Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning

Georgios Papoudakis, **Filippos Christianos**, **Arrasy Rahman**, **Stefano V. Albrecht**
School of Informatics, The University of Edinburgh
{g.papoudakis, f.christianos, arrasy.rahman, s.albrecht}@ed.ac.uk

Abstract

Recent developments in deep reinforcement learning are concerned with creating decision-making agents which can perform well in various complex domains. A particular approach which has received increasing attention is multi-agent reinforcement learning, in which multiple agents learn concurrently to coordinate their actions. In such multi-agent environments, additional learning problems arise due to the continually changing decision-making policies of agents. This paper surveys recent works that address the non-stationarity problem in multi-agent deep reinforcement learning. The surveyed methods range from modifications in the training procedure, such as centralized training, to learning representations of the opponent's policy, meta-learning, communication, and decentralized learning. The survey concludes with a list of open problems and possible lines of future research.

1 Introduction

Deep learning has revolutionized the development of agents which can act autonomously in complex environments. Traditional reinforcement learning (RL) methods, using tabular representations or linear function approximators, are difficult to scale to high-dimensional environments, and are mostly applied to small grid worlds or environments that provide a high-level state representation. The combination of deep learning with existing RL algorithms enabled the development of agents that act in environments with larger state spaces, such as images. Recently, some promising deep RL algorithms that can handle large state environments both with continuous and discrete action spaces have been proposed (Mnih *et al.*, 2015; Schulman *et al.*, 2017; Fujimoto *et al.*, 2018).

Multi-agent systems consist of multiple agents acting and learning in a shared environment. Many real-world decision making problems can be modeled as multi-agent systems, such as autonomous vehicles, resource allocation problems, robot swarms, and human-robot interaction. Despite the recent success of deep RL in single-agent environments, there are additional challenges in multi-agent RL. One major challenge, and the focus of this survey, is the non-stationarity of multi-agent environments created by agents that change their policies during the training procedure. This non-stationarity stems from breaking the Markov assumption that governs most single-agent RL algorithms. Since the transitions and rewards depend on actions of all agents, whose decision policies keep changing in the learning process, each agent can enter an endless cycle of adapting to other agents.

Additional problems in multi-agent systems include multi-agent credit assignment, partial observability, and heterogeneity. In the first problem, only a subset of agents contribute to a reward, and we need to identify and reward them while avoiding punishing agents that acted optimally. Partial observability consists of agents having access only to their local observations and not the actual state of the environment, which can significantly hinder training performance. Heterogeneity refers to the fact that agents may have different sensing and acting capabilities, and that their learning algorithms may differ (Albrecht and Ramamoorthy, 2012).

Multi-agent RL is a widely studied topic with surveys ranging from modelling other agents (Albrecht and Stone, 2018) to transfer learning (Da Silva and Costa, 2019) and non-stationarity (Hernandez-Leal *et al.*, 2017). However, there is a recent focus on multi-agent deep RL which is not discussed in the above surveys. In this work, we aim to consolidate progress in the area, and discuss non-stationarity in deep multi-agent RL settings. The survey concludes by discussing open problems and possible future research directions.

2 Background

In this section, we will briefly refer to definitions and previous works on RL, and we will formulate the non-stationarity problem of multi-agent RL.

2.1 Markov Decision Processes

We can mathematically model a decision making problem using Markov Decision Processes (MDP). An MDP is defined as (S, A, r, T), where S is the set of possible states, A is the set of available actions, $r : S \times A \times S \to \mathbb{R}$ is the reward function and $T : S \times A \times S \to [0, 1]$ is the probability distribution over next states given the current state and action. Additionally, in every MDP a stochastic policy function $\pi : S \times A \to [0, 1]$ or a deterministic policy function $\mu : S \to \mathbb{R}^{|A|}$ is defined to select an action for the current state.

Given a policy π, an MDP has two value functions. The state value function $V : S \to \mathbb{R}$ is the expected sum of discounted rewards from a given state under the policy π; $V^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{t=0}^{N} \gamma^t r_t | s]$, and the state-action value function $Q : S \times A \rightarrow \mathbb{R}$ is the expected sum of discounted rewards from a given state and action under the policy π; $Q^{\pi}(s, a) = \mathbb{E}_{\pi}[\sum_{t=0}^{N} \gamma^t r_t | s, a]$. In the equations γ is the discount factor, taking values in $[0, 1]$, and r_t is the reward after t time steps.

Solving an MDP requires computing the optimal policy. That is the policy that maximizes the expected discounted sum of rewards $V^*(s) = \max_{\pi} V^{\pi}(s)$. If access to the reward and transition function is available, this problem can be solved by iterating the Bellman equations and using dynamic programming. However, these two functions are unknown in most scenarios. In this case, RL approaches are used to compute the optimal policy.

2.2 Reinforcement Learning Methods

RL consists of multiple methods for solving MDPs. Temporal difference (TD) methods work by estimating the value functions (or the state-action value function) by interacting with the environment. The greedy policy is computed using the following expression: $\pi(s) = \arg\max_a Q(s, a)$. A commonly used method is Q-learning (Watkins and Dayan, 1992). The state-action value function (Q-values) can be expressed in tabular form (storing a separate value for every state and action) or can be approximated using a parameterized function. It is then updated by minimizing the following loss function: $L = \frac{1}{2}(r + \gamma \max_{a'} Q(s', a') - Q(s, a))^2$. This loss can be minimized using a gradient optimizer, either with respect to Q-values (in the tabular case) or the parameters.

An alternative model-free approach are policy gradient (PG) methods. Given a stochastic or a deterministic policy parameterized by parameters θ, the goal is to compute the policy that maximizes the expected discounted sum of rewards from the first state. Therefore, the objective function is $J = V^{\pi}(s_0)$. Using the policy gradient theorem, the gradient of the objective with respect to the parameters of the policy can be computed by maximizing the objective using a gradient-based optimizer. In the case of stochastic policy (Sutton *et al.*, 2000) $\nabla_{\theta} J = \mathbb{E}[G_t \nabla_{\theta} \log \pi_{\theta}(a|s)]$. In the case of deterministic policy (Silver *et al.*, 2014) $\nabla_{\theta} J = \mathbb{E}[\nabla_{\theta} \mu(s) \nabla_a Q(ts, a)|_{a=\mu(s)}]$.

In the tabular case or when combined with linear approximators, both TD and PG methods are difficult to scale to large action and state spaces. Thus, deep networks have been used (Mnih *et al.*, 2015) to address this issue. Mnih *et al.* (2015) proposed *experience replay* and the use of *target networks* in order to deal with instability issues that come with deep networks.

2.3 Markov Games

Markov games (Littman, 1994) are a generalization of MDP to multi-agent settings. A Markov game is defined as (I, S, A, r, T) where, I is the set of N agents, S is the set of states, $A = A_1 \times A_2 ... \times A_N$ is the set of actions of all the agents, $r = (r_1, r_2, ..., r_N)$, where $r_i : S \times A \times S \rightarrow \mathbb{R}$ is the reward function of agent i. Usually, in cooperative setting $r_1 = r_2 ... = r_N$. $T : S \times A \times S \rightarrow [0, 1]$ is the probability distribution over next states, given the current state and the actions of all agents.

In this survey, we assume that each agent has access only to a local observation o and not to the full state of the environment. As a result, the policy of the agents is conditioned on their history of observations $h_i = [o_{i,t}, o_{i,t-1}, o_{i,t-2}, ...]$. Given that the history of all agents $H = [h_1, h_2, ..., h_N]$, the joint policy is described as $\pi(\mathbf{a}|H) = (\pi_1(a_1|h_1), \pi_2(a_2|h_2), ..., \pi_N(a_N|h_N))$. A common simplification is to condition the policy of each agent's only on the most recent local observation, $\pi(\mathbf{a}|o_t) = (\pi_1(a_1|o_{1,t}), \pi_2(a_2|o_{2,t}), ..., \pi_N(a_N|o_{N,t}))$. However, conditioning on the history of observations experimentally leads to better results in partially observable environment. Similarly to MDP, the goal of each agent is to maximize its expected discounted sum of rewards $V_i^{\pi_i}(h_i) = E[\sum_{i=0} \gamma^i r_i | h_i]$.

2.4 Centralized and Decentralized Architectures

Two main architectures can be used for learning in multi-agent systems. The first architecture is centralized learning. In centralized learning, agents are jointly modelled and a common centralized policy for all the agents is trained. The input to the network is the concatenation of the observation of all the agents, and the output is the combination of all the actions. The most important issue of centralized architectures is the large input and output space. The input increases linearly and the output exponentially with respect to the number of agents.

On the other hand, in decentralized learning, the agents are trained independently from the others. Each agent has its policy network, takes a local observation as input and outputs an action. Although this method does not suffer from scalability issues, many different problems arise as well. Some of these issues are non-stationarity of the environment, the credit assignment problem and lack of explicit coordination. In this survey, we focus on works that attempt to handle issues related to non-stationarity.

2.5 The Non-Stationarity Problem

In Markov games, the state transition function T and the reward function of each agent r_i depend on the actions of all agents. During the training of multiple agents, the policy of each agent changes through time. As a result, each agents' perceived transition and reward functions change as well. Single-agent RL procedures which commonly assume stationarity of these functions might not quickly adapt to such changes.

As an example, consider a simple game like repeated Rock-Paper-Scissors. In it, two agents simultaneously perform actions $a_i \in \{a_{paper}, a_{rock}, a_{scissors}\}$ and receive their respective rewards. The agents, seeking to maximize their reward, learn a policy π_i using the interaction history. Ideally, the policy would learn to play actions that beat the opponent's most used actions. For instance, if agent 1 showed a preference to a_{paper} then π_2 would learn to prioritize $a_{scissor}$. Subsequently, agent 2 accumulates a history with that action, to which agent 1 reacts by changing its policy accordingly. This process may continue in this fashion, leading to what we refer to as non-stationarity due to changing behaviors.

3 Dealing with Non-Stationarity

The following subsections survey various approaches that have been proposed to tackle non-stationarity in multi-agent deep

RL. These approaches range from using modifications of standard RL training methods to computing and sharing additional opponent information. Details are summarized in Table 1.

3.1 Centralized Critic Techniques

A step toward dealing with non-stationarity is the centralized critic architecture. For this architecture, an actor-critic algorithm is used, which consists of two components. The critics' training is centralized and has access to the observations and actions of all agents, while the actors' training is decentralized. Since the actor computes the policy, the critic component can be removed during testing, and therefore the approach has fully decentralized execution. By having access to the observations and the actions of the opponent during training, the agents do not experience unexpected changes in the dynamics of the environment, which results in stabilization of the procedure.

Foerster *et al.* (2018b) used the actor-critic algorithm with stochastic policies to train agents and evaluated their method in Starcraft. The authors proposed a single centralized critic for all the agents and a different actor for each agent. Additionally, they proposed a modification in the advantage estimation of the actor-critic $A(s,a) = Q(s,a) - \sum_{a'_i} \pi_i(a'_i|o_i) Q(s,(a_{-i},a'_i))$. This modification serves two purposes. First of all, the advantage estimation, which is used in the policy gradient, is conditioned on the observations and actions of all the agents. Since each agent has access to the observations and actions of all the other agents, the policy gradient estimation is conditioned on the policy of the other agents and therefore the non-stationarity is addressed. Additionally, this counterfactual advantage can be interpreted as the value of action a compared to all the other action values. As a result, this advantage might be utilized to address the credit assignment problem, and this is the core contribution of the paper.

Lowe *et al.* (2017) proposed a multi-agent architecture using the deterministic policy gradient (Silver *et al.*, 2014) algorithm (MADDPG). In this method, each agent uses a centralized critic and a decentralized actor. Since the training of each agent is conditioned on the observation and action of all the other agents, each agent perceives the environment as stationary. Another extension of MADDPG is MiniMax MADDPG (M3DDPG) (Li *et al.*, 2019), which uses Minimax Q-learning (Littman, 1994) in the critic to exhibit robustness against different opponents with altered policies.

3.2 Decentralized Learning Techniques

Handling non-stationarity in multi-agent systems does not necessarily require centralized training techniques. An alternative decentralized approach that has been explored to handle non-stationarity in multi-agent deep RL problems is self-play. This approach trains a neural network, using each agents' own observation as input, by playing it against its current or previous versions to learn policies that can generalize to any opponents. This approach can be traced back to the early days of TD-Gammon (Tesauro, 1995) which managed to win against the human champion in Backgammon. More recently, self-play was extended to more complex domains such as Go (Silver *et al.*, 2017) and even complex locomotion environments with continuous state and action space (Bansal *et al.*, 2018).

In TD-Gammon, the neural networks are trained using temporal difference methods to predict the outcome of a game. Unlike recent approaches, self-play in TD-Gammon only plays against the current parameter setting of the neural network. It is noted in their work (Tesauro, 1995) that using self-play in this way might result in a lack of exploration during training due to the neural networks always choosing the same sequence of actions in different episodes in self-play. In their case, this problem did not occur due to the dynamics in Backgammon being stochastic.

In more recent applications of self-play (Silver *et al.*, 2017; Bansal *et al.*, 2018), an additional modification to the training process was made to ensure that the training is effective in environments with deterministic dynamics. In this case, recent self-play approaches stored the neural network parameters at different points during learning. Subsequently, the opponent during the self-play process is chosen by randomly choosing between the current and previous versions of neural network parameters. Apart from extending self-play to deterministic environments, this allowed the neural network to generalize against a broader range of opponents. As a result, self-play managed to train policies that can generalize well in environments like Go (Silver *et al.*, 2017) and even complex locomotion tasks (Bansal *et al.*, 2018).

Another technique which has been used to allow decentralized training is by stabilizing experience replay. Despite playing an essential part in single-agent deep RL approaches (Mnih *et al.*, 2015), due to environments' non-stationarity, experience replay might store experiences that are no longer relevant for decentralized learning which results in worse performance. Foerster *et al.* (2017) proposed importance sampling corrections to adjust the weight of previous experience to current environment dynamics. When combined with independent Q-learning (Tan, 1993), this resulted in significantly better performance in a specific task in the Starcraft game.

3.3 Opponent Modelling

Another feasible direction that deals with non-stationarity is opponent modelling. By modelling the intentions and policies of other agents, the training process of the agents might be stabilized. Modelling other agents in multi-agent systems has been widely studied and offers many research opportunities (Albrecht and Stone, 2018). In this survey, we mostly focus on recent methods that learn models of opponents or use them to condition the agents' policy on them.

Raileanu *et al.* (2018) suggested an approach where agents use their policy to predict the behaviour of other agents. This method employs an actor-critic architecture and reuses the same network for estimating the goals of the other agents. In more details, a network $f(s_{s/o}, z_s, \bar{z}_o)$, with the inputs being the state, the goal of the agent and the goal of the other agent respectively, is being used in a forward pass to decide on an action. However, the same network is also used by switching the order of z_s and $\bar{z}_o$ to infer the goal of the other agent. Observing the actual actions of the opponent allows the agent to back-propagate and optimize the vector of trainable parameters $\bar{z}_o$. In contrast, He *et al.* (2016) developed a second, separate network to encode the opponent's behaviour. The combination of the two networks is done either by concatenating their hidden states or by the use of a mixture of experts. This separate network enables faster learning and even allows modelling of changing opponent behavior.

Zhang and Lesser (2010) and Foerster *et al.* (2018a) proposed a modification of the optimization function in policy gradient methods to incorporate the learning procedure of the opponents in the training of their agent. Given two agents with parameters θ_1 and θ_2 respectively, the authors proposed the optimization of $V_1(\theta_1, \theta_2 + \Delta\theta_2)$, where $\Delta\theta_2 = \nabla_{\theta_2} V_2(\theta_1, \theta_2)$ instead of the standard $V_1(\theta_1, \theta_2)$. In this way, the training agent has access to the learning trajectory of the opponents, and therefore, the training procedure does not suffer from non-stationarity. Zhang and Lesser (2010) assumed that the term $\Delta\theta_2$ is not differentiable with respect to θ_1 and they proved convergence to Nash equilibrium in 2-player 2-action games. On the other hand, Foerster *et al.* (2018a) proposed *Learning with Opponent Learning Awareness* (LOLA) where the term $\Delta\theta_2$ is differentiable with respect to θ_1 to exploit the opponent learning dynamics. LOLA experimentally led to tit-for-tat behaviour in a different number of games and successfully managed to cooperate in the Independent Prisoners Dilemma (IPD). Most works, such as Bowling and Veloso (2001) deviate in the IPD settings.

In order to be able to keep the stability of Zhang and Lesser (2010) and the opponent dynamics exploitation of Foerster *et al.* (2018a), Letcher *et al.* (2019) proposed Stable Opponent Shaping (SOS). Letcher *et al.* (2019) provided examples of differentiable games, where Stable Fixed Points exhibit better behaviour than Nash equilibrium and examples where LOLA fails to converge in an Stable Fixed Point. For this reason, the authors proposed a *partial stop-gradient operator*, which controls the trade-off between the two methods. SOS has guarantees of convergence, while at the same time, it results in the same or better performance than LOLA.

Another facet of opponent modelling that was enabled by the recent advances in training neural networks is related to learning representations for multi-agent learning. Approaches that fall under this category learn the representations by imposing a certain model structure to compute the representations. Architectures such as graph neural networks (Tacchetti *et al.*, 2019), feed-forward neural networks (Grover *et al.*, 2018), and recurrent neural networks (Rabinowitz *et al.*, 2018) can be used to produce the representations.

Specific loss functions are used along with gradient-based optimization to train these models to output representations which can predict specific information of the opponents such as their actions (Rabinowitz *et al.*, 2018; Grover *et al.*, 2018) or returns (Tacchetti *et al.*, 2019) received by the modelled agent. The representation models can be trained using loss functions commonly used in supervised learning. Additionally, Grover *et al.* (2018) provided an example loss function that combines action prediction loss while also maximizing the difference between representations of different agents' policies.

The representation networks are provided with opponents' observations as input during learning. The policy networks are then trained by receiving agent observations which have been augmented with output representations from the representation networks as input. Under the assumption that these trained representation models could generalize to opponents that have yet been encountered, these models should be able to provide additional information which might characterize opponents' policy. Empirically, this produced an increased performance against either learning (Rabinowitz *et al.*, 2018) or stationary opponents in various learning environments.

3.4 Meta-Learning

Before advances in deep RL, approaches like tracking (Sutton *et al.*, 2007) and context detection (Da Silva *et al.*, 2006) were proposed to allow quicker adaptation in non-stationary environments. Both approaches adopt a more reactive view on handling non-stationarity by using learning approaches that attempt to quickly change the policy or environment models once changes in environment dynamics occur. However, the results of locomotion tasks proposed by Finn *et al.* (2017) highlighted how reactive approaches such as tracking still cannot produce a quick adaptation of deep RL algorithms to changing dynamics using only a few learning updates.

Instead of formulating a learning algorithm which can train deep neural networks to react to changing environment dynamics, another approach is to anticipate the changing dynamics. An optimization process can then be formulated to find initial neural network parameters that, given the changing dynamics, can learn using small amounts of learning updates. Meta-learning approaches like *Model Agnostic Meta Learning* (MAML) (Finn *et al.*, 2017) specifically addresses optimization for this particular problem. Al-Shedivat *et al.* (2018) further extended MAML to handle non-stationarity in multi-agent problems.

The proposed method by Al-Shedivat *et al.* (2018) provided an optimization process to search for initial neural network parameters, θ, which can quickly adapt to non-stationarity. The optimization process first generates rollout data, $\tau_{\mathcal{T}_i}$, from task $\mathcal{T}_i$. It then updates θ according to a policy gradient update rule to produce updated parameters ϕ. The method subsequently runs the policy parameterized by ϕ in the following task, $\mathcal{T}_{i+1}$, which results in a performance denoted by $\mathcal{R}_{i+1}(\phi)$. The proposed approach finally utilizes a policy gradient algorithm to search for θ which maximizes the expected performance after update, $\mathbb{E}_{\tau_{\mathcal{T}_i} \sim P_T(\tau|\theta)} \left[\mathbb{E}_{\tau_{\mathcal{T}_{i+1}} \sim P_{T+1}(\tau|\phi)} \left[\mathcal{R}_{i+1}(\phi) | \tau_{\mathcal{T}_i}, \theta \right] \right]$. By explicitly optimizing the initial model parameters based on their expected performance after learning, the proposed meta-learning approach was able to significantly outperform adaptation methods such as tracking and other meta-learning adaptation strategies which have performed well in single-agent environments. This was tested in iterated adaptation games where an agent repeatedly play against the same opponent while only allowed to learn in between each game.

3.5 Communication

Finally, the last category of methods that we discuss for handling non-stationarity is communication. Through communication, the different training agents can exchange information about their observations, actions and intentions to stabilize their training. While communication in multi-agent systems is a well-studied topic, we will focus on recent methods that learn to communicate using multi-agent deep RL.

A step in this direction is the work of Foerster *et al.* (2016b), who proposed the *Deep Distributed Recurrent Q-Networks*, an architecture where all the agents share the same hidden layers and learn to communicate to solve riddles. In the same direction, Sukhbaatar *et al.* (2016) proposed *CommNet*, an architecture, where the input to each hidden layer $h_j^i = f(h_{j-1}^i, c_{j-1}^i)$ for each agent i, is the previous layer h_{j-1}^i and a communication message $c_{j-1}^i = \frac{1}{I-1}\sum_{i'} h_{j-1}^i$, which is the average of the previous hidden layers of all the other agents. Therefore, the agents learn to communicate by sharing the extracted features of their observations in some cooperative scenarios. Singh *et al.* (2019) proposed the *Individualized Controlled Continuous Communication Model* (IC3NET), which is an extension of CommNet in competitive setting. In IC3NET, there is an additional communication gate, which either allows or blocks communication between agents, for example in competitive settings.

All of the previous approaches assume that all agents have access to the hidden layers of the other agents. An elimination of this constraint was proposed by Foerster *et al.* (2016a). The authors first suggested the *Reinforced Inter-Agent Learning*, where each agent has two Q-networks. The first network outputs an action and the second a communication message which is fed in the Q-networks of the other agents. Both networks are trained using DQN. In the same work, they also developed the *differentiable inter-agent learning*, where they train only the action network with DQN, while they push the gradients of the other agents, through the communication channel to the communication network. This approach is similar to the work of Mordatch and Abbeel (2018), where the authors proposed a model that takes as input the messages of other agents and learns to output an action and a new communication message.

4 Open Problems

Based on the methods outlined in this survey, we identify several open problems with regards to non-stationarity and possible lines of future research.

4.1 Transfer Learning for Non-Stationarity

Several transfer learning approaches for multi-agent systems have been covered in this survey. In this case, the learned representations and initialization values from meta-learning (Al-Shedivat *et al.*, 2018) and approaches which learn opponent representations (Grover *et al.*, 2018; Tacchetti *et al.*, 2019; Rabinowitz *et al.*, 2018) can be viewed as transferred knowledge which might result in quicker adaptation to non-stationarity. Despite recent advances, there are still open questions regarding the form of the knowledge being transferred and how to leverage them for quicker adaptation.

4.2 Open Multi-Agent Systems

In real-world problems, the number of agents in an environment can be large and diverse. Furthermore, the number of agents might change due to agents leaving or entering the environment. This problem setting is often termed an *open* multi-agent system (Chandrasekaran *et al.*, 2016). The change in the number of agents can cause an action to have different consequences at different points during learning. For example, a certain action might lead to situations with high returns when another cooperative agent is in the environment, while also being inconsequential when the other agent left the environment.

None of the techniques presented in this survey were tested in environments with changing number of agents. In general, transfer learning approaches that can reuse knowledge between problems with a varying number of agents might be a potential avenue of research in this topic. Furthermore, research on how agents can effectively deal with heterogeneity in capabilities and learning algorithms (Albrecht and Ramamoorthy, 2012) will play an important part.

4.3 Limited Access to Opponent Information

A large number of the works that we presented in opponent modelling, such as (He *et al.*, 2016; Grover *et al.*, 2018), requires access to the opponent's observations and chosen actions. While this is not a strong assumption during centralized training, it is very limiting during testing, especially when there is not established communication between the agents. More precisely, assuming that we have access in the observations and actions of the opponents during testing is too strong. Therefore, it is an open problem to create models that do not rely on this assumption.

4.4 Convergence Properties

An open problem with current multi-agent deep RL methods is the lack of theoretical understanding of their convergence properties and what types of outcomes they tend to achieve. A theoretical concept that could be used to encourage convergence are game-theoretic equilibria, such as Correlated and Nash equilibrium. One drawback of these approaches is the requirement of computing equilibrium solutions (Greenwald *et al.*, 2003), as well as non-uniqueness of equilibria which requires some form of coordinated equilibrium selection. A recent example in this direction is the work by Li *et al.* (2019) who used an approximate solution of the Minimax equilibrium. Therefore, an interesting research direction is to investigate approximate solutions in order to incorporate the notion of these equilibria in multi-agent deep RL.

4.5 Multi-Agent Credit Assignment

Another problem which arises from approaches with decentralized execution is related to assigning credit to agents in the environment. Despite improving centralized training (Foerster *et al.*, 2018b) or learning opponent representations (Tacchetti *et al.*, 2019), methods to handle this problem can still be improved. In general, finding alternative neural network architectures and learning approaches that can decompose rewards (Rashid *et al.*, 2018; Sunehag *et al.*, 2018) for smaller group of agents can be a possible future research direction.

5 Conclusion

To summarize our contribution, we identified five different approaches for handling non-stationarity in multi-agent deep RL. In order to understand their characteristics and their differences, we provided a detailed categorization in Table 1. Finally, we outlined several open problems with respect to non-stationarity and possible directions of future research.

	Settings	Training	Execution	Modelling	Opp. Info	Algorithm	Num agents
Tacchetti *et al.* (2019)	Mixed	Centr.	Decentr.	Explicit	Obs / actions	A2C	≥ 2
Singh *et al.* (2019)	Mixed	Decentr.	Decentr.	No	None	PG	≥ 2
Letcher *et al.* (2019)	Mixed	Decentr.	Decentr.	Explicit	Parameters	PG	2
Li *et al.* (2019)	Mixed	Centr.	Decentr.	No	Obs / actions	DDPG	≥ 2
Al-Shedivat *et al.* (2018)	Comp.	Decentr.	Decentr.	No	None	PPO	2
Bansal *et al.* (2018)	Comp.	Decentr..	Decentr.	No	None	PPO	2
Raileanu *et al.* (2018)	Mixed	Centr.	Centr.	Explicit	Obs / actions	A3C	2
Mordatch and Abbeel (2018)	Coop.	Decentr.	Decentr.	No	None	PG	≥ 2
Foerster *et al.* (2018a)	Mixed	Decentr.	Decentr.	Explicit	Parameters	PG	2
Grover *et al.* (2018)	Mixed	Centr.	Centr.	Explicit	Obs / actions	PPO / DDPG	2
Rabinowitz *et al.* (2018)	Mixed	Centr.	Centr.	Explicit	Obs / actions	Imit.	≥ 2
Foerster *et al.* (2018b)	Coop.	Centr.	Decentr.	No	Obs / actions	Actor-critic	≥ 2
Lowe *et al.* (2017)	Mixed	Centr.	Decentr.	No	Obs / actions	DDPG	≥ 2
Foerster *et al.* (2017)	Mixed	Decentr.	Decentr.	No	None	Q-learning	≥ 2
Sukhbaatar *et al.* (2016)	Coop.	Decentr.	Decentr.	No	None	PG	≥ 2
Foerster *et al.* (2016a)	Coop.	Centr.	Decentr.	No	None	Q-learning	≥ 2
He *et al.* (2016)	Mixed	Centr.	Centr.	Implicit	Obs	Q-learning	2
Zhang and Lesser (2010)	Mixed	Decentr.	Decentr.	Explicit	Parameters	PG	2

Table 1: Categorization of the surveyed algorithms that deal with non-stationarity. The algorithms are categorized based on the environment setting (cooperative, competitive, mixed), their training and execution method (centralized, decentralized), their type of modelling, the opponent information that they require, the learning algorithm that they use, and the number of agents that can be handled.

References

Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *International Conference on Learning Representations*, 2018.

Stefano V Albrecht and Subramanian Ramamoorthy. Comparative evaluation of MAL algorithms in a diverse set of ad hoc team problems. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 349–356, 2012.

Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, pages 66–95, 2018.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations*, 2018.

Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *International Joint Conference on Artificial Intelligence*, pages 1021–1026, 2001.

Muthukumaran Chandrasekaran, Adam Eck, Prashant Doshi, and Leenkiat Soh. Individual planning in open and typed agent systems. In *Conference on Uncertainty in Artificial Intelligence*, pages 82–91, 2016.

Felipe Leno Da Silva and Anna Helena Reali Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, pages 645–703, 2019.

Bruno C Da Silva, Eduardo W Basso, Ana LC Bazzan, and Paulo M Engel. Dealing with non-stationary environments using context detection. In *International Conference on Machine Learning*, pages 217–224, 2006.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate to solve riddles with deep distributed recurrent q-networks. *arXiv preprint arXiv:1602.02672*, 2016.

Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 1146–1155. JMLR.org, 2017.

Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 122–130, 2018.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*, pages 2974–2982, 2018.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596, 2018.

Amy Greenwald, Keith Hall, and Roberto Serrano. Correlated Q-learning. In *International Conference on Machine Learning*, pages 242–249, 2003.

Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In *International Conference on Machine Learning*, pages 1797–1806, 2018.

He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning*, pages 1804–1813, 2016.

Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

Alistair Letcher, Jakob Foerster, David Balduzzi, Tim Rocktäschel, and Shimon Whiteson. Stable opponent shaping in differentiable games. *International Conference on Learning Representations*, 2019.

Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *AAAI Conference on Artificial Intelligence*, 2019.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 157–163. 1994.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *AAAI Conference on Artificial Intelligence*, pages 1945–1952, 2018.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International Conference on Machine Learning*, pages 4215–4224, 2018.

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2018.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4292–4301, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pages 387–395, 2014.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *International Conference on Learning Representations*, 2019.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 2085–2087, 2018.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 2000.

Richard S Sutton, Anna Koop, and David Silver. On the role of tracking in stationary environments. In *International Conference on Machine Learning*, pages 871–878, 2007.

Andrea Tacchetti, H. Francis Song, Pedro A. M. Mediano, Vinicius Zambaldi, János Kramár, Neil C. Rabinowitz, Thore Graepel, Matthew Botvinick, and Peter W. Battaglia. Relational forward models for multi-agent learning. In *International Conference on Learning Representations*, 2019.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *International Conference on Machine Learning*, pages 330–337, 1993.

Gerald Tesauro. Temporal difference learning and TD-gammon. *Communications of the ACM*, (3):58–68, 1995.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, (3-4):279–292, 1992.

Chongjie Zhang and Victor Lesser. Multi-agent learning with policy prediction. In *AAAI Conference on Artificial Intelligence*, pages 927–934, 2010.