Cooperation in Multi-Agent Reinforcement Learning

Karush Suri, Dian Gadjov, Lacra Pavel
Department of Electrical & Computer Engineering, University of Toronto, Canada.
karush.suri@mail.utoronto.ca

Abstract

Advancements in Multi-Agent Reinforcement Learning (MARL) are motivated by cooperation in agents arising from Game Theory (GT). Agents must collaborate in practical scenarios in order to achieve complex objectives and attain strategies which depict optimal behavior. The need for cooperation is further highlighted in the case of partially-observed settings wherein agents have restricted access to environment observations. We revisit cooperation in MARL from the viewpoint of GT and stochastic dynamics of environments. The contributions of our work are twofold. (1) We analyze and demonstrate the effectiveness of cooperative MARL in the case of complex and partially-observed tasks consisting of high-dimensional action spaces and stochastic dynamics. (2) We leverage the empirical demonstrations to construct a novel optimization objective which addresses the detrimental effects of stochastic states across agents. Our large-scale experiments carried out on the StarCraft II benchmark depict the effectiveness of cooperative MARL and our novel objective for obtaining optimal strategies under stochastic dynamics.

1 Introduction

Reinforcement Learning (RL) has seen tremendous growth in applications such as arcade games [1], board games [2, 3], robot control tasks [4, 5] and lately, real-time games [6]. The rise of RL has led to an increasing interest in the study of multi-agent systems [7, 8], commonly known as Multi-Agent Reinforcement Learning (MARL). MARL provides significant benefits in comparison to contemporary single-agent methods [9]. The Multi-Agent framework allows the modelling of complex real-world systems which consist of dynamic and large-scale interactions between multiple agents [10]. Additionally, MARL enables the learning of diverse strategies which are essential for executing a range of different tasks by the same set of agents.

In the case of partially observable settings, MARL enables the learning of strategies from a GT perspective by utilizing cooperation across agents[11]. Agents collaborate with each other in a given environment to optimize the cumulative payoffs by means of a single utility function. Optimization of the joint utility function leads to optimal behavior [12, 13] in the long-horizon which is characterized by each agent executing its optimal strategy irrespective of other agents. Such a framework of learning strategies with collaborators and executing behaviors independently is often referred to as centralized training with decentralized control [14].

The regime of decentralized control is hindered by intrinsic stochasticity in the environment. Fast-paced dynamics are a common phenomenon observed in the case of single-agent RL methods. In the case of model-based RL [15], agents build a model of the environment which learns the dynamics of the environment. Such a scheme is used as an effective planning tool in the case of long-horizon tasks [16]. In the case of model-free RL methods, environment stochasticity is addressed by utilizing robust utility functions [17, 18] and effective exploration strategies [19]. On the other hand, MARL does not account for abrupt dynamics across agents as a result of which the system remains unaware of drastic changes in the environment [20]. Thus, addressing the learning of stochastic dynamics in the case of multi-agent settings requires attention from a critical standpoint.

We revisit cooperation in MARL from the perspective of GT and stochastic dynamics in the agents' environment. Our work assesses and demonstrates collaborative schemes in MARL under partially-observed settings which pose ill-conditioned objectives for the multi-agent system. More specifically, our twofold contributions are the following-

- We analyze and demonstrate the effectiveness of cooperative MARL for complex and partially-observed tasks consisting of high-dimensional action spaces and abrupt dynamics.
- We leverage the empirical demonstrations to construct a novel optimization objective which addresses the detrimental effects of stochastic states across agents.

Our large-scale experiments carried out on the StarCraft II benchmark depict the effectiveness of cooperative MARL and our novel objective for obtaining optimal strategies under stochastic dynamics.

2 Related Work

Growing advances in GT have given rise to efficient MARL algorithms and implementations in stochastic scenarios. Thsi section highlights some of the main contributions in learning of stochastic games which have paved the way for Multi-Agent learning.

2.1 Learning in Games

Learning in games is an active area of development which is motivated by the framework of repeated games [21]. Repetitions of games are also modeled as episodes which has given rise to episodic play and continuous control in the case of single-agent systems. While episodic play serves as the basis for fictitous [22] and best-response type learning [21], learning algorithms in games are primarily motivated by developments in the reinforcement play regime[9]. Learning algorithms are coupled with fast optimization techniques [1] to iterate over complex strategy spaces and achieve optimal behavior [9]. Additionally, developments in the learning regime such as the introduction of complex function approximators [23] for optimizing higher order utility functions has played a significant role in expanding computational capabilities of game theoretic learning [24].

[1] demonstrates the large-scale suitability of reinforcement learning to single-agent learning by making use of Q-learning [9, 25] which allows the agent to learn complex utility functions and generalize to different games [26] by making use of a common function approximator. Other methods in literature [27, 28, 4, 5] have improved upon the Q-learning framework to provide stability [27] and diversity [29] in learning. These improvements have played a key role in yielding state-of-the-art performance [8] on real-world games [2, 3] wherein the structure of payoff function is sparse [2] and the agent needs to explore a larger action space [3] in order to achieve optimal strategies.

2.2 Multi-Agent Learning

Most multi-agent methods are based on the paradigm of centralized training and decentralized control [14, 30, 31] wherein agents learn to collaborate [32] and optimize their utility function [33]. The fundamental work on MARL originates from the IQL [34] framework wherein agents learn to collaborate with independent utilities. While the IQL framework serves as a critical point for advances in MARL, the work of [35] presents the common knowledge framework wherein agents collaborate by gaining mutual information about the task and establishing a structured protocol for communication [36]. Such methods have given rise to large-scale agents capable of optimal behavior on high-dimensional control tasks [7, 37, 38]. Some of these methods suffer from estimation biases [39, 40] stemming from the function approximator [41] used to maximize the utility function. Various MARL methods [42] make use of a dual function approximator approach which increases the accuracy of estimates. The Weighted Double Deep Q-Network (WDDQN) provides stability and sample efficiency for fully-observable settings. In the case of partially-observed scenarios, Weighted-QMIX (WQMIX) [43] yields a more sophisticated weighting scheme which aids in the retrieval of optimal strategy [44].

Despite the recent success of RL [28, 45] MARL agents suffer from spurious state spaces and encounter sudden changes in trajectories. These anomalous transitions between consecutive states are often termed as surprise [17]. Quantitatively, surprise can be inferred as a measure of deviation [16, 19] among states encountered by the agent during its interaction with the environment. In the case of single-agent methods, surprise results in sample-inefficient learning [17]. This can be tackled by making use of rigorous exploration strategies [46, 47]. However, such solutions do not show evidence for multiple agents consisting of individual partial observations [48].

3 Preliminaries

3.1 Stochastic Markov Games

We revisit Stochastic Markov Games [49] which serve as the fundamental basis for MARL. A Markov Game [12] is a generalization of a Markov Decision Process (MDP) [9] which is described using the tuple $(\mathcal{S}, \mathcal{A}^1, \mathcal{A}^2 ... \mathcal{A}^n, r^1, r^2, ... r^n, N, P, \gamma)$ where $\mathcal{S}$ is the finite state space, $\mathcal{A}^a$ is the action space corresponding to agent a such that $a \in N$ where $N = \{1, 2, ... n\}$ is the set of all agents, $r^a : \mathcal{S} \times \mathcal{A}^a \rightarrow [r^a_{min}, r^a_{max}]$ is the payoff observed by agent a and bounded in $[r^a_{min}, r^a_{max}]$, $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A}^1 \times \mathcal{A}^2 \times ... \mathcal{A}^n \rightarrow [0, \infty)$ presents the unknown transition model consisting of the transition probabilities to the next state $s' \in \mathcal{S}$ given the current state $s \in \mathcal{S}$ and γ is the discount factor. Each agent a performs its own action u^a which gives rise to the joint action $u = \{u^{(1)}, u^{(2)}, ... u^{(n)}\}$. Analogously, the action space can be written as the combination of all agents' action spaces $\mathcal{A} : \mathcal{A}^1 \times \mathcal{A}^2 \times ... \mathcal{A}^n$. Markov Games wherein each agent observes its own payoffs are called General Markov Games (GMGs) [49]. On the other hand, Markov Games in which all agents observe the same payoffs $r^1 = r^2 = ... r^n = r$ such that $r : \mathcal{S} \times \mathcal{A} \rightarrow [r_{min}, r_{max}]$ are called Team Markov Games (TMGs) [12]. Thus, a TMG can be compactly defined as a tuple of the form $(\mathcal{S}, \mathcal{A}, r, N, P, \gamma)$. The general framework of cooperative multi-agent learning makes use of TMGs.

3.2 Multi-Agent Learning

We review the MARL setup. The problem is modeled as a Partially Observable and Stochastic TMG [9] defined by the tuple $(\mathcal{S}, \mathcal{A}, r, N, P, Z, O, \gamma)$ where the state space $\mathcal{S}$ and action space $\mathcal{A}$ are discrete, $r : \mathcal{S} \times \mathcal{A} \rightarrow [r_{min}, r_{max}]$ presents the payoff observed by agents $a \in N$ bounded in the interval $[r_{min}, r_{max}]$ where N is the set of all agents, $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, \infty)$ presents the unknown transition model consisting of the transition probability to the next state $s' \in \mathcal{S}$ given the current state $s \in \mathcal{S}$ and joint action $u \in \mathcal{A}$ where $u = \{u_t^{(1)}, u_t^{(2)} ... u_t^{(n)}\}$ at time step t and γ is the discount factor. We consider a partially observable setting in which each agent a draws individual observations $z \in Z$ according to the observation function $O(s, u) : \mathcal{S} \times \mathcal{A} \rightarrow Z$. We consider a joint policy $\pi_\theta(u|s)$ which quantifies the probability of taking action u in state s as a function of the multi-agent model with its control parameters as θ. Standard RL defines the agent's objective to maximize the expected discounted payoff $\mathbb{E}_{\pi_\theta}[\sum_{t=0}^{T} \gamma^t r(s_t, u_t)]$ as a function of the parameters θ.

3.3 Q-Learning

We review the Q-learning setup in MARL. The action-value function, which is the expected sum of payoffs obtained in state s upon performing action u by following the policy π_θ, for an agent is represented in Equation 1.

$$Q(u, s; \theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=1}^{T} \gamma^t r(s, u) | s = s_t, u = u_t] \tag{1}$$

We denote the optimal policy π^*_θ such that $Q(u, s; \theta^*) \geq Q(u, s; \theta) \forall s \in S, u \in A$. In the case of multiple agents, the joint optimal policy can be expressed as the Nash Equilibrium [50] of the Stochastic TMG as expressed in Equation 2.

$$\pi^* = (\pi^{1,*}, \pi^{2,*}, ...\pi^{N,*}) \quad s.t. \quad Q(u^a, s; \theta^*) \geq Q(u^a, s; \theta) \quad \forall s \in S, u \in A, a \in N \tag{2}$$

Q-Learning is an off-policy, model-free algorithm suitable for continuous and episodic tasks. The algorithm uses semi-gradient descent to minimize the Temporal Difference (TD) error expressed in Equation 3.

$$\mathbb{L}(\theta) = \underset{b \sim R}{\mathbb{E}}[(y - Q(u, s; \theta))^2] \tag{3}$$

Here $y = r + \gamma \max_{u_{t+1} \in A} Q(u_{t+1}, s_{t+1}; \theta^-)$ is the TD target consisting of θ^- as the target parameters and b is the batch of tuples (s, u, r, s_{t+1}) sampled from memory R.

4 Cooperation in Multi-Agent Learning

We assess and lay out the framework for cooperative multi-agent learning in this section. The setting of partially-observable states consisting of stochastic dynamics is discussed from an intuitive viewpoint followed by detailed learning mechanisms in state-of-the-art MARL algorithms.

4.1 The Partial Observability Setting

Assessing an agent's actions with unknown and stochastic dynamics requires a partially-observable setting. In the case of a partially-observable TMG, the multi-agent system observes a common state s with each of its agents observing individual observations $z \in Z$. These individual observations serve as the agent's basis for selecting action a and optimizing the policy distribution $\pi(a_t|s_t)$. The agent optimizes over its policy by maintaining a belief b over its actions and partial observations. Since the environment is Markovian, the belief of the agent b_{t+1} in state s_{t+1} depends on its belief b_t in previous state s_t. This leads to the formulation of a belief update as expressed in Equation 4 where τ is a belief operator.

$$b_{t+1} = \tau(b_t, a_t, z_t) \tag{4}$$

Upon reaching state s_{t+1} the agent observes $z \in Z$ with probability $O(z|s_t, a_t)$. Since the belief b is a probability distribution over the state space $\mathcal{S}$, $b(s_t)$ denotes the probability that the environment is in state s_t. Thus, the updated belief can be expressed as in Equation 5 where η is a normalizing constant and $p(s_{t+1}|s_t, a_t)$ is the transition probability from state s_t to state s_{t+1} upon performing action a_t.

$$b_t(s_t) = \eta O(z|s_t, a_t) \sum_{s\ in \mathcal{S}} p(s_{t+1}|s_t, a_t) b_t(s_t) \tag{5}$$

Assessing multi-agent behavior in partially-observable settings requires a benchmark which can accurately model the stochastic dynamics of real-world systems. We select StarCraft II [51] scenarios particularly for two reasons. Firstly, micromanagement scenarios consist of a larger number of agents with different action spaces. This requires a greater deal of coordination. Lastly, micromanagement scenarios in StarCraft II consist of multiple opponents which introduce a greater degree of surprise within consecutive states. Irrespective of the time evolution of an episode, environment dynamics of each scenario change rapidly as the agents need to respond to enemy's behavior.

4.2 Learning Model-Free Behaviors

Learning model-free behaviors can be assessed from by making use of state-of-the-art RL algorithms as these allow the agent to learn from a scalar reward signal which is representative of the dynamics of the environment. RL methods have proven to be sample-efficient [1] with suitable convergence properties [9] in the case of well-defined tasks. Corresponding to the discrete action and state space settings, one of the most popular RL algorithms is Q-learning [9]. Q-learning consists of a number of variants which have been improved and utilized in the case of single-agent RL methods. However, its applications in MARL remain limited due to the scalability in the large number of agents [31] and computational expense incurred as a result of gradient-based methods.

We assess 4 Q-learning agents from the perspective of multi-agent learning, namely Independent Q-Learning (IQL) [34], Value Decomposition Networks (VDNs), QMIX [31] and QMIX with Surprise Minimization for RL (QMIX-SMiRL) [16]. Corresponding to each agent we retrieve the cost function $J(\theta)$ by establishing links between the method and standard Q-learning. Following the formulation of the cost function, a mathematical analysis of Q-values is carried out which forms the basis for empirical evaluation of multiple agents.

4.2.1 IQL

A natural way to model a multi-agent system is by defining a cost function $L_i(\theta)$ for each agent $i \in N$. The agent i is then allowed to minimize $L_i(\theta)$ in order to optimize its expected payoffs $Q_i(u^{(i)}, s; \theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=1}^{T} \gamma^t r(s, u)|s = s_t, u^{(i)} = u_t^{(i)}]$. This form of reward maximization is directly based on standard Q-learning as expressed in Equation 1 [34]. Each agent maintains its set of Q-values which are directly optimized only by the agent's action $u_t^{(i)}$ in a given state s_t.

Q-learning makes use of the TD error as expressed in Equation 3. TD error serves as a practical choice for multi-agent learning due to its long-horizon suitabiity. The objective makes use of discounted bootstrapped estimates to prevent the agent from exhibiting short-sighted behavior. Additionally, the gradient of the TD error is computationally tractable in comparison to other policy-based methods [9] which require likelihood ratio gradients in learning. Thus, TD error is utilized to formulate the objective for IQL. Since each agent exhibits individual behavior and consists of its own unique policy $\pi^i(a_t|s_t)$ governed by Q-values $Q_i(u^{(i)}, s; \theta)$, the TD error corresponding to this agent can be expressed as in Equation 6 where θ^- denotes the parameters of the target approximator.

$$L_i(\theta) = \underset{b \sim R}{\mathbb{E}}[(r + \gamma \max_{u_{t+1}^{(i)} \in A^{(i)}} Q_i(u_{t+1}^{(i)}, s_{t+1}; \theta^-) - Q_i(u_t^{(i)}, s_t; \theta))^2] \tag{6}$$

In practice, Equation 6 is optimized using the batch gradient descent algorithm wherein the gradient is computed over batches of Q-values in order to reduce variance in estimates. While the IQL objective provides a framework for multi-agent learning, it does not guarantee convergence due to three main reasons. Firstly, with respect to each agent, all other agents are a part of the environment dynamics and their actions will affect state transitions when the agent interacts with the environment. Thus, each agent faces a non-stationary problem as other agents change their behaviors during learning. Secondly, each agent is initialized with its own objective function which leads to a total of n objective functions. This results in a complexity which scales linearly in the number of agents, thus resulting in intractable optimization schemes for large number of agents. Lastly, while all agents are a part of the same system they do not share information or communicate with each other. This does not lead to mutual collaboration.

4.2.2 VDN

Limitations of independent objective functions in Q-learning methods highlight the need for mutual collaboration. In the case of large action spaces [52] and fast-paced dynamics, actions of a single agent have little impact in the maximization of per step returns. Additionally, the long-horizon nature [52] of problems may lead to sparse rewards which hinders credit assignment in hindsight.

A suitable alternative to tackle large action spaces is the decentralization of tasks for action selection [30]. Since the large action space corresponds to the increased number of control inputs, each set of inputs is grouped as the actions $u_t^{(i)}$ corresponding to an agent which is a part of the multi-agent system. Each agent in the system consists of its own Q-values which are utilized for decentralized execution of the task. These Q-values are a linear decomposition of the overall Q-value of the multi-agent system. Agents learn the linear additive decomposition of values which leads to centralized optimization of control inputs. This is the fundamental idea behind VDNs [30].

VDNs exploit the assumption that the joint action-value function $Q(u, s; \theta)$ can be additively decomposed into action-value functions across agents. The assumption is mathematically expressed in Equation 7 where $z^{(i)}$ denotes the observation of each agent i.

$$Q(u, s; \theta) \approx \sum_{i=1}^{n} Q_i(u^{(i)}, z^{(i)}; \theta) \tag{7}$$

In the case of TMGs, the decomposition corresponds to the case when the agents observe their own goals but not necessarily those of the teammates. Thus, the decomposition can be expressed more generally as in Equation 8 which is obtained from the joint Q-value of the agents.

$$\begin{aligned} Q(u, s, ; \theta) &= \mathbb{E}_{\pi^\theta}[\sum_{t=1}^{\infty} \gamma^{t-1} r(s, u) | s = s_t, u = u_t] \\ &= \mathbb{E}_{\pi^\theta}[\sum_{t=1}^{\infty} \gamma^{t-1} r_1(z^{(1)}, u^{(1)}) | z = z_t^{(1)}, u^{(1)} = u_t^{(1)}] \\ &+ \mathbb{E}_{\pi^\theta}[\sum_{t=1}^{\infty} \gamma^{t-1} r_2(z^{(2)}, u^{(2)}) | z = z_t^{(2)}, u^{(2)} = u_t^{(2)}] \\ &... + \mathbb{E}_{\pi^\theta}[\sum_{t=1}^{\infty} \gamma^{t-1} r_n(z^{(n)}, u^{(n)}) | z = z_t^{(n)}, u^{(n)} = u_t^{(n)}] \end{aligned} \tag{8}$$

Expressing the joint Q-value as a linear combination of individual agent Q-values results in mutual collaboration. The gradient of the joint Q-value $\nabla_\theta Q(u, s; \theta)$ accumulates information about other agents' actions which is propagated to each agent. VDNs make use of neural networks [23] as function approximators to estimate $Q(u, s, ; \theta)$. This enables the approximator to exploit gradient information using the backpropagation algorithm [23].

While VDNs serve as one of the first motivating examples for mutual collaboration using centralized optimization, their generalization capability is hindered by linearity in combinations [31]. Most practical scenarios consist of non-linear dynamics and ill-posed conditions which restrict a closed-form factorization of Q-values in VDNs. This often results in sub-optimal convergence on high-dimensional tasks [31].

4.2.3 QMIX

Linear decompositions can be generalized to non-linear expressions by utilizing non-linear approximators. Such a generalization results in inreased expressiveness of the action-value function. Q-values can be generalized to the larger family of monotonic functions which motivate consistency among estimates. However, monotonicity constraints need to be enforced which would satisfy Equation 9.

$$\frac{\partial Q}{\partial Q_i} \geq 0. \quad \forall i \in N \tag{9}$$

Equation 9 lays out the key idea behind the QMIX [31] algorithm. While VDN is realized as a "hard" factorization for extracting decentralized policies, QMIX is a "soft" generalization of the value factorization problem posed by VDN. In order to maintain consistency, QMIX ensures that a global $\arg\max$ performed on Q results in the same vector as obtained upon performing individual $\arg\max$ operations on each Q_i. The statement is mathematically expressed in Equation 10.

$$\arg\max_u Q(u, s) = \begin{pmatrix} \arg\max_{u^{(1)}} Q_1(u^{(1)}, z^{(1)}; \theta) \\ \arg\max_{u^{(2)}} Q_2(u^{(2)}, z^{(2)}; \theta) \\ \vdots \\ \arg\max_{u^{(n)}} Q_n(u^{(n)}, z^{(n)}; \theta) \end{pmatrix} \tag{10}$$

QMIX approximates each $Q_{(i)}(u^{(i)}, z^{(i)}; \theta)$ using a neural network when selecting its actions. During its updates, QMIX makes use of a *mixer-network* f as a neural network. The *mixer-network* f serves as a mapping from agents' individual action-values to the global action-values. This is mathematically expressed in Equation 11.

$$\begin{aligned} Q(u, s) = f(Q_1(u^{(1)}, z^{(1)}; \theta), ...Q_n(u^{(n)}, z^{(n)}; \theta)), \\ s.t.\ f : Q_i(u^{(i)}, z^{(i)}; \theta) \to Q(u, s; \theta),\ \forall i \in N \end{aligned} \tag{11}$$

Monotonicity constraints in the *mixer-network* are enforced by making use of the absolute *tanh* operation as the suitable choice of non-linearity in units of the neural network. Additionally, the weight constants of the network are conditioned on the global state s_t. State dependence of $Q_{(i)}(u^{(i)}, z^{(i)}; \theta)$ relaxes the monotonicity constraint on $Q(u, s; \theta)$ and allows the full state information in the joint action-value estimates. This allows the formulation of the QMIX objective based on the Q-learning objective in Equation 3 and the *mixer-network* formulation in Equation 11. The objective is mathematically expressed in Equation 12 where $Q(u_t, s_t; \theta)$ is the global action-value function estimated using the *mixer-network* in Equation 11.

$$L(\theta) = \mathbb{E}_{b \sim R}[(r + \gamma \max_{u_{t+1} \in A} Q(u_{t+1}, s_{t+1}; \theta^-) - Q(u_t, s_t; \theta))^2] \tag{12}$$

Monotonic value factorization of QMIX allows a generalized decomposition which is capable of approximating richer decentralized policies.

Additionally, QMIX is scalable in the number of agents as a result of the centralized *mixer-network* which retains global Q-value information in the individual Q-value estimates. However, the QMIX scheme depicts sub-optimal robustness in the case of fast-paced dynamics [51] which hurts convergence in cases of challenging tasks.

4.2.4 QMIX-SMiRL

5 Tackling Stochastic Dynamics

In this section we introduce the novel objective for surprise minimization. The motivation behind surprise minimization in multi-agent learning stems from stochasticity in states leading to estimation biases among agents in the case of partially-observed settings. The proposed objective aims to address these challenges by making use of an energy-based scheme which takes into account observations of each agent in the multi-agent system.

Firstly, we formulate the energy-based objective consisting of surprise as a function of states s, joint actions u and deviation σ within states for each agent a. We call this function as the surprise value function $V^a_{surp}(s,u,\sigma)$ which serves as a mapping from agent and environment dynamics to surprise. We then define an energy operator presented in Equation 13 which sums the free energy across all agents.

$$\mathcal{T}V^a_{surp}(s,u,\sigma) = \log\sum_{a=1}^{N}\exp\left(V^a_{surp}(s,u,\sigma)\right) \tag{13}$$

We make use of the Mellowmax operator [53] as our energy operator. The energy operator is similar to the SQL energy formulation [54] where the energy across different actions is evaluated. In our case, inference is carried out across all agents with actions as prior variables. However, in the special case of using an energy-based model as a Q-function, the objective reduces to the SQL objective..

Our choice of the energy operator is based on its unique mathematical properties which result in better convergence. Of these properties, the most useful result is that the energy operator forms a contraction on the surprise value function indicating a guaranteed minimization of surprise within agents. This is formally stated in Theorem 1.

Theorem 1. *Given a surprise value function $V^a_{surp}(s,u,\sigma)\forall a\in N$, the energy operator $\mathcal{T}V^a_{surp}(s,u,\sigma) = \log\sum_{a=1}^{N}\exp\left(V^a_{surp}(s,u,\sigma)\right)$ forms a contraction on $V^a_{surp}(s,u,\sigma)$.*

Proof. We first define a norm on surprise values $||V_1 - V_2|| \equiv \underset{s,u,\sigma}{max}|V_1(s,u,\sigma) - V_2(s,u,\sigma)|$. Suppose $\epsilon = ||V_1 - V_2||$,

$$\begin{aligned}
\log\sum_{a=1}^{N}\exp\left(V_1(s,u,\sigma)\right) &\leq \log\sum_{a=1}^{N}\exp\left(V_2(s,u,\sigma)+\epsilon\right)\\
= \log\sum_{a=1}^{N}\exp\left(V_1(s,u,\sigma)\right) &\leq \log\exp\left(\epsilon\right)\sum_{a=1}^{N}\exp\left(V_2(s,u,\sigma)\right)\\
= \log\sum_{a=1}^{N}\exp\left(V_1(s,u,\sigma)\right) &\leq \epsilon + \log\sum_{a=1}^{N}\exp\left(V_2(s,u,\sigma)\right)\\
= \log\sum_{a=1}^{N}\exp\left(V_1(s,u,\sigma)\right) - \log\sum_{a=1}^{N}\exp\left(V_2(s,u,\sigma)\right) &\leq ||V_1 - V_2||
\end{aligned} \tag{14}$$

Similarly, using ϵ with $\log\sum_{a=1}^{N}\exp\left(V_1(s,u,\sigma)\right)$,

$$\begin{aligned}
\log\sum_{a=1}^{N}\exp\left(V_1(s,u,\sigma)+\epsilon\right) &\geq \log\sum_{a=1}^{N}\exp\left(V_2(s,u,\sigma)\right)\\
= \log\exp\left(\epsilon\right)\sum_{a=1}^{N}\exp\left(V_1(s,u,\sigma)\right) &\geq \log\sum_{a=1}^{N}\exp\left(V_2(s,u,\sigma)\right)\\
= \epsilon + \log\sum_{a=1}^{N}\exp\left(V_1(s,u,\sigma)\right) &\geq \log\sum_{a=1}^{N}\exp\left(V_2(s,u,\sigma)\right)\\
= ||V_1 - V_2|| &\geq \log\sum_{a=1}^{N}\exp\left(V_2(s,u,\sigma)\right) - \log\sum_{a=1}^{N}\exp\left(V_1(s,u,\sigma)\right)
\end{aligned} \tag{15}$$

Results in Equation 14 and Equation 15 prove that the energy operation is a contraction. □

The energy-based surprise minimization objective can then be formulated by simply adding the approximated energy-based surprise to the initial Bellman objective [9] of QMIX as expressed below.

$$L(\theta) = \underset{b\sim R}{\mathbb{E}}[\frac{1}{2}(y - (Q(u,s;\theta) + \beta\log\sum_{a=1}^{N}\exp\left(V^a_{surp}(s,u,\sigma)\right)))^2]$$

where,

$$y = r + \gamma\underset{u'}{\max}Q(u',s';\theta^-) + \beta\log\sum_{a=1}^{N}\exp\left(V^a_{surp}(s',u',\sigma')\right) \tag{16}$$

This leads to the formulation of the objective,

$$\begin{aligned}
L(\theta) = \underset{b\sim R}{\mathbb{E}}[\frac{1}{2}(r + \gamma\underset{u'}{\max}Q(u',s';\theta^-)\\
+ \beta\log\frac{\sum_{a=1}^{N}\exp\left(V^a_{surp}(s',u',\sigma')\right)}{\sum_{a=1}^{N}\exp\left(V^a_{surp}(s,u,\sigma)\right)} - Q(u,s;\theta))^2]
\end{aligned} \tag{17}$$

$$L(\theta) = \underset{b\sim R}{\mathbb{E}}[\frac{1}{2}(r + \gamma\underset{u'}{\max}Q(u',s';\theta^-) + \beta E - Q(u,s;\theta))^2] \tag{18}$$

Here, E is defined as the surprise ratio with β as a temperature parameter and $\sigma^{'}$ as the deviation among next states in the batch. The surprise value function is approximated by a universal function approximator (in our case a neural network) with its parameters as ϕ. $V_a(s^{'}, u^{'}, \sigma^{'})$ is expressed as the negative free energy and $\sum_{a=1}^{N} \exp{(V_a(s, u, \sigma))}$ the partition function. Alternatively, $V_a(s, u, \sigma)$ can be formulated as the negative free energy with $\sum_{a=1}^{N} \exp{(V_a(s^{'}, u^{'}, \sigma^{'}))}$ as the partition function. The objective incorporates the minimization of surprise across all agents as minimizing the energy in stochastic states. Such a formulation of surprise acts as intrinsic motivation [55] and at the same time provides robustness to multi-agent behavior. Furthermore, the energy formulation in the form of energy ratio E is a suitable choice as it guarantees convergence to minimum surprise at optimal policy π^*. This is formally expressed in Theorem 2.

Theorem 2. *Upon agent's convergence to an optimal policy π^*, total energy of π^*, expressed by E^* will reach a thermal equilibrium consisting of minimum surprise among consecutive states s and $s^{'}$.*

Proof. We begin by initializing a set of M policies $\{\pi_1, \pi_2 ..., \pi_M\}$ having energy ratios $\{E_1, E_2 ..., E_M\}$. Consider a policy π_1 with surprise value function V_1. E_1 can then be expressed as

$$E_1 = \log[\frac{\sum_{a=1}^{N} \exp{(V_1^a(s^{'}, u^{'}, \sigma^{'}))}}{\sum_{a=1}^{N} \exp{(V_1^a(s, u, \sigma))}}]$$

Assuming a constant surprise between s and $s^{'}$, we can express $V_1^a(s^{'}, u^{'}, \sigma^{'}) = V_1^a(s, u, \sigma) + \zeta_1$ where ζ_1 is a constant. Using this expression in E_1 we get,

$$E_1 = \log[\frac{\sum_{a=1}^{N} \exp{(V_1^a(s, u, \sigma) + \zeta_1)}}{\sum_{a=1}^{N} \exp{(V_1^a(s, u, \sigma))}}]$$
$$E_1 = \log[\frac{\exp{(\zeta_1)} \sum_{a=1}^{N} \exp{(V_1^a(s, u, \sigma))}}{\sum_{a=1}^{N} \exp{(V_1^a(s, u, \sigma))}}]$$
$$E_1 = \zeta_1$$

Similarly, $E_2 = \zeta_2$,$E_3 = \zeta_3$...,$E_M = \zeta_M$. Thus, the energy residing in policy π is proportional to the surprise between consecutive states s and $s^{'}$. Clearly, an optimal policy π^* is the one with minimum surprise. Mathematically,

$$\pi^* \geq \pi_1, \pi_2 ..., \pi_M \implies \zeta^* \leq \zeta_1, \zeta_2 ..., \zeta_M$$
$$= \pi^* \geq \pi_1, \pi_2 ..., \pi_M \implies E^* \leq E_1, E_2 ..., E_M$$

Thus, proving that the optimal policy consists of minimum surprise at thermal equilibrium. □

6 Experiments

6.1 The StarCraft II Benchmark

6.2 Performance

6.3 Stochastic Dynamics

7 Conclusion

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[2] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[3] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model, 2019.

[4] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[6] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. Starcraft ii: A new challenge for reinforcement learning, 2017.

[7] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI 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.

[8] Oriol Vinyals, Igor Babuschkin, Wojciech Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John Agapiou, Max Jaderberg, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575, 11 2019.

[9] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2018.

[10] Gonçalo Neto. From single-agent to multi-agent reinforcement learning: Foundational concepts and methods. *Learning theory course*, 2005.

[11] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

[12] Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere. Game theory and multi-agent reinforcement learning. In *Reinforcement Learning*, pages 441–470. Springer, 2012.

[13] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635*, 2019.

[14] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients, 2017.

[15] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari, 2019.

[16] Glen Berseth, Daniel Geng, Coline Devin, Dinesh Jayaraman, Chelsea Finn, and Sergey Levine. Smirl: Surprise minimizing rl in entropic environments. 2019.

[17] Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning, 2017.

[18] Luis Macedo, Rainer Reisezein, and Amilcar Cardoso. Modeling forms of surprise in artificial agents: empirical and theoretical study of surprise functions. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 26, 2004.

[19] Jerry Zikun Chen. Reinforcement learning generalization with surprise minimization, 2020.

[20] Luis Macedo and Amilcar Cardoso. The role of surprise, curiosity and hunger on exploration of unknown environments populated with entities. In *2005 portuguese conference on artificial intelligence*, 2005.

[21] Drew Fudenberg and David Levine. *The Theory of Learning in Games*, volume 1. The MIT Press, 1 edition, 1998.

[22] Michel Benaim and Morris W Hirsch. Learning processes, mixed equilibria and dynamical systems. *Games and Economic Behavior*, 29:36–72, 1999.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.

[24] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Int. Res.*, page 253–279, 2013.

[25] Hado V. Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems 23*. 2010.

[26] Sebastian B Thrun. Efficient exploration in reinforcement learning. 1992.

[27] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[28] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 2016.

[29] Tuomas Haarnoja. *Acquiring Diverse Robot Skills via Maximum Entropy Deep Reinforcement Learning*. PhD thesis, UC Berkeley, 2018.

[30] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 2085–2087, 2018.

[31] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML 2018: Proceedings of the Thirty-Fifth International Conference on Machine Learning*, 2018.

[32] Jianye Hao, Dongping Huang, Yi Cai, and Ho-Fung Leung. Reinforcement social learning of coordination in networked cooperative multiagent systems. In *AAAI workshop on multiagent interaction without prior coordination (MIPC 2014)*, 2014.

[33] Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *ICML*, volume 2, pages 227–234, 2002.

[34] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *In Proceedings of the Tenth International Conference on Machine Learning*, 1993.

[35] Jakob N Foerster. *Deep multi-agent reinforcement learning*. PhD thesis, University of Oxford, 2018.

[36] 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.

[37] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 2961–2970. PMLR, 2019.

[38] Rose E Wang, Michael Everett, and Jonathan P How. R-maddpg for partially observable environments and limited communication. *arXiv preprint arXiv:2002.06684*, 2020.

[39] Johannes Ackermann, Volker Gabler, Takayuki Osa, and Masashi Sugiyama. Reducing overestimation bias in multi-agent domains using double centralized critics. *arXiv preprint arXiv:1910.01465*, 2019.

[40] Xueguang Lyu and Christopher Amato. Likelihood quantile networks for coordinating multi-agent reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020.

[41] Hado V Hasselt. Double q-learning. In *Advances in neural information processing systems*, pages 2613–2621, 2010.

[42] Zipeng Fu, Qingqing Zhao, and Weinan Zhang. Reducing overestimation in value mixing for cooperative deep multi-agent reinforcement learning. *ICAART*, 2020.

[43] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation, 2020.

[44] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 2020.

[45] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

[46] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[47] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[48] Wei Ren, Randal W Beard, and Ella M Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the 2005, American Control Conference, 2005.*, 2005.

[49] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. In *Innovations in multi-agent systems and applications-1*. 2010.

[50] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1), 1950.

[51] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge, 2019.

[52] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, 2016.

[53] Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, 2017.

[54] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

[55] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.