralising the training operation, where the critics use all available information to embed into the actors the dynamics of the environment and the dynamics of the rest of the agents as well. Then, during the operational phase only the actors are used. Actors only use local information. Each generation unit is defined by an actor and a critic. Each actor i estimates a change in action Δz_i given the state of the environment $\Delta\omega$ and its current secondary control action z_i. The critic takes the current state of the environment and the action estimated by all the actors of the system. It estimates the Q^π value of the state-action pair that is used to train each actor, see Fig. 1, where we denote by Δz_{-i} (Δz_{-j}) the action predicted by all other actors besides i (j) and z_{-i} (z_{-j}) the control action state of all other actors besides i (j).

Each actor and each critic are implemented by a deep Recurrent Neural Network, specifically by a Long Short-Term Memory Network (LSTM) [18]. MDPs rely on the Markov assumption, according to which the next state only depends on the current state so that past history must be embedded in the current state. However, when dynamics show complex behaviour, the Markov assumption may not hold. LSTMs implement memory gates to ensure that previous history is stored and acted upon if and when necessary [19].

The actor depicted in Fig. 2a is characterised as a network with a two-neuron input layer and a one-neuron output layer: it inputs $\Delta\omega$ and z_i and computes Δz_i directly. Then, the first hidden layer is a 100-neuron LSTM that acts as a memory device, and that is followed by three more fully-connected hidden layers composed of 1000, 100 and 50 neurons respectively.

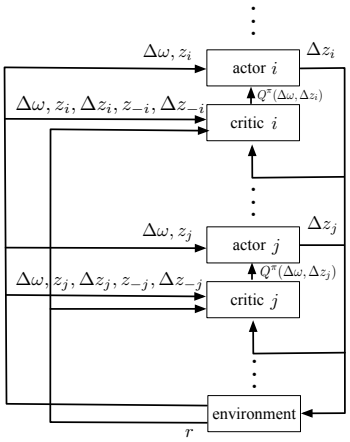
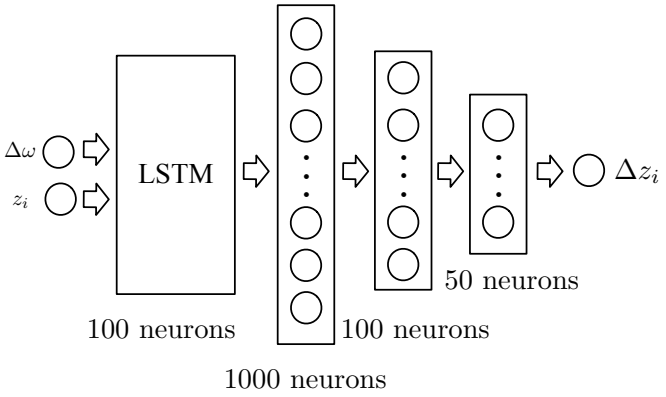
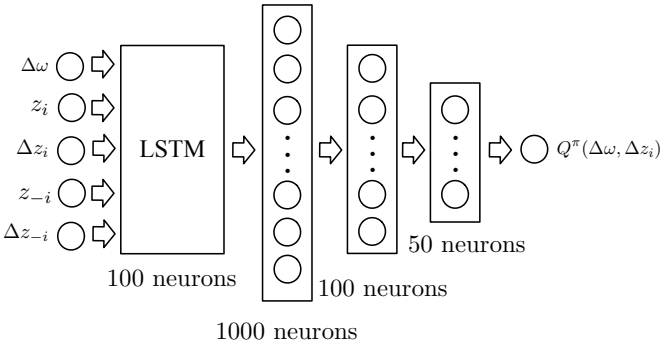


Fig. 1: MADDPG schema in a frequency control scenario.



(a) Architecture of the MADDPG actor.



(b) Architecture of the MADDPG critic.

Fig. 2: Architectures of the MADDPG function approximators.

The input to the critic depicted in Fig. 2b is the frequency state of the network $\Delta\omega$, the secondary control action z_i, the change in the action predicted by the actor associated to that critic Δz_i, the secondary control actions z_{-i} and the changes in the action predicted by all other actors Δz_{-i}. Again, after the input layer, there are a 100-neuron LSTM and three 1000-neuron, 100-neuron and 50-neuron hidden layers. The output layer only has one neuron that computes the $Q^\pi(\cdot)$ value of the state-action pair estimated by the actor associated to that critic.

Nominal frequency	$f^{\text{nom}} = 50$ Hz
Initial operating point (two agents)	$P_i = 1.5$pu, $i = 1, 2$
Initial operating point (four agents)	$P_i = 0.75$pu, $i = 1, ..., 4$
Initial operating point (eight agents)	$P_i = 0.375$pu, $i = 1, ..., 8$
Inertia parameter	$M = 0.1$pu
Droop	$R_D = 0.1$pu
Load damping	$D = 0.0160$pu
Generator dynamics time constant	$T_G = 30$s

TABLE I: System data; pu refers to 100 MVA base power reference.

The reward function must be based on the frequency state of the environment. This is straightforward since we can set a higher reward for smaller frequency deviations $\Delta\omega$. We may formulate a general form reward function as the following exponential function:

$$r = ab^{|\Delta_\omega|}, \tag{11}$$

where a is the maximum achievable reward and $b \in (0, 1)$ is a parameter that controls the rate of decayment of the reward; this condition ensures that r will reward actions that help in frequency restoration. The closer Δ_ω is to zero, the higher the reward.

IV. Numerical Results

In this section, we validate the proposed approach with three test systems. We formulate the reward function and present the results of the training period. Next we demonstrate that the generators are able to restore the system frequency back to nominal when a change of load occurs in a decentralised way.

The test cases comprise of a group of generation units or agents that interact with a load. More precisely, the proposed solution has been tested with two, four and eight generators-agents interacting with a load. The parameters of the environment can be found in Table I. In each training episode, the load varies around a nominal set point randomly. The modification is indicated by $P_L \pm \Delta P_L = 3 \pm \beta$ pu, where β follows a uniform distribution. The reward function has been derived following (11). We set $a = 10$, $b = 0.5$pu. Thus we have:

$$r = 10(0.1^{|\Delta_\omega|}). \tag{12}$$

During operation, only the actors interact with the environment. They only observe local information about the frequency of the system and the control action that they are executing. As a consequence of the training phase, they know how to act according to the states of the environment in order to keep load and generation balanced. The validation of the training is tested by changing the load by 0.15pu and seeing how the generators modify their output.

A. Training period

In Fig. 3 the cumulative reward obtained by the agents during training can be seen. The maximum achievable return obtained in one episode is $1,000$. The maximum reward per step is 10 and the number of steps per episode is 100. The performance of the agents shows an ascending trend oscillating around 900. This means that the agents are learning and have discovered how to obtain higher rewards.

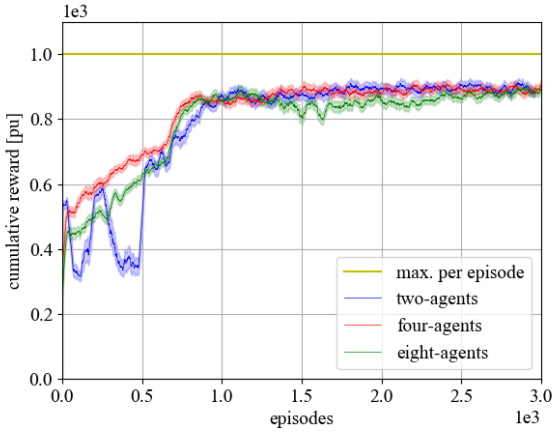


Fig. 3: Cumulative reward 95% confidence.

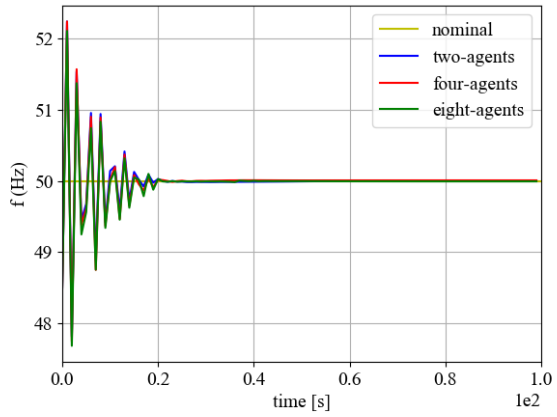


Fig. 4: Frequency after change in load by 0.15pu.

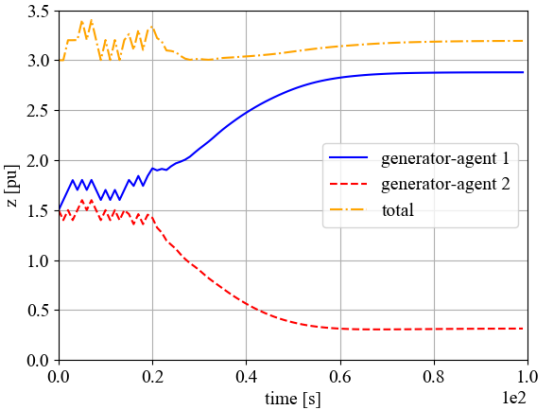
B. Operational period

In order to validate the training we change the load by 0.15pu and see how the frequency changes as well as the total system power. In Fig. 4 we can see that frequency is restored around the nominal set point and demand generation balance is rapidly achieved in all test cases. This result shows that actors learn what to do when load increases without the need of centralising any information. This is achieved since primary and secondary layers are embedded inside the agents as a consequence of them learning during training that keeping $\Delta\omega$ close to 0 is associated with high rewards.

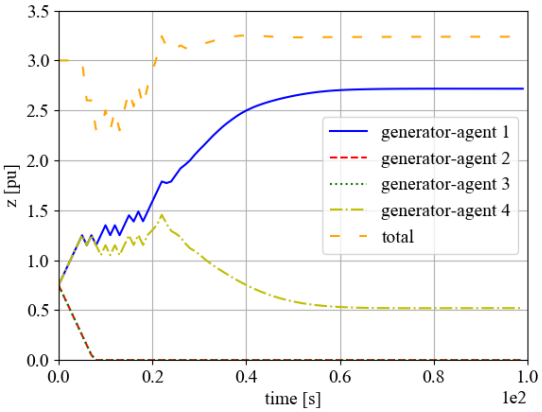
As it can be seen in Fig. 5, the agents find valid policies to solve the problem. However, the behavior may be unrealistic, e.g., some agents don't produce energy at all and just one or two agents are in charge of balancing generation and load. Introducing information about the generation cost of each unit should help the agents inferring how much they have to produce exactly, which is part of future work.

V. Concluding Remarks

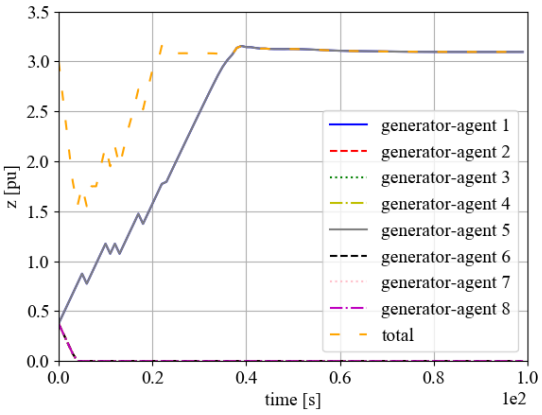
In this paper, we developed a framework to perform load frequency control in a decentralised way. In this regard, we



(a) Two-agents system.



(b) Four-agents system.



(c) Eight-agents system.

Fig. 5: Output after change in load by 0.15pu.

presented the basics of load frequency control, i.e. primary, secondary and tertiary control. Next, we casted the problem in a MARL setting, specified the reward function, and the actor-critic architecture that was suitable for implementing MADDPG in the load frequency control problem. We defined the reward function that ensures frequency restoration. We validated the proposed approach through numerical results in a small-scale test system. More specifically, the results showed that MADDPG performed efficiently when implementing primary and secondary control, i.e., frequency restoration, in a two-, four- and eight-agents system. The advantage of the proposed approach is that it offers a completely decentralised solution for the frequency control problem.

For future work, we mainly plan working on adding the tertiary control layer to the proposed solution, i.e., take operational costs into account. The applicability and scalability of these techniques in more complex scenarios also needs to be investigated. It would be interesting as well to check the validity of MADDPG to deal with different types of generation plants and to estimate to what extent MARL could be applied in this kind of contexts.

REFERENCES

[1] A. Singh and B. S. Surjan, "Microgrid: A review," *IJRET: International Journal of Research in Engineering and Technology*, vol. 3, no. 2, pp. 185–198, Feb. 2014.

[2] X. Wang, J. M. Guerrero, F. Blaabjerg, and Z. Chen, "A review of power electronics based microgrids," *Journal of Power Electronics*, vol. 12, no. 1, pp. 181–192, Jan. 2012.

[3] S. T. Cady, M. Zholbaryssov, A. D. Domínguez-García, and C. N. Hadjicostis, "A distributed frequency regulation architecture for islanded inertialess ac microgrids," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 1961–1977, Nov. 2017.

[4] M. Ahmed, "An overview on microgrid control strategies," *International Journal of Engineering and Advanced Technology*, vol. 4, pp. 93–98, 06 2015.

[5] D. Apostolopoulou, P. W. Sauer, and A. D. Domínguez-García, "Distributed optimal load frequency control and balancing authority area coordination," in *2015 North American Power Symposium (NAPS)*, Oct. 2015, pp. 1–5.

[6] D. Apostolopoulou, P. W. Sauer, and A. D. Domínguez-García, "Balancing authority area coordination with limited exchange of information," in *2015 IEEE Power & Energy Society General Meeting*. IEEE, 2015, pp. 1–5.

[7] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Transactions on Neural Networks*, vol. 16, pp. 285–286, 1988.

[8] A. J. Wood and B. F. Wollenberg, *Power Generation, Operation and Control*. New York: Wiley, 1996.

[9] P. W. Sauer and M. A. Pai, *Power System Dynamics and Stability*. Upper Saddle River, NJ: Prentice Hall, 1998.

[10] H. Glavitsch and J. Stoffel, "Automatic generation control," *International Journal of Electrical Power & Energy Systems*, vol. 2, no. 1, pp. 21 – 28, 1980.

[11] D. S. Kirschen and G. Strbac, *Fundamentals of Power System Economics*. Wiley, 2004.

[12] H. S. Nwana, "Software agents: an overview," *The Knowledge Engineering Review*, vol. 11, no. 3, pp. 205–244, 1996.

[13] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *In Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann, 1994, pp. 157–163.

[14] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[16] L. Bosoniu, R. Babuŝka, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[17] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 6379–6390.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," 2016.