Behavioural Analysis of Independent Value-Based Learning in Non-cooperative Games

Yang Li[a], Marco Perez Hernandez[a] and Mehmet Emin Aydin[a]

[a] University of the West of England, Frenchay Campus, Coldharbour Lane, Bristol, UK

ARTICLE HISTORY
Compiled June 7, 2025

ABSTRACT
Multi-agent reinforcement learning has received a lot of updates in cooperative games recently. However, research in general non-cooperative games is currently lagging behind. Independent value-based algorithms such as independent Q-learning have been one of the most widely studied methods in the field due to their simplicity and scalability. Some research has shown that independent algorithms can heavily outperform other approaches in some cooperative scenarios. In this paper, we discuss the feasibility of these algorithms in non-cooperative settings. As a result, we explain the conditions that a game must satisfy for the algorithms to work. We further propose the game Food Chain that simulates an ecosystem and is designed to challenge the algorithms in a way aligned with our analysis. Our results show that independent value-based learning algorithms are guaranteed to fail in some non-cooperative games but are more likely to work in cooperative games.

KEYWORDS
Multi-agent; reinforcement learning; independent value-based learning; game theory; non-cooperative games; competitive games; Nash equilibrium

1. Introduction

Recent multi-agent reinforcement learning (MARL) research has mostly focused on fully cooperative games [1–3] where there is no conflict between agents as they share a common reward. When agents are allowed to share information, we can apply single-agent reinforcement learning (RL) algorithms to a fully cooperative game where the collective agent tries to find the optimal joint action, given the joint observation, although the problem is still challenging in practice because of what the computational complexity of the joint action brings. In a traditional single-agent setting, the agent needs to learn which action gives the best reward. A Pareto-optimal pure strategy is guaranteed to exist in fully cooperative finite games, meaning that the agents can always deterministically choose the optimal action that has the highest payoff.

Non-cooperative environments, however, involve conflicts that make learning more complicated. Agents may belong to different parties that would like to have private control to protect their own interests. Rational and self-interested agents will have to fight for what is best for them, as increasing one agent's reward might lead to the decline of other's gains. We can no longer intuitively define the solution for a game

CONTACT Yang Li. Email: yang4.li@uwe.ac.uk

as the optimal strategy resulting in the highest reward. One of the possible solution concepts is the Nash equilibrium (NE). NEs describe the stable points in games, which are associated with the convergence to optimal behaviour of agents.

Many real-world problems can be modelled as non-cooperative games, such as business competition, political campaigns, stock trading, bidding in auctions, cyber security, and traffic routing. For example, if we omit the long-term economic growth, the stock market is often considered a 0-sum fully competitive game where one person's profit is another person's loss. Another example is cyber security, which must study the conflicts between attackers and defenders. These problems feature the competitive nature of games because they share one thing in common, parties with conflicting interests. Substantial research has been done on MARL in fully cooperative games, and non-cooperative games have received less attention [3,4].

Some studies have explored the application of MARL to solve games. One of which is independent value-based learning (IVBL). IVBL is a family of algorithms that models the multi-agent environments as single-agent environments, and they compute for each agent the expected reward of each action under each state. IVBL is simple and highly scalable because it is decentralized. It has been adopted for both cooperative [5–8] and non-cooperative tasks [9–12]. This paper analyses the performance of IVBL in non-cooperative finite games. The analysis considers characteristics of the action space and the NEs of the game, combining both formal analysis and experimental evaluation in selected two-player non-cooperative games. The results show NEs with uniformly random policies is a necessary condition for stability convergence of IVBL algorithms in non-cooperative environments. These condition highly restricts the wider applicability of IVBL.

The main contributions of this work include the following points:

- Behavioural analysis identifying necessary conditions for a stable convergence of IVBL which can drive the choice of algorithms in non-cooperative finite games.
- Introducing a non-cooperative game that has the characteristics which challenge the agents from a game theory perspective aligned with our analysis.
- Experimental evaluation of IVBL algorithms in selected two-player scenarios in the game proposed above.

The remainder of the paper is organized as follows: The next section provides the literature review on the key features of non-cooperative games and the most popular MARL algorithms that is related to our work. We then state the formal definitions of games and other concepts discussed in this paper in Section 3. Section 4 analyses the stability of IVBL in non-cooperative games. To verify our analysis with experimental results, a new game called Food Chain is introduced in Section 5. The game will have the key features discussed in our analysis that bring specific challenges to MARL agents. Then, we present the design as well as the results of our experiments and relevant discussions in Section 6. Finally, the drawn conclusions pulled in the last section.

2. Background and Literature Review

This literature review focuses on key features of non-cooperative games, as well as the MARL approaches to solve these games.

2.1. Non-Cooperative Games

In single-agent or fully cooperative environments, there is one reward function for all agents. Thus, the agents only need to learn which (joint) action has the highest payoff. In non-cooperative games, there may be no optimal joint action that satisfy all agents. Several problem solving concepts/approaches for non-cooperative games are discussed in game theory [13]. These include Pareto optimum, Nash equilibrium, and correlated equilibrium. Nash Equilibrium (NE), proposed by John Nash, is a renown approach for non-cooperative games [14]. NE offers a framework to characterise and analyse stability in games and describes a stability condition with which no player can benefit from deviating from their current strategy independently. The stability is associated to the optimal outcome in non-cooperative games and hence the objective agents aim for. Generally, there may only be a mixed NE, meaning that players must adopt randomized action-selection policy consisting of probability distributions over the action space. This is the mixed policy problem.

Many fully competitive games are studied in the MARL community such as Pommerman [15], Go [16], StarCraft II [17], and Dota 2 [18]. There are also some general non-cooperative environments such as hide-and-seek [19], Neural MMO [20]. PettingZoo and Melting Pot [21,22] are two integrated MARL environments including many cooperative and non-cooperative games. Kopazs et al. [12] studied the non-cooperative game predator-and-prey in the PettingZoo library, which is closely related to the game defined in this work. However, we lack a standard benchmark for MARL algorithms in non-cooperative games due to the difficulty of measuring agents' performance in general non-cooperative games. In cooperative games, the reward simply increases as the agents do better at the game. However, this is not the case in non-cooperative games where increasing one agent's outcome often results in the drop of another agent's return. FightLadder [4] used the Elo rating system to test MARL algorithms but they were restricted to two-player zero-sum fighting games.

2.2. MARL Algorithms

Independent learning (IL) algorithms that are carried over from the traditional single-agent RL domain treat other agents as part of the environment. The agents are essentially unaware of the existence of one another. IL algorithms are popular for handling many complex problems [2,23–25]. Among these, independent value-based learning (IVBL) algorithms, such as the classic independent Q-learning (IQL) [26], learn the expected long-term cumulative return (the Q-value) of each action at each state through sampling. IVBL algorithms including IQL are often considered as baseline and natural choice in a wide range of MARL studies because of their simplicity and scalability. For example, they were studied in various cooperative environments [5–8]. The problem is that IVBL agents only help produce the average reward of each action for each agent, but do not pay attention on what the action-selection policy. This is not a problem in single-agent or cooperative environments as we can simply pick the action with the highest return. Some studies addressed this issue in non-cooperative games by simply choosing the action with the highest empirical return [9,10,12] and observed reasonably good outcome. Kopacz et al [12] has examined IVBL in the non-cooperative game predator-and-prey. Both the algorithm and the game are closely related to our work. Other researchers [11] tried to address this problem using the softmax function given in Equation (1) to calculate a policy based on the payoff. Softmax takes a vector as input and computes a probability distribution over the original vector entries, such that the higher the original value, the higher the probability.

$$\mathbf{softmax}([x_1, ..., x_n]) = [\frac{e^{x_1}}{\sum_i e^{x_i}}, ..., \frac{e^{x_n}}{\sum_i e^{x_i}}] \tag{1}$$

The latter approach emphasis the action with higher empirical return. However, these approaches have a huge problem that guarantees failure of learning in a wide range of games. We will take a closer look in the analysis section.

Unlike IL, joint action learning (JAL) algorithms allow the agent to gain access to the information processed by all other agents, such as their choices of actions, so agents can learn the expected return of joint actions. Minimax Q-learning [27] is a JAL algorithm that can be applied to two-player zero-sum games. It computes local minimax solutions for each state based on the learned joint Q-values. Nash Q-learning [28] is a similar algorithm that computes the local NE instead. It converges to a NE under highly strict assumptions that reduce the applicability of the approach to a limited set of games. Correlated Q-learning [29] computes a correlated equilibrium and has no known convergence guarantees. Zinkevich et al [30] showed that JAL cannot converge in certain types of ill-conditioned games. Centralized learning requires full knowledge of the action spaces of all agents, which may not always be available if we are dealing with real-world problems. Another major disadvantage is that the number of joint actions is the number of possible combinations of the individual actions. This introduces combinatorics to the computational complexity of our algorithms, making them unscalable.

In contrast to the value-based approach, policy-based algorithms, such as the policy gradient methods, work around this issue by learning the policies directly, without putting an estimation on the individual action return. Policy gradient methods optimize a parametrized policy function using gradient decent to maximize the return. An advantage of policy gradient methods is that they naturally work on continuous action spaces. The actor-critic algorithms are a family of RL algorithm which trains a parametrized policy called the actor and a state-value function called the critic to estimate the policy gradient. Soft actor-critic (SAC) [31] has become one of the most popular algorithms recently. Its multi-agent variant (MASAC) has been considered the state-of-the-art (SOTA) baseline [1,32]. There is also naturally an independent version (ISAC) for MARL. The proximal policy optimization algorithm (PPO) [33] was based on actor-critic as well and was extended to multi-agent environments by some researchers [1,34] (MAPPO). Surprisingly, the IL version of PPO (IPPO) has proven to substantially outperform MAPPO and other algorithms in some cooperative tasks in StarCraft II [24]. MADDPG [35] is another policy gradient algorithm that is often considered SOTA. Because it only works for continuous action spaces, we do not discuss them further in the context of finite games.

Other notable MARL algorithms that have shown good performance include VDN [36] and its improved version QMIX [37]. The weakness is that they can only work for fully cooperative games where agents share a common reward.

2.3. Summary

As shown in this review, the issue of deriving the policy from the Q-values learned by IVBL agents in non-cooperative games has not been fully addressed by the MARL community, [9–12] made progress in this direction, but they did not analyse the behaviour of the agents and identify the flaws from a game theory point of view.

3. MARL Preliminaries

In this section, we provide preliminaries including the formal definition of a game and NE.

3.1. Stochastic Game

A game is formally defined by a tuple:
$G = \langle N, S, S_0, S_t, \{A_1, \dots, A_n\}, \{R_1, \dots, R_n\}, T\rangle$

- $N = \{1, ..., n\}$: the set of agents, numbered from 1 to n.
- S: the game's state space.
- $S_0 \subset S$: the set of possible initial states.
- $S_t \subset S$: the set of terminal states.
- $\{A_1, \dots, A_n\}$: one action space for each agent. Let $a_i \in A_i$ be an action of agent i, and $a^n = \langle a_1, \dots, a_n\rangle \in A^n$ denote a joint action in the joint action space.
- $\{R_1, \dots, R_n\}$: one reward function for each agent. $R_i : S \times A^n \times S \to \mathbb{R}$.
- T: the state transition function. $T : S \times A^n \times S \to [0, 1]$. It defines the probability of the game transitions from one state to another given the joint action. It must satisfy the axiom of probability:

$$\forall s \in S, a^n \in A^n : \sum_{s' \in S} T(s, a^n, s') = 1 \tag{2}$$

Starting from a randomly chosen $s_0 \in S_0$, at each time step $t \in \mathbb{N}$ with the current state s_t, each agent observes the current state s_t and chooses an action forming a joint action $a^n = \langle a_1, \dots, a_n\rangle$. The game then moves into the next state s_{t+1}, randomly sampled from the distribution P and a reward R_i is assigned to each agent. If $s_t \in S_t$, the game ends. Otherwise, the game continues in the next time step.

The way a player plays the game defines a policy (strategy) as a function that computes the probability of selecting each action given the current state, $\pi : S \times A^n \to [0, 1]$. A strategy is said to be pure if it is deterministic ($\pi : S \times A^n \to \{0, 1\}$). Otherwise, we call it a mixed strategy. A set of strategies, one for each player, is called a profile π^n. Given the profile, we can compute the expected reward for each player in the game. The goal is to find the optimal strategy for an agent, or even the optimal profile, such that expected cumulative rewards are maximized. Cooperative games are special cases where agents share the same reward. The optimal profile is the one that maximizes the common reward. Because they share the same reward function, the problem can be formulated as a joint player selecting a joint action and receives a reward, which makes cooperative games essentially single-agent games from a game theory perspective. In general non-cooperative games, however, it may not be possible to make everyone happy due to the conflicting interests. A more profound definition of optimum need to be discussed in the non-cooperative games section.

Formally, let $R_i^*(\pi^n)$ be the function to compute the expected reward of the i^{th} player given the profile. Let π_i be a policy of the i^{th} player and Π_i be the policy space of the i^{th} player. Let π^{-i} denote $\pi^n - \{\pi_i\}$, the set of strategies of all other players. A profile π^n is a NE if

$$\forall i \in N, \forall \pi_i \in \Pi_i : R_i^*(\{\pi_i\} \cup \pi^{-i}) \leqslant R_i^*(\pi^n) \tag{3}$$

Table 1. Rock-Paper-Scissors Reward Matrix

	r	p	s
r	0,0	-1,1	1,-1
p	1,-1	0,0	-1,1
s	-1,1	1,-1	0,0

Table 2. Biased Rock-Paper-Scissors Reward Matrix

	r	p	s
r	50,50	25,75	100,0
p	75,25	50,50	45,55
s	0,100	55,45	50,50

The reward matrix is used to define the reward for each player for each choice of action. The row player chooses a row and gets the first reward. Vice versa for the column player.

The idea is that NEs are the stable points in the game, because if an agent can benefit from switching strategy on its own, they would do so.

A game is finite if $|N|, |S|$ and all $|A_i|$ are finite, and the game always ends in a finite number of turns. John Nash proved that every finite game has at least one NE. To solve a game in the context of this paper means to find at least one NE. Games that are not finite can be ill-conditioned and have no NE. For example, a game where the player who names the greatest number wins has no best strategy. For the rest of the paper, we only refer to finite games. A normal form game has only two states, the starting state S_0 and the end state S_1. The game ends immediately after all agents pick an action. Thus, they can be represented using the reward matrix. In the book on game theory originally published in 1944 [38], Von Neumann and Morgenstern showed that finite games can be translated into normal form games. If we encapsulate the following states as a subgame, we end up with a reward matrix for the initial state with the rewards being the expected rewards of the subgames. We focus on normal form games because they are easier to work with for the analysis. Since they only have one state S_0 for agents to take actions, we simply write $\pi(a)$ instead of $\pi(S_0, a)$ to denote the probability of taking action a.

4. Stability of IVBL in Non-Cooperative Games

In this part of the paper, we show the feasibility of IVBL algorithms in non-cooperative games with behavioural analysis. We demonstrate the necessary conditions the game must satisfy in order to apply the IVBL algorithms.

To demonstrate the problem with competitive games, we study two simple normal form games. Consider the simple one-shot matrix game rock-paper-scissors (RPS) in Table 1. Each entry represents the reward for the row player and the column player, respectively. The only NE in the game is that two players play each of the three actions with 1/3 chance. If any player deviates from the NE, their expected reward does not increase, and the other player can exploit them by playing the action that counters their most frequent action. Table 2 defines a more complicated game called biased rock-paper-scissors. The only NE in the game is that both players adopt the following policy.

$$\pi = (P(r) = 0.0625, P(p) = 0.6250, P(s) = 0.3125) \tag{4}$$

A well-known direct consequence of Equation (3) in game theory is that at any NE π^n, given the policy set of the other players π^{i-1}, player i is indifferent from all actions that are chosen with a positive probability [14]. This means that the actions are split into two sets, the worthy actions that are chosen with a positive probability and the worthless ones that will never be piked. For example, a variation of RPS could incorporate an always-losing action, where the player receives negative reward once executing it, regardless of the opponent's action. Such an action should never be selected by the players. Additionally, the expected reward will be the same for every worthy action. In other words, the player can choose the worthy actions arbitrarily, given that other players commit to the NE. If we calculate the expected reward of each action in the biased rock-paper-scissors example where the opponent uses the NE strategy, we will see all three actions indeed have the same payoff. The above fact can be formally stated as the following. Assume π^n is a NE. Let $Q_i(a_j, \pi^{-i})$ be the expected reward of action $a_j \in A_i$ for agent i, given the rest of the profile π^{-i}. Let A_i^+ denote the set of worthy actions of agent i, namely $A_i^+ = \{a | a \in A_i \wedge \pi_i(a) > 0\}$. We have:

$$\forall a_j, a_k \in A_i^+, Q_i(a_j, \pi^{-i}) = Q_i(a_k, \pi^{-i}) \tag{5}$$

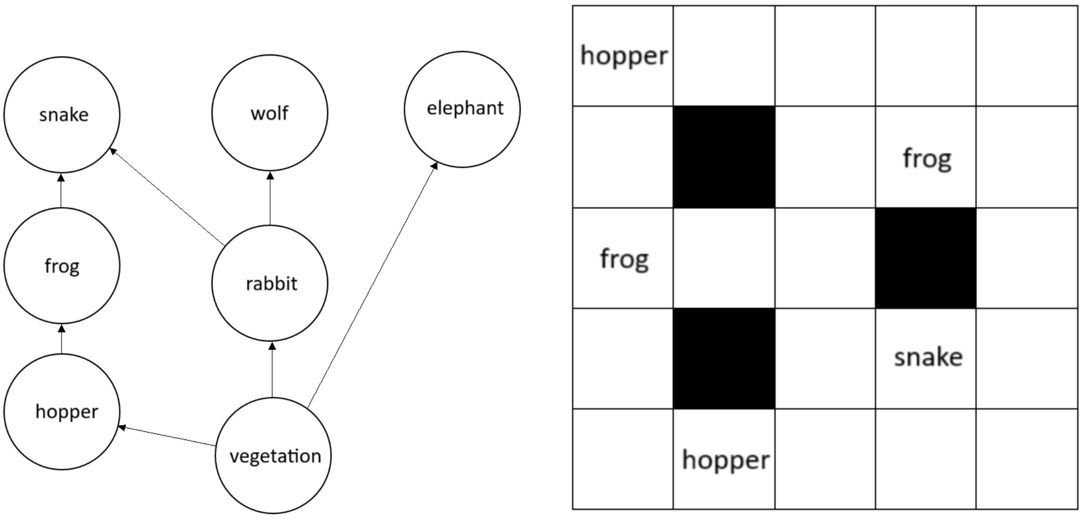
Suppose a IVBL algorithm can converge to a NE, then all worthy action's Q-values must converge to the same for an agent. It does not matter how the algorithm computes a probability distribution over the actions at the end, it cannot make any discriminations among those worthy actions based on equal statistics. Such an algorithm will output at best a uniform distribution over the worthy actions at a NE. This contradicts with the assumption that they converge to a NE unless the NE is full of uniformly random policies which is not the case in most of the games. We have shown that pure IVBL approaches does not work through proof by contradiction.

This sets a necessary condition for IVBL algorithms to work. IVBL algorithms can converge to a NE only if the NE consists of uniformly random policies over the set of all worthy actions. Note that this also includes any pure (deterministic) policies. It can be formally stated as follows. Let Π^n be the set of all possible profiles, and Π^{n*} be the subset containing all the NEs.

$$\exists \pi^n \in \Pi^{n*}, \forall \pi_i \in \pi^n, \forall a_j, a_k \in A_i^+, \pi_i(a_j) = \pi_i(a_k) \tag{6}$$

Because in cooperative games, there is no need to use mixed strategies, the NEs are essentially pure. This means that all cooperative games satisfy Equation (6), which explains why IVBL works in cooperative games. Additionally, according to Zermelo's theorem [39], perfect-information two-player finite asynchronous (players take turns) games such as chess or Go also satisfy the condition, because they also have a pure NE.

Then how did the researchers still find the appealing outcome of IVBL in some non-cooperative games [9–12]? The behaviour of IVBL can be highly problem-dependent, so we provide only high-level insights here. MARL problems studied in the real world can have large state/action spaces. For example, a 19 × 19 Go board can have roughly $3^{19\times19} = 10^{172}$ states. It may be too difficult for the agents to converge to a global NE because of the limited computational power. Because of the complexity of the environment mechanics itself, the competitiveness of the game is not going to play an enormous role before agents can reliably predict the state transitions. Picking the action with the best Q-value still gives reasonably good results, since at least the agents learned to interact with the environment.



(a) A Food Web

(b) The Food Chain Game

Figure 1. (a), A food web represents a simple ecosystem. The food web is a cluster of multiple food chains showing what-eats-what. The arrows point in the direction of the energy flow. (b), In the food chain game, an agent represents an individual animal of some type. They navigate on a playground trying to feed themselves while avoiding enemies.

5. The Food Chain Game

To verify our analysis, we test the algorithms in a game that can turn on and off the necessary condition given Equation (6) described in the previous section. We define the mechanics of the game Food Chain. An agent in the game represents an animal of some type. They can move around the map and feed themselves while trying to avoid enemies. Example game scenarios are shown in Figure 2 and Figure 3. The game simulates an ecosystem to a certain extend, by defining a food web structure. A food web is a cluster of food chains showing what-eats-what in the ecosystem as illustrated in Figure 1. The food web in the game is completely customizable and we can define an arbitrary number of species (agent types) and any number of agents of each type. This game is essentially a generalized version of the classic predator-and-prey problem. Recent research has tested IQL in the predator-and-prey game and found the algorithm to be seemingly performing well [12]. As we stated, these results could be misleading.

The game is played on a $w \times h$ grid. Each agent is placed in one of the tiles. There are also obstacles placed on the map. These could be randomly generated using a procedural algorithm or predefined. The state space S contains all possible configurations of the $w \times h$ game board. A game board configuration includes the current timestep t, as well as the locations of all the agents, vegetation and obstacles. All positions (x, y) must be unique and satisfy $1 \leq x \leq w, 1 \leq y \leq h$. Any state could be an initial state. A state is terminal if t reaches some preset maximum timestep. The action space $A = \{$**up, down, left, right, pass**$\}$ is the same for all agents. The state transition function maps the current state and the choices of actions to the next state by executing the agents' actions in a natural way. Agents choose their actions to move along the direction by one cell. Moves resulting in an illegal position will be regarded as pass. If multiple agents try to occupy the same cell, only a randomly selected agent will succeed and the rest will execute the pass action. If an agent is moving away from its current cell, then it is allowed for another agent to enter this cell without conflicts, unless two agents are trying to swap location (this rule prevents agents from passing through each other). The reward function works as follows. After the state transition, if a predator is adjacent to (i.e. within a Manhattan distance of 1) a prey, the prey dies and is removed from the game. Agents receive some negative reward (-1 by default) when they die. Each empty tile also has a chance to grow vegetation each turn. Vegetation is part of the food web and will be destroyed when an agent enter the tile. If the vegetation is a prey to the agent, it will be consumed (this is different from agents eating agents where they only need to be adjacent). Agents get reward (0.1 by default) when they eat. If multiple predators capture a prey at the same time, they must share the reward equally.

The size of the state space grows exponentially with respect to the size of the map and the number of agents. The number of possible states in a game with n agents playing on a board of k cells is lower bounded by $C\binom{k}{n}$. This computational complexity eliminates any tabular method, a brute-force approach that simply records the value of all possible state-action pairs in a table. Function approximation methods, such as deep neural networks or Monte Carlo methods should be used to solve the whole game [40]. The game is still simple enough to be human playable. The mechanics of the game (the state transition function) is straightforward, so that agents can focus on interactions. Because of the large state space, it is difficult for us to check whether the agents converged to a global NE since it is NP-Hard. However, the way predators hunt for preys often force agents to address the mixed policy problem, as we will explain in the next section with two example scenarios illustrated in Figure 2 and Figure 3. We can actually check whether the agents converge to NEs in those subcases. In summary:

- The navigation in the environment where agents are placed is simple enough, so the agents can focus on learning interactions instead of the state transitions, yet the game is still complex enough to be interesting as it blocks out brute-force algorithms.
- The game is extendable in many dimensions, the number of agents of each type, the number of species, the relationship between the species, and the size of the map.
- The game forces the agents to constantly address the mixed strategy problem.
- We can analyse the performance of the agents with game theory.

6. Experimental Evaluation

Too see how Food Chain can turn on/off the necessary condition (6) for IVBL to converge, we design two scenarios. The key difference between the two is that one satisfies the condition (6) and the other does not. We define a simple food web consisting of frogs and hoppers. Hoppers (represented by h) can consume vegetation (drawn as *) and will be eaten by frogs (f). Both scenarios will turn off random food generation.

6.1. Biased Rock-Paper-Scissors in Food Chain

This case violates condition (6) and forces the agents to address the mixed policy problem. The initial state of the scenario consists of a frog trying to hunt down a hopper as shown in Figure 2. We can see that in some cases the hopper can escape.

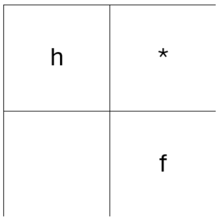


Figure 2. The initial state of Biased RPS in Food Chain. The frog (f) is trying to capture the hopper (h). The only NE in the game contains non-uniformly random mixed policies.

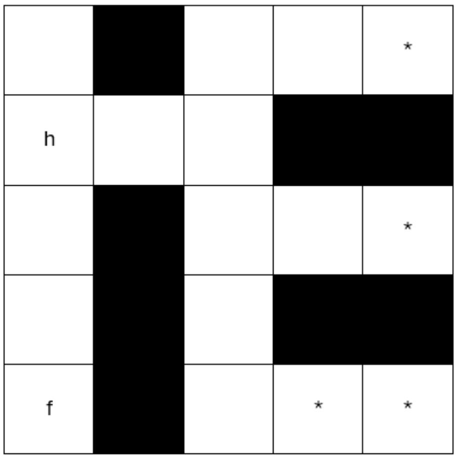


Figure 3. The initial state of Tree Maze Chase in Food Chain. The frog (f) is chasing the hooper (h) in the maze. There is an optimal pure (deterministic) policy for both players.

If they both pass, they will face the exact same situation again. Figure A1 in the appendix shows all possible next states. For simplicity, we set the max timestep to 1, so the game ends after one move. The scenario becomes a normal form mini game. To break the symmetry so that condition (6) is violated, we add food in one cell, making the scenario essentially biased RPS. We also raise the feeding reward for both agents to 0.5 to highlight the asymmetry. The reward matrix is shown in Table A1 in the appendix. We can verify that there is a unique NE in the game.

$$\pi_h = (P(r) = \frac{1}{3}, P(d) = \frac{1}{3}, P(p) = \frac{1}{3}) \tag{7}$$

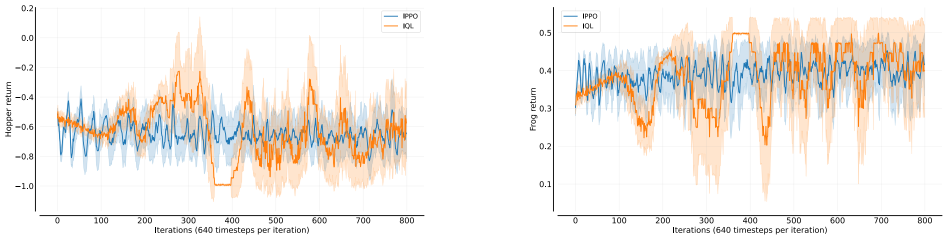
$$\pi_f = (P(l) = \frac{1}{13}, P(u) = \frac{6}{13}, P(p) = \frac{6}{13}) \tag{8}$$

The expected reward for the hooper and the frog should be $-7/13 \approx -0.538$ and $1/3 \approx 0.333$, respectively.

In the Food Chain game, because all agents have the same mobility, a single predator can only capture a prey by trapping them in a corner. Thus, any predator, unless working in a team, has to face this subcase at some point during a general Food Chain game played on an open ground. This means that the scenario represents a subgame which will show up frequently in a full game.

6.2. Tree Maze Chase in Food Chain

This scenario essentially turns off the mixed policy problem, so condition (6) is met. We set up the initial state of the game as shown in Figure 3 where a frog is chasing a hopper in a maze. This map resembles a tree structure (no cycles) which ensures that there is a pure NE in the game, because there is always an optimal action for both players. The hooper should go for the tunnel with more food and the frog should simply chase the hopper. This means the necessary condition (6) is satisfied. To make



(a) Biased RPS Hopper Rewards

(b) Biased RPS Frog Rewards

Figure 4. Biased RPS. IVBL (IQL) shows significantly more instability and does not converge to the NE.

sure that at most two food can be picked up by the hopper, we set the max timestep to 12. The optimal rewards for the hopper and the frog are -0.8 and 1, respectively.

6.3. Results and Discussion

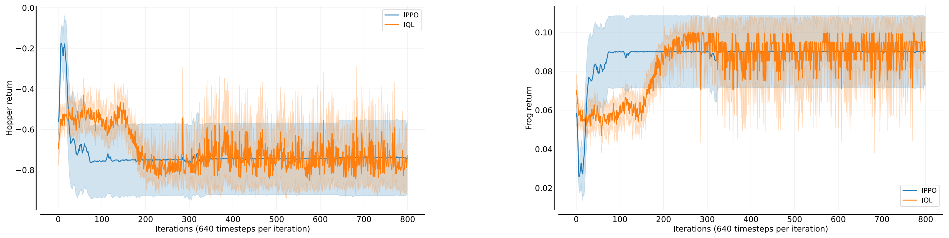
In both scenarios, we test the IVBL algorithm IQL and the independent policy-based algorithm IPPO. All experiments were run for 800 iterations. For each iteration, the algorithm collects 640 timesteps as training samples and train the neural network. After each training iteration, 50 evaluation games are run during the evaluation phase. The plots show the average reward for each agent during the evaluation phase. Each experiment is run 20 times to mitigate the effect of randomness and the graphs show the boots trapped 95% confidence interval. In total, each data point on the graph is the average result of $20 \times 50 = 1000$ evaluation games. More details of the parameters for the experiments can be found in Table B1 in the appendix.

Figure 4 demonstrates the performance of the algorithms in the biased RPS scenario where the necessary condition (6) for IVBL to converge is not met. The results show that the IVBL algorithm IQL failed to converge to a defined solution range and were struggling to stabilize itself. The hopper's reward fluctuates between -1 and -0.2, where the optimal value is -0.538. These value can be as far as 85% away from the desired amount. The frog reward oscillates in the range [0.2, 0.5], which is in a ±70% range from the actual NE (0.333). For comparison, the independent policy-based learning algorithm IPPO achieves a reward that stays more in the optimal range and is far more stable which indicates the agents are converging towards the NE.

In contrast, IVBL does significantly better in the tree maze scenario, as shown in Figure 5. This time, the game scenario satisfies the condition (6). Although still unstable, IQL can roughly reach the optimal reward at the end. The optimal rewards for the hopper and the frog are -0.8 and 0.1, respectively. Even in this case, the policy-based method still appears to be more reliable than IVBL.

7. Conclusion

In this work, we have analysed the stability of IVBL algorithms in non-cooperative games and tested their performance in specific game scenarios. A necessary condition (6) for IVBL algorithms to converge to a NE in finite games was identified. The condition requires the game to have a NE where every agent's policy evenly chooses their worthy actions (actions that are played with a positive probability). This explains



(a) Tree Maze Chase Hopper Rewards

(b) Tree Maze Chase Frog Rewards

Figure 5. Tree Maze Chase. Both algorithms quickly learned to play the game and tend to stay in the optimal reward range. Note that increasing frog reward reduces the hopper reward as the game is non-cooperative.

why we may observe satisfying results when applying IVBL to games with pure NEs including cooperative games. We further proposed the game Food Chain which has certain traits that can verify our analysis. The test results show that IVBL algorithms perform poorly in scenarios where the condition (6) is violated. Our work is limited to small two-player scenarios in the Food Chain game. There are still interesting characteristics of the game that may be explored in the future. Such as the extension in both the number of agents and the number of species in the food web.

Acknowledgements

The authors sincerely gratitude the creators of the MARL library BenchMARL [41], which was used to run the experiments.

Funding

This research was funded by the University of the West of England.

References

[1] Yu C, Velu A, Vinitsky E, et al. Benchmarking multi-agent deep reinforcement learning algorithms. In: Workshop in Conference on Neural Information Processing Systems (NeurIPS); 2020.

[2] Papoudakis G, Christianos F, Schäfer L, et al. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. arXiv preprint arXiv:200607869. 2020;.

[3] Zhu C, Dastani M, Wang S. A survey of multi-agent deep reinforcement learning with communication. Autonomous Agents and Multi-Agent Systems. 2024;38(1):4.

[4] Li W, Ding Z, Karten S, et al. Fightladder: A benchmark for competitive multi-agent reinforcement learning. arXiv preprint arXiv:240602081. 2024;.

[5] Foerster J, Nardelli N, Farquhar G, et al. Stabilising experience replay for deep multi-agent reinforcement learning. In: International conference on machine learning; PMLR; 2017. p. 1146–1155.

[6] Omidshafiei S, Pazis J, Amato C, et al. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In: International Conference on Machine Learning; PMLR; 2017. p. 2681–2690.

[7] Palmer G, Tuyls K, Bloembergen D, et al. Lenient multi-agent deep reinforcement learning. arXiv preprint arXiv:170704402. 2017;.

[8] Palmer G, Savani R, Tuyls K. Negative update intervals in deep multi-agent reinforcement learning. arXiv preprint arXiv:180905096. 2018;.

[9] Bjornsson Y, Finnsson H. Cadiaplayer: A simulation-based general game player. IEEE Transactions on Computational Intelligence and AI in Games. 2009;1(1):4–15.

[10] Jiang H, He H, Liu L, et al. Q-learning for non-cooperative channel access game of cognitive radio networks. In: 2018 International Joint Conference on Neural Networks (IJCNN); IEEE; 2018. p. 1–7.

[11] Qu Z, Pan Z, Chen Y, et al. A distributed control method for urban networks using multi-agent reinforcement learning based on regional mixed strategy nash-equilibrium. IEEE Access. 2020;8:19750–19766.

[12] Kopacz A, Csató L, Chira C. Evaluating cooperative-competitive dynamics with deep q-learning. Neurocomputing. 2023;550:126507.

[13] Albrecht SV, Christianos F, Schäfer L. Multi-agent reinforcement learning: Foundations and modern approaches. MIT Press; 2024. Chapter 4.

[14] Nash JF. Non-cooperative games. In: The foundations of price theory vol 4. Routledge; 2024. p. 329–340.

[15] Resnick C, Eldridge W, Ha D, et al. Pommerman: A multi-agent playground. arXiv preprint arXiv:180907124. 2018;.

[16] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge. nature. 2017;550(7676):354–359.

[17] Vinyals O, Babuschkin I, Czarnecki WM, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. nature. 2019;575(7782):350–354.

[18] Berner C, Brockman G, Chan B, et al. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:191206680. 2019;.

[19] Baker B, Kanitscheider I, Markov T, et al. Emergent tool use from multi-agent autocurricula. In: International conference on learning representations; 2019.

[20] Suarez J, Du Y, Zhu C, et al. The neural mmo platform for massively multiagent research. arXiv preprint arXiv:211007594. 2021;.

[21] Terry J, Black B, Grammel N, et al. Pettingzoo: Gym for multi-agent reinforcement learning. Advances in Neural Information Processing Systems. 2021;34:15032–15043.

[22] Leibo JZ, Dueñez-Guzman EA, Vezhnevets A, et al. Scalable evaluation of multi-agent reinforcement learning with melting pot. In: International conference on machine learning; PMLR; 2021. p. 6187–6199.

[23] Gupta JK, Egorov M, Kochenderfer M. Cooperative multi-agent control using deep reinforcement learning. In: Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16; Springer; 2017. p. 66–83.

[24] De Witt CS, Gupta T, Makoviichuk D, et al. Is independent learning all you need in the starcraft multi-agent challenge? arXiv preprint arXiv:201109533. 2020;.

[25] Palmer G. Independent learning approaches: Overcoming multi-agent learning pathologies in team-games. The University of Liverpool (United Kingdom); 2020.

[26] Tan M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the tenth international conference on machine learning; 1993. p. 330–337.

[27] Littman ML. Markov games as a framework for multi-agent reinforcement learning. In: Machine learning proceedings 1994. Elsevier; 1994. p. 157–163.

[28] Hu J, Wellman MP. Nash q-learning for general-sum stochastic games. Journal of machine learning research. 2003;4(Nov):1039–1069.

[29] Greenwald A, Hall K, Serrano R, et al. Correlated q-learning. In: ICML; Vol. 3; 2003. p. 242–249.

[30] Zinkevich M, Greenwald A, Littman M. Cyclic equilibria in markov games. Advances in neural information processing systems. 2005;18.

[31] Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: Off-policy maximum entropy deep

reinforcement learning with a stochastic actor. In: International conference on machine learning; Pmlr; 2018. p. 1861–1870.

[32] Zhang Q, Dong H, Pan W. Lyapunov-based reinforcement learning for decentralized multi-agent control. In: Distributed Artificial Intelligence: Second International Conference, DAI 2020, Nanjing, China, October 24–27, 2020, Proceedings 2; Springer; 2020. p. 55–68.

[33] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms. arXiv preprint arXiv:170706347. 2017;.

[34] Yu C, Velu A, Vinitsky E, et al. The surprising effectiveness of ppo in cooperative multi-agent games. Advances in neural information processing systems. 2022;35:24611–24624.

[35] Lowe R, Wu YI, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems. 2017;30.

[36] Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:170605296. 2017;.

[37] Rashid T, Samvelyan M, De Witt CS, et al. Monotonic value function factorisation for deep multi-agent reinforcement learning. Journal of Machine Learning Research. 2020; 21(178):1–51.

[38] Von Neumann J, Morgenstern O. Theory of games and economic behavior: 60th anniversary commemorative edition. In: Theory of games and economic behavior. Princeton university press; 2007.

[39] Schwalbe U, Walker P. Zermelo and the early history of game theory. Games and economic behavior. 2001;34(1):123–137.

[40] Silver D, Huang A, Maddison CJ, et al. Mastering the game of go with deep neural networks and tree search. nature. 2016;529(7587):484–489.

[41] Bettini M, Prorok A, Moens V. Benchmarl: Benchmarking multi-agent reinforcement learning. Journal of Machine Learning Research. 2024;25(217):1–10. Available from: http://jmlr.org/papers/v25/23-1612.html.

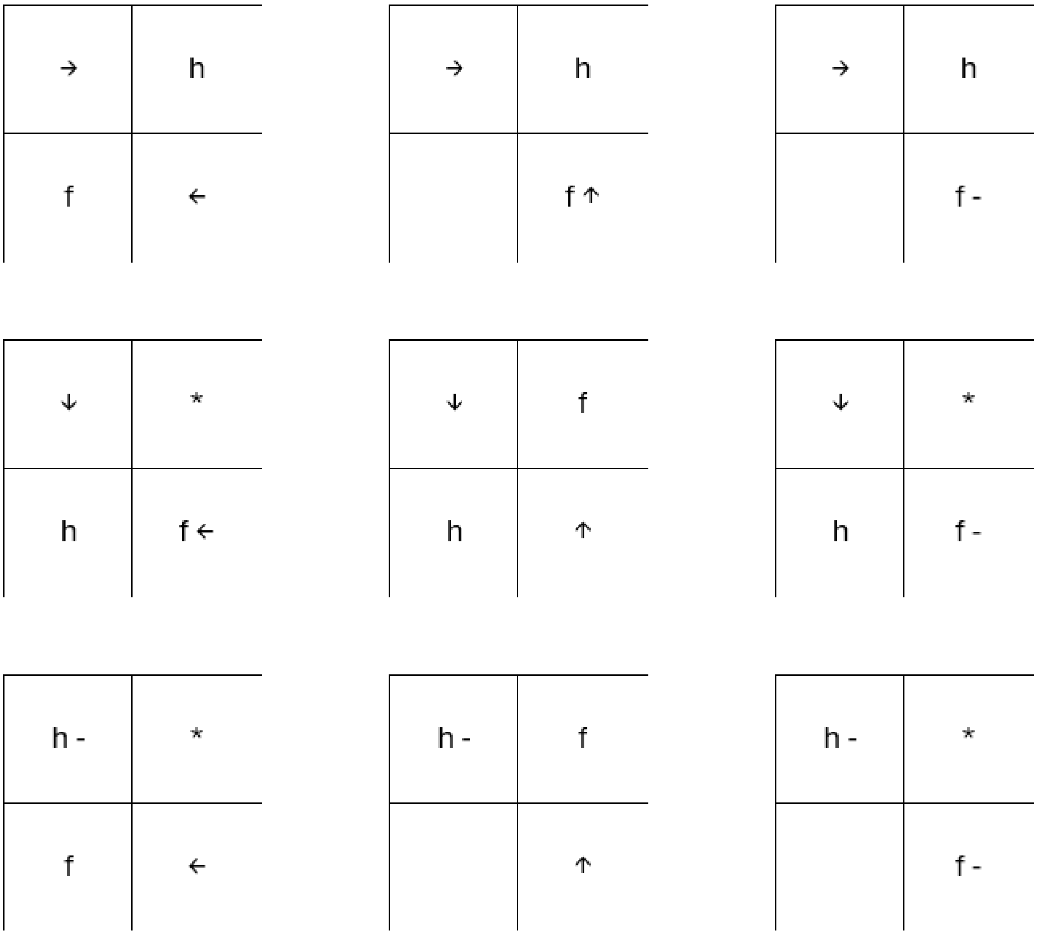


Figure A1. Possible Next States in Biased RPS in Food Chain (omitting illegal moves)

Appendix A. Biased RPS in Food Chain

Figure A1 shows all the possible next state of the Biased RPS scenario in Food Chain. Table A1 describes the reward matrix of the scenario (note that if both agents try to move into the food tile, the hopper only has a 50% chance to get food).

Appendix B. Experiments Parameters

Table A1. Biased RPS in Food Chain Reward Matrix

	left	up	pass
right	0.5,0	-0.75,0.5	-0.5,0.5
down	-1,0.5	0,0	-1,0.5
pass	-1,0.5	-1,0.5	0,0

Hopper is the row player.

Table B1. Experiments Parameters

game parameters		
	Biased RPS	Tree Maze Chase
size of the map	2×2	5×5
max timestep	1	12
feeding reward	0.5	0.1
death reward	-1	
food spawn chance	0	
number of agents	2	
agent types	2	
experiment parameters		
repetitions of each experiment	20	
number of iterations	800	
evaluation games after each iteration	50	
training hyper parameters		
	IPPO	IQL
on-policy minibatch optimization iterations[a]	10	
off-policy optimizer steps[b]		100
off-policy memory buffer size		20,000
frames collected for each iteration	640	
minibatch size	32	
discount factor γ	0.99	
learning rate α	5×10^{-5}	
the ε parameter of the adam optimizer	1×10^{-6}	

[a] The number of times collected frames will be split into minibatches and trained.

[b] The number of times a minibatch will be sampled from the memory buffer and trained over.