Multiagent Reinforcement Learning in Stochastic Games

Junling Hu and Michael P. Wellman
Artificial Intelligence Laboratory
University of Michigan
Ann Arbor, MI 48109-2110, USA
{junling, wellman}@umich.edu
http://ai.eecs.umich.edu/people/{junling,wellman}

Abstract

We adopt stochastic games as a general framework for dynamic noncooperative systems. This framework provides a way of describing the dynamic interactions of agents in terms of individuals' Markov decision processes. By studying this framework, we go beyond the common practice in the study of learning in games, which primarily focus on repeated games or extensive-form games. For stochastic games with incomplete information, we design a multiagent reinforcement learning method which allows agents to learn Nash equilibrium strategies. We show in both theory and experiments that this algorithm converges. From the viewpoint of machine learning research, our work helps to establish the theoretical foundation for applying reinforcement learning, originally defined for single-agent systems, to multiagent systems.

1 Introduction

The goal of an agent in a dynamic environment is to make optimal decisions over time. Learning servers such purpose by biasing the agent's action choices through information gathered over time. An agent can base its action choice on prediction of the environment or directly on the reward received from the environment. Reinforcement learning is a systematic method that associates an agent's action with its rewards.

Reinforcement learning is related to two other types of learning. In *supervised learning*, an agent observes input/output pairs as examples, and learns about the function mapping the inputs to the outputs. The values of the outputs are provided by the environment, which can be thought of as a supervisor or a teacher. Decision tree methods [15, 11] and neural network learning [6] fall into this category. In *unsupervised learning*, the agent is given a collection of input values but no output values. The agent has to find "regularity" in the inputs by itself. Clustering [5] and discovery [8] methods fall into this category. Unsupervised learning methods are widely applied under the popular name *data mining* nowadays. Reinforcement learning [19] falls in between supervised and unsupervised learning. In reinforcement learning, an agent does not receive input/output examples from the environment. But the agent receives rewards from the environment, and it can use the rewards as feedback (reinforcement) for its actions. In this sense, the agent is guided or supervised by the environment.

In reinforcement learning, an agent need not explicitly model the environment since its actions can be directly based on the rewards. Thus this learning method is particularly useful for the cases where agents have little knowledge of the environment.

An agent in a multiagent system may know little about others because information is distributed. Even when an agent has some prior information about others, the behavior of others may change over time because they are learning. It is therefore natural to apply reinforcement learning. We have seen such applications, for example, in robotic soccer [1, 18], the pursuit game [21, 3], and distributed team coordination [2].

In most cases, single-agent reinforcement learning methods are applied without much modification. Such an approach treats other agents as a part of the environment. There are two problems with this approach: First, the environment in this treatment is non-stationary: other agents are learning and changing their responses. But the convergence of single-agent reinforce-

ment learning is based on the assumption that the environment is stationary. Second, an agent who does not take into account of other agents may have worse performance than the one does. In Claus and Boutilier's study of a repeated coordination game [2], two types of learning agents are compared: the *individual learner* who uses the standard reinforcement learning method, and the *joint learner* who incorporate joint actions in reinforcement learning. The learning object is one-period's reward function. Claus and Boutilier show that the joint learner performs better because it uses extra information in the game. Even though their result is not conclusive because it concerns only one-period's decision while standard reinforcement learning problems concern sequential decisions, it is one of the first attempts to model the joint actions in reinforcement learning. Littman [9] implement agents who use minimax Q-learning method in 2-player zero-sum stochastic games. In minimax Q-learning, an agent chooses an action based on possible reactions from the other agent. Such agents are compared to the ones who use standard (single-agent) Q-learning method. He found that agents using minimax Q-learning are more likely to win.

We propose that a learning agent should explicitly consider other agents in the system. We formulate the multiagent reinforcement learning problem in the framework of general-sum stochastic games. This framework can be used to model a wide range of dynamic multiagent systems. By studying reinforcement learning in this framework and investigating the convergence of our learning algorithm, we want to provide a theoretical foundation for applying reinforcement learning to multiagent systems.

Our work is closely related to Littman's [9], which studies reinforcement learning in zero-sum stochastic games. In zero-sum games, one agent's gain is always the other agent's loss, thus agents have strictly opposite interests. In general-sum games, the payoffs of agents are not always correlated. The solution concept for general-sum games is Nash equilibrium. In a Nash equilibrium, each agent's choice is the best response to the other agents' choices. No agent can gain by unilateral deviation.

If the payoff structure and state transition probabilities are known to all the agents, we can solve for Nash equilibrium using a nonlinear programming method proposed by Filar and Vrieze [4]. We are interested in situations where agents have little information of other agents' payoff functions and the state transition probabilities. We show that a multiagent Q-learning algorithm can be designed, and it converges to the Nash equilibrium Q values (defined later in this chapter) under certain assumptions of the games dur-

ing learning. Our algorithm is designed for 2-player general-sum stochastic games, but can be extended to n-player general-sum cases.

We implement the multiagent Q-learning algorithm in a grid-world game, and show that agents can use this learning algorithm to learn stationary Nash equilibrium strategies. In our theory, the convergence of our learning algorithm relies on certain restrictions of the game structure during learning, which imply a unique Nash equilibrium value in each bimatrix game. In the experiments, we relax the restrictions and show that convergence is still possible when there are multiple Nash equilibria.

2 Single-Agent Reinforcement Learning

The concept of reinforcement learning originated from Thorndike's research on animals in 1911 [19]. Animals could learn to associate their actions with the rewards through trial and errors. Reinforcement learning was first formalized in the learning automata model in 1970s [13]. In early 1980s, Sutton and Barto [19] developed temporal-difference learning, which is another form of reinforcement learning. Further attention was drawn to reinforcement learning after Watkins and Dayan proposed Q-learning in 1992 [22], which established the connection between reinforcement learning and Markov decision processes. Our work is a further development of Q-learning. Originally, Q-learning is designed for single-agent systems. We define Q-learning in a general class of multi-agent systems. We also redefine the learning algorithm itself. For a comprehensive survey of reinforcement learning, see [7, 19].

Q-learning is defined in the framework of Markov decision process.

Definition 1 *A Markov Decision Process is a tuple* $< S, A, r, p >$*, where* S *is the discrete state space,* A *is the discrete action space,* $r : S \times A \to R$ *is the reward function, and* $p : S \times A \to \Delta$ *is the transition function, where* Δ *is the set of probability distributions over state space* S*.*

In a Markov decision process, the objective of the agent is to find a strategy (policy) π so as to maximize the expected sum of discounted rewards,

$$v(s, \pi) = \sum_{t=0}^{\infty} \beta^t E(r_t | \pi, s_0 = s), \tag{1}$$

where s_0 is the initial state, r_t is the reward at time t, and $\beta \in [0, 1)$ is the discount factor. We can rewrite Equation (1) as

$$v(s, \pi) = r(s, a_\pi) + \beta \sum_{s'} p(s'|s, a_\pi) v(s', \pi), \tag{2}$$

where a_π is the action dictated by policy π given initial state s. It has been proved that there exists an optimal policy π^* such that for any $s \in S$, the following equation holds:

$$v(s, \pi^*) = \max_a \left\{ r(s, a) + \beta \sum_{s'} p(s'|s, a) v(s', \pi^*) \right\}, \tag{3}$$

where $v(s, \pi^*)$ is called the *optimal value* for state s.

If the agent knows the reward function and the state transition function, it can solve for π^* by iterative search methods [14]. The learning problem arises when the agent does not know the reward function or the state transition probabilities. Now the agent needs to interact with the environment to find out its optimal policy. The agent can learn about the reward function and the state transition function, and then solve for its optimal policy using Equation (3). Such an approach is called *model-based reinforcement learning*. The agent can also directly learn about its optimal policy without knowing the reward function or the state transition function. Such an approach is called *model-free reinforcement learning*. One of the model-free reinforcement learning methods is Q-learning.

The basic idea of Q-learning is that we can define the right-hand side of (3) as

$$Q^*(s, a) = r(s, a) + \beta \sum_{s'} p(s'|s, a) v(s', \pi^*) \tag{4}$$

By this definition, $Q^*(s, a)$ is the total discounted reward attained by taking action a in state s and then following the optimal policy thereafter. Then by (3),

$$v(s, \pi^*) = \max_a Q^*(s, a). \tag{5}$$

If we know $Q^*(s, a)$, then the optimal policy π^* can be found, which is always taking an action so as to maximize $Q^*(s, a)$ under any state s.

In Q-learning, the agent starts with arbitrary initial values of $Q(s,a)$ for all $s \in S, a \in A$. At each time t, the agent chooses an action and observes its reward, r_t. The agent then updates its Q-values as follows:

$$Q_{t+1}(s,a) = (1-\alpha_t)Q_t(s,a) + \alpha_t[r_t + \beta \max_b Q_t(s_{t+1}, b)]. \tag{6}$$

where $\alpha_t \in [0,1)$ is the learning rate. The learning rate α_t needs to decay over time in order for the learning algorithm to converge. Watkins and Dayan [22] proved that sequence (6) converges to $Q^*(s,a)$ under the assumption that all states and actions have been visited infinitely often.

3 The stochastic game framework

Markov decision process (MDP) is a single agent decision problem. A natural extension of MDP to multiagent systems is stochastic games, which essentially are n-agent Markov decision processes. In this paper, we focus on 2-player stochastic games since they have been well studied.

3.1 Definition of stochastic games

Definition 2 *A 2-player stochastic game Γ is a 6-tuple $< S, A^1, A^2, r^1, r^2, p >$, where S is the discrete state space, A^k is the discrete action space of player k for $k = 1, 2$, $r^k : S \times A^1 \times A^2 \to R$ is the payoff function for player k, $p : S \times A^1 \times A^2 \to \Delta$ is the transition probability map, where Δ is the set of probability distributions over state space S.*

To have a closer look at a stochastic game, consider a process that is observable at discrete time points $t = 0, 1, 2, \ldots$. At each time point t, the state of the process is denoted by s_t. Assume s_t takes on values from the set S. The process is controlled by 2 decision makers, referred to as player 1 and player 2, respectively. In state s, each player independently chooses actions $a^1 \in A^1, a^2 \in A^2$ and receives rewards $r^1(s, a^1, a^2)$ and $r^2(s, a^1, a^2)$, respectively. When $r^1(s, a^1, a^2) + r^2(s, a^1, a^2) = 0$ for all s, a^1, a^2, the game is called *zero sum*. When the sum is not restricted to 0 or any constant, the game is called a *general-sum* game.

It is assumed that for every $s, s' \in S$, the transition from s to s' given that the players take actions $a^1 \in A^1$ and $a^2 \in A^2$, is independent of time.

That is, there exist stationary transition probabilities $p(s'|s, a^1, a^2)$ for all $t = 0, 1, 2, \dots$, satisfying the constraint

$$\sum_{s'=1}^{m} p(s'|s, a^1, a^2) = 1, \tag{7}$$

The objective of each player is to maximize a discounted sum of rewards. Let $\beta \in [0, 1)$ be the discount factor, let π^1 and π^2 be the strategies of players 1 and 2 respectively. For a given initial state s, the two players receive the following values from the game:

$$v^1(s, \pi^1, \pi^2) = \sum_{t=0}^{\infty} \beta^t E(r_t^1 | \pi^1, \pi^2, s_0 = s) \tag{8}$$

$$v^2(s, \pi^1, \pi^2) = \sum_{t=0}^{\infty} \beta^t E(r_t^2 | \pi^1, \pi^2, s_0 = s) \tag{9}$$

A strategy $\pi = (\pi_0, \dots, \pi_t, \dots)$ is defined over the whole course of the game. π_t is called the *decision rule* at time t. A strategy π is called a *stationary strategy* if $\pi_t = \bar{\pi}$ for all t, where the decision rule is fixed over time. π is called a *behavior strategy* if $\pi_t = f(h_t)$, where h_t is the history up to time t,

$$h_t = (s_0, a_0^1, a_0^2, s_1, a_1^1, a_1^2, \dots, a_{t-1}^1, a_{t-1}^2, s_t). \tag{10}$$

A stationary strategy is a special case of behavior strategy when $h_t = \emptyset$.

A decision rule assigns mixed strategies to different states. A decision rule of a stationary strategy has the following form: $\bar{\pi} = (\bar{\pi}(s^1), \dots, \bar{\pi}(s^m))$, where m is the maximal number of states. $\bar{\pi}(s)$ is a mixed strategy under state s.

A Nash equilibrium for stochastic games is defined as following, assuming that the players have complete information about the payoff functions of both players.

Definition 3 *In stochastic game* Γ*, a* Nash equilibrium point *is a pair of strategies* (π_*^1, π_*^2) *such that for all* $s \in S$

$$v^1(s, \pi_*^1, \pi_*^2) \geq v^1(s, \pi^1, \pi_*^2) \quad \forall \pi^1 \in \Pi^1$$

and

$$v^2(s, \pi_*^1, \pi_*^2) \geq v^2(s, \pi_*^1, \pi^2) \quad \forall \pi^2 \in \Pi^2$$

The definition of Nash equilibrium requires that each agent's strategy is a best response to the other's strategy. Such definition of Nash equilibrium is similar as in other games. The strategies that constitute a Nash equilibrium can be behavior strategies, Markov strategies, or stationary strategies. In this paper, we are interested in stationary strategies, which are the most simple strategies. The following theorem shows that there always exist a Nash equilibrium in stationary strategies for any stochastic game.

Theorem 1 *(Filar and Vrieze [4], Theorem 4.6.4) Every general-sum discounted stochastic game possesses at least one equilibrium point in stationary strategies.*

3.2 Stochastic games and bimatrix games

We can view each stage of a stochastic game as a bimatrix game, as in Figure 2.

At each time period of a stochastic game, under state s, agent 1 and 2 choose their actions independently and receive their payoffs according to the bimatrix game $(r^1(s), r^2(s))$. Repeated games can be seen as a degenerate case of stochastic games when there is only one state. For example, let $\bar{s}$ be the index of the only state, a repeated game will always have the bimatrix game $(r^1(\bar{s}), r^2(\bar{s}))$ at each time period.

4 A Multiagent Q-Learning Algorithm

The central issue of learning in multiagent systems is an agent's conjectures about other agents. Although from any individual agent's perspective the other agents may well be treated as part of the environment, a decision on the analyst's part to accord all of them *agent* status imposes an essential symmetry on the problem. It also helps the analyst to take into account the non-stationarity of the environment resulting from the learning behavior of others. An equilibrium concept characterizes some steady-state balance relationship among the agents. In multiagent systems, all the agents are simultaneously optimizing. The equilibrium (consistent joint optimization) thus represents the logical multiagent extension of individual optimization.

Nash equilibrium is a steady state of the play of a game in which each player holds the correct expectation about the other players' behavior and act

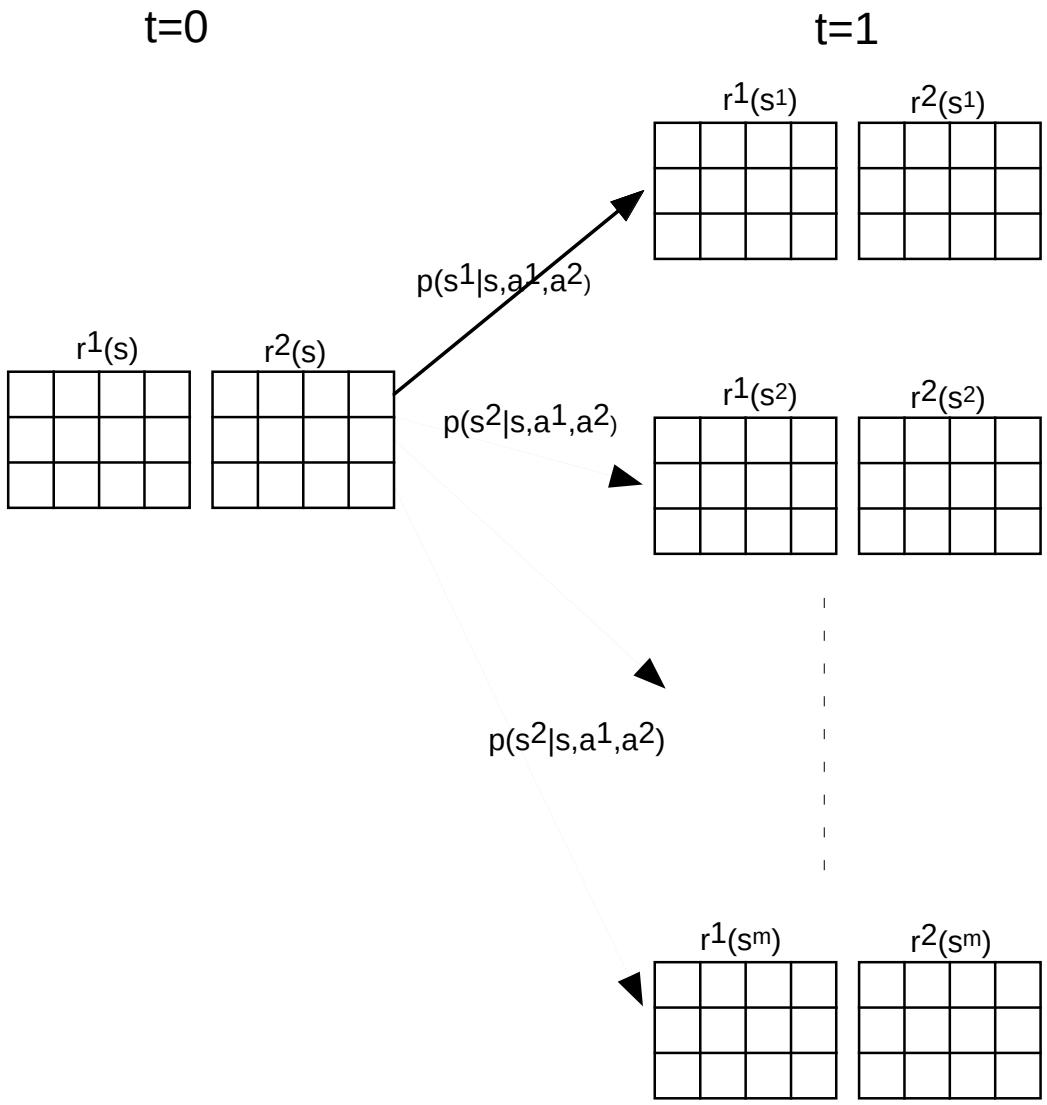


Figure 1: Stochastic games and bimatrix games

rationally. Acting rationally means each agent's strategy is the best response to the others' strategies. The justification for applying Nash equilibrium as a solution concept is based on the assumption of common knowledge of rationality and strategic behavior of all agents.

In stochastic games with incomplete information, agents do not know other agents' (and even their own) payoff functions. We assume that the agents can observe the immediate payoff of everyone at each time period. Using this information, the agents can gradually construct the payoff function of every other agent. Similarly, agents can learn about the transition probabilities in the game. Over time, all the information in the game is revealed to the agents, and the game becomes a complete information game, for which a Nash equilibrium can be derived. Q-learning serves two purposes. First, as a computational method it solves for the Nash equilibrium without requiring the knowledge of the transition probabilities. Second, the Q-values during learning provides the best approximation to the optimal values under complete information.

4.1 Multiagent Q Functions

For an n-player stochastic game, we define the *Nash equilibrium Q-values* for agent k, $k = 1, \ldots, n$, as the following:

$$Q_*^k(s, a^1, \ldots, a^n) = r^k(s, a^1, \ldots, a^n) + \beta \sum_{s' \in S} p(s'|s, a^1, \ldots, a^n) v^k(s', \pi_*^1, \ldots, \pi_*^n) \tag{11}$$

The Nash equilibrium Q-value is defined on state s and joint action $(a^1, \ldots, a^n)$. It is the total discounted reward received by an agent when the agents execute the joint action $(a^1, \ldots, a^n)$ in state s and follow Nash equilibrium strategies $(\pi_*^1, \ldots, \pi_*^n)$ thereafter.

To learn about these Q-values, an agent needs to maintain n Q-tables, one for each agent in the game. For agent k, an element of its own Q-table Q^k is represented by $Q^k(s, a^1, \ldots, a^n)$. The total number of entries in a Q-table is $m \prod_{i=1}^n |A^i|$, where m is the number of states, and $|A^i|$ is the size of action space A^i. Thus an agent needs to maintain $nm \prod_{i=1}^n |A^i|$ entries in its memory. Assuming $|A^1| = \cdots = |A^n| = |A|$, the space requirement is $nm|A|^n$, which is exponential in the number of agents. Thus for large numbers of agents, we would need to find some compact representation of action space.

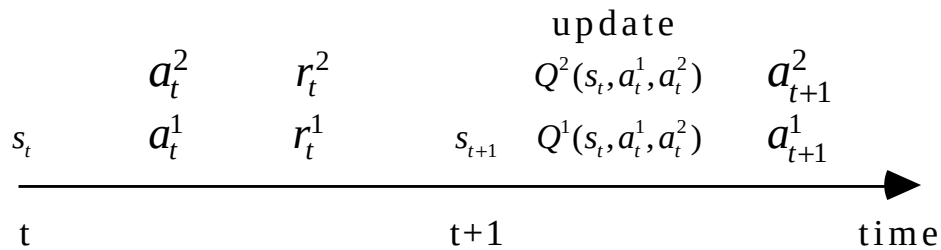


Figure 2: Time line of actions

As in single-agent Q-learning, the learning agent in multiagent systems updates its Q tables for a given state after it observes the state, actions taken by all the agents, and the rewards received by all the agents. The difference is in the updating rule. In single-agent Q-learning, the Q-values are updated as in (6). In multiagent Q-learning, we cannot just maximize over our own actions since the Q-values depend on the action of the other agent. We propose that an agent adopt a Nash strategy to update its Q-values. Given our argument at the beginning of this chapter, this is the most reasonable strategy an agent can take in noncooperative games.

4.2 Algorithm Description

Let $Q^k = (Q^k(s^1), \ldots, Q^k(s^m))$ be agent k's own Q-table. $Q^k(s^j)$ is the Q-table under state s^j, with each element represented by $Q^k(s^j, a^1, \ldots, a^n)$. The total number of entries in $Q^k(s^j)$ is $\prod_{i=1}^n |A^i|$. Agent k updates its Q-values according to the following rule:

$$Q_{t+1}^k(s, a^1, \ldots, a^n) = (1-\alpha_t)Q_t^k(s, a^1, \ldots, a^n) + \alpha_t[r_t^k + \beta\pi^1(s_{t+1}) \cdots \pi^n(s_{t+1})Q_t^k(s_{t+1})] \tag{12}$$

where $(\pi^1(s_{t+1}), \ldots, \pi^n(s_{t+1}))$ is a mixed-strategy Nash equilibrium for the normal-form game $(Q_t^1(s_{t+1}), \ldots, Q_t^n(s_{t+1}))$. Thus agent k needs to know all $Q_t^1(s_{t+1}), \ldots,$
$Q_t^n(s_{t+1})$ in order to derive $(\pi^1(s_{t+1}), \ldots, \pi^n(s_{t+1}))$. The information about other agents' Q-tables are not given, and agent k has to learn about them in the game. At the beginning of the game, agent k has no knowledge about other agents except their action spaces. As the game proceeds, agent k observes other agents' immediate rewards and previous actions. That information can then be used to update agent k's conjectures on other agents'

Q-tables. Agent k updates its belief on agent j's Q-table, for all $j \neq k$, according to the following rule:

$$Q^j_{t+1}(s, a^1, \ldots, a^n) = (1-\alpha_t)Q^j_t(s, a^1, \ldots, a^n) + \alpha_t[r^j_t + \beta\pi^1(s_{t+1}) \cdots \pi^n(s_{t+1})Q^j_t(s_{t+1})] \tag{13}$$

Our multiagent Q-learning algorithm is stated in Table 1.

Table 1: Multiagent Q-learning algorithm for Agent i

Initialize:
Let $t = 0$.
For all $s \in S$, all $a^k \in A^k$, $k = 1, \ldots, n$, let $Q^k_t(s, a^1, \ldots, a^n) = 0$.
Initialize the state, assign a value to s_0.
Loop
Choose action a^i_t.
Observe $r^1_t, \ldots, r^n_t; a^1_t, \ldots, a^n_t$, and s_{t+1}
Update Q^j for $j = 1, \ldots, n$
$Q^j_{t+1}(s, a^1, \ldots, a^n) = (1-\alpha_t)Q^j_t(s, a^1, \ldots, a^n) + \alpha_t[r^k_t + \beta\pi^1(s_{t+1}) \cdots \pi^n(s_{t+1})Q^j_t(s_{t+1})]$
where $(\pi^1(s_{t+1}), \ldots, \pi^n(s_{t+1}))$ is a mixed-strategy Nash equilibrium for
the normal-form game $(Q^1_t(s_{t+1}), \ldots, Q^n_t(s_{t+1}))$.
Let $t := t + 1$.

When the game is a 2-player zero-sum game, $Q^1(s, a^1, a^2) = -Q^2(s, a^1, a^2) = Q(s, a^1, a^2)$. Thus agent 1 needs to learn only one Q-table for every state. Our Q-learning algorithm becomes

$$Q_{t+1}(s, a^1, a^2) = \\ (1-\alpha_t)Q_t(s, a^1, a^2) + \alpha_t[r_t + \beta \max_{\pi^1(s_{t+1}) \in \sigma(A^1)} \min_{\pi^2(s_{t+1}) \in \sigma(A^2)} \pi^1(s_{t+1})Q_t(s_{t+1})\pi^2(s_{t+1})]$$

This is different from Littman's minimax-Q learning algorithm where Q-value is updated as

$$Q_{t+1}(s, a^1, a^2) = \\ (1-\alpha_t)Q_t(s, a^1, a^2) + \alpha_t[r_t + \beta \max_{\pi^1(s_{t+1}) \in \sigma(A^1)} \min_{a^2 \in A^2} \pi^1(s_{t+1})Q_t(s_{t+1}, a^2)]$$

In Littman's Q-learning algorithm, it is assumed that the other agent will always choose a pure strategy rather than a mixed strategy.

Note that an agent's action choice at each time t is not essential for the convergence of learning as long as the agent choose each action in each state for infinitely many times. But the action choices are important for short-term performance. We will discuss the issue of action choice in the end of this chapter.

4.3 Convergence of the Algorithm

We prove the convergence of our Q-learning algorithm for 2-player games. The results can be extended to n-player games, but with some modification. Following (11), the Nash equilibrium Q-values in 2-player games are defined as

$$Q_*^1(s, a^1, a^2) = r^1(s, a^1, a^2) + \beta \sum_{s' \in S} p(s'|s, a^1, a^2) v^1(s', \pi_*^1, \pi_*^2) \tag{14}$$

$$Q_*^2(s, a^1, a^2) = r^2(s, a^1, a^2) + \beta \sum_{s' \in S} p(s'|s, a^1, a^2) v^2(s', \pi_*^1, \pi_*^2) \tag{15}$$

Our convergence result relies on certain assumptions. The first two assumptions are standard ones in Q-learning:

Assumption 1 *Every state and action have been visited infinitely often.*

Assumption 2 *The learning rate α_t satisfies the following conditions for all s, t, a^1, a^2:*

1. $0 \leq \alpha_t(s, a^1, a^2) < 1, \sum_{t=0}^{\infty} \alpha_t(s, a^1, a^2) = \infty$, and $\sum_{t=0}^{\infty} [\alpha_t(s, a^1, a^2)]^2 < \infty$,

2. $\alpha_t(s, a^1, a^2) = 0$ if $(s, a^1, a^2) \neq (s_t, a_t^1, a_t^2)$.

Assumption 2 states that the learning rate is decreasing. Item 2 in the assumption states that the agent updates only the element in its Q-table corresponding to current state s_t and joint actions (a_t^1, a_t^2).

Notice that the learning rate $\alpha_t(s, a^1, a^2)$ is local to each tuple (s, a^1, a^2). This allows the agent to keep high learning rates for states that have not been visited much and low learning rates for frequently visited states.

We make further assumptions regarding the structure of the game:

Assumption 3 *The bimatrix game* $(Q_*^1(s), Q_*^2(s))$*, whose elements defined as in (14) and (15), as well as any bimatrix game* $(Q^1(s), Q^2(s))$ *encountered during learning has a Nash equilibrium* $(\pi^1(s), \pi^2(s))$ *satisfying one of the following properties:*

1. *The Nash equilibrium is optimal for both agents, meaning both agents receive their highest payoffs when they choose the Nash equilibrium strategies.*

   $\pi^1(s)Q^k(s)\pi^2(s) \geq \hat{\pi}^1(s)Q^k(s)\hat{\pi}^2(s)$ *for all* $\hat{\pi}^1(s) \in \sigma(A^1), \hat{\pi}^2(s) \in \sigma(A^2)$*, and* $k = 1, 2$.

2. *The Nash equilibrium is a saddle point, which means an agent receives a higher payoff when the other agent deviates from the equilibrium strategy.*
   $\pi^1(s)Q^1(s)\pi^2(s) \leq \pi^1(s)Q^1(s)\hat{\pi}^2(s)$ *for all* $\hat{\pi}^2(s) \in \sigma(A^2)$*, and*
   $\pi^1(s)Q^2(s)\pi^2(s) \leq \hat{\pi}^1(s)Q^2(s)\pi^2(s)$ *for all* $\hat{\pi}^1(s) \in \sigma(A^1)$.

Assumption 3 implies that a Nash equilibrium is either unique or has the same value as all others.

Our convergence proof is based on the following two Lemmas proved by Szepesvári and Littman [20].

Lemma 1 *(Conditional Average Lemma) Under Assumptions 1-2, the process*
$Q_{t+1} = (1 - \alpha_t)Q_t + \alpha_t w_t$ *converges to* $E(w_t|h_t, \alpha_t)$*, where* h_t *is the history at time t.*

Lemma 2 *Under Assumptions 1-2, if the process defined by*
$U_{t+1}(x) = (1 - \alpha_t)U_t(x) + \alpha_t[P_t v^*](x)$ *converges to* v^* *, and* P_t *satisfies* $\| P_t V - P_t v^* \| \leq \gamma \| V - v^* \| + \lambda_t$ *for all* V*, where* $0 < \gamma < 1$ *and* $\lambda_t \geq 0$ *converges to 0, then the iteration defined by*

$$V_{t+1}(x) = (1 - \alpha_t)V_t(x) + \alpha_t[P_t V_t](x) \tag{16}$$

converges to v^*.

This lemma can be explained as following: Given a sequence $\{V_t\}$ defined as in (16), if P_t maps every V_t closer to v^*, then the sequence $\{V_t\}$ updated by P_t will converge to v^*.

Our convergence proof is also based on a theorem by Filar and Vrieze [4]. The theorem states a Nash equilibrium of the bimatrix game $(Q_*^1(s), Q_*^2(s))$, with $(Q_*^1(s)$ and $Q_*^2(s))$ defined as in (14) and (15), is also a part of a stationary Nash equilibrium of the whole game.

Theorem 2 *(Filar and Vrieze [4]) The following assertions are equivalent:*

1. *For each $s \in S$, the pair $\left(\pi^1(s), \pi^2(s)\right)$ constitutes an equilibrium point in the static bimatrix game $\left(Q^1(s), Q^2(s)\right)$ with equilibrium payoffs $\left(v^1(s, \pi^1, \pi^2), v^2(s, \pi^1, \pi^2)\right)$, and for $k = 1, 2$,*

$$Q^k(s, a^1, a^2) = r^k(s, a^1, a^2) + \beta \sum_{s \in S} p(s'|s, a^1, a^2) v^k(s', \pi^1, \pi^2).$$

2. *(π^1, π^2) is an equilibrium point in the discounted stochastic game Γ with equilibrium payoff $\left(\boldsymbol{v}^1(\pi^1, \pi^2), \boldsymbol{v}^2(\pi^1, \pi^2)\right)$,*
where $\boldsymbol{v}^k(\pi^1, \pi^2) = (v^k(s^1, \pi^1, \pi^2), \cdots, v^k(s^m, \pi^1, \pi^2))$, $k = 1, 2$.

The above theorem states that the Nash solution of the bimatrix game $(Q^1(s),$
$Q^2(s))$, where $Q^1(s)$ and $Q^2(s)$ are defined as in Theorem 2, will also be part of the Nash solution for the whole dynamic game. If the sequence in our Q-learning algorithm converges to the Q-values defined in Theorem 2, then a pair of stationary Nash equilibrium strategies $(\bar{\pi}^1, \bar{\pi}^2)$ can be derived, where $\bar{\pi}^k = (\bar{\pi}^k(s^1), \cdots, \bar{\pi}^k(s^m))$ for $k = 1, 2$. For each state s, $\bar{\pi}^k(s)$ is part of a Nash equilibrium solution of the bimatrix game $(Q^1(s), Q^2(s))$.

Lemma 3 *Let $P_t Q = (P_t^1 Q^1, P_t^2 Q^2)$, with*

$$\begin{aligned} P_t^1 Q^1(s, a^1, a^2) &= r_t^1 + \beta \pi^1(s_t) Q^1(s_t) \pi^2(s_t) \\ P_t^2 Q^2(s, a^1, a^2) &= r_t^2 + \beta \pi^1(s_t) Q^2(s_t) \pi^2(s_t) \end{aligned}$$

where $\left(\pi^1(s_t), \pi^2(s_t)\right)$ is a mixed strategy Nash equilibrium for the bimatrix game $(Q^1(s_t), Q^2(s_t))$.[1] *Then $\| P_t Q - P_t Q_* \| \leq \beta \| Q - Q_* \|$ for all Q, where $Q_* = (Q_*^1, Q_*^2)$ is defined as in (14) and (15).*

[1] A more precise notation would write $\left(\pi^1(s_t), \pi^2(s_t)\right)$ as $\left(\pi^1(Q^1(s_t), Q^2(s_t)), \pi^2(Q^1(s_t), Q^2(s_t))\right)$. We use the notation $\left(\pi^1(s_t), \pi^2(s_t)\right)$ with the understanding that both $\pi^1(s_t)$ and $\pi^2(s_t)$ are functions of $Q^1(s_t)$ and $Q^2(s_t)$.

Proof. Case 1: $P_t^k Q^k(s,a^1,a^2) \geq P_t^k Q_*^k(s,a^1,a^2)$ for $k=1$ or 2.

$k=1$. Suppose property 1 of Assumption 3 is satisfied, meaning a Nash equilibrium is also a global optimal point, we have

$$\begin{aligned}
0 &\leq P_t^1 Q^1(s) - P_t^1 Q_*^1(s) \\
&= \beta\left(\pi^1(s)Q^1(s)\pi^2(s) - \pi_*^1(s)Q_*^1(s)\pi_*^2(s)\right) \\
&\leq \beta\left(\pi^1(s)Q^1(s)\pi^2(s) - \pi^1(s)Q_*^1(s)\pi^2(s)\right) \qquad (17) \\
&= \beta\sum_{a^1}\sum_{a^2}\pi^1(s,a^1)\pi^2(s,a^2)(Q^1(s,a^1,a^2) - \\
&\quad Q_*^1(s,a^1,a^2)) \qquad (18) \\
&\leq \beta\sum_{a^1}\sum_{a^2}\pi^1(s,a^1)\pi^2(s,a^2) \parallel Q^1(s) - Q_*^1(s) \parallel \\
&= \beta \parallel Q^1(s) - Q_*^1(s) \parallel,
\end{aligned}$$

where $\parallel Q^k(s) - Q_*^k(s) \parallel = \max_{a^1,a^2} |Q^k(s,a^1,a^2) - Q_*^k(s,a^1,a^2)|$. Inequality (17) is from property 1 of Assumption 3.

Suppose property 2 of Assumption 3 is satisfied, we have

$$\begin{aligned}
0 &\leq P_t^1 Q^1(s) - P_t^1 Q_*^1(s) \\
&= \beta\left(\pi^1(s)Q^1(s)\pi^2(s) - \pi_*^1(s)Q_*^1(s)\pi_*^2(s)\right) \\
&\leq \beta\left(\pi^1(s)Q^1(s)\pi^2(s) - \pi^1(s)Q_*^1(s)\pi_*^2(s)\right) \qquad (19) \\
&\leq \beta\left(\pi^1(s)Q^1(s)\pi_*^2(s) - \pi^1(s)Q_*^1(s)\pi_*^2(s)\right) \qquad (20) \\
&= \beta\sum_{a^1}\sum_{a^2}\pi^1(s,a^1)\pi_*^2(s,a^2)(Q^1(s,a^1,a^2) - \\
&\quad Q_*^1(s,a^1,a^2)) \qquad (21) \\
&\leq \beta\sum_{a^1}\sum_{a^2}\pi^1(s,a^1)\pi_*^2(s,a^2) \parallel Q^1(s) - Q_*^1(s) \parallel \\
&= \beta \parallel Q^1(s) - Q_*^1(s) \parallel .
\end{aligned}$$

Inequality (19) derives from the fact that $\pi_*^1(s)$ is the best response to $\pi_*^2(s)$ since they constitute a Nash equilibrium. Inequality (20) derives from property 2 of Assumption 3.

$k=2$, similar proof as above.

Case 2: $P_t^k Q^k(s,a^1,a^2) \leq P_t^k Q_*^k(s,a^1,a^2)$ for $k=1$ or 2. Similar proof as in Case 1.

For $k = 1$, under property 1 of Assumption 2, we have

$$\begin{aligned} 0 &\leq P_t^k Q_*^1(s) - P_t^k Q^1(s) \\ &\leq \beta \sum_{a^1} \sum_{a^2} \pi_*^1(s,a^1)\pi_*^2(s,a^2) \parallel Q_*^1(s) - Q^1(s) \parallel \\ &= \beta \parallel Q_*^1(s) - Q^1(s) \parallel . \end{aligned}$$

under property 2 of Assumption 2, we have

$$\begin{aligned} 0 &\leq P_t^k Q_*^1(s) - P_t^k Q^1(s) \\ &\leq \beta \sum_{a^1} \sum_{a^2} \pi_*^1(s,a^1)\pi^2(s,a^2) \parallel Q_*^1(s) - Q^1(s) \parallel \\ &= \beta \parallel Q_*^1(s) - Q^1(s) \parallel . \end{aligned}$$

$k = 2$, similar proof as above.

Therefore we have $|P_t^k Q^k(s) - P_t^k Q_*^k(s)| \leq \beta \parallel Q^k(s) - Q_*^k(s) \parallel$. Since this holds for every state s, we have $\parallel P_t^k Q^k - P_t^k Q_*^k \parallel \leq \beta \parallel Q^k - Q_*^k \parallel$. □

Now we proceed to prove our main theorem, which states that the multiagent Q-learning methods converges to the Nash equilibrium Q-values.

Theorem 3 *In stochastic game Γ, under Assumptions 1-3, the coupled sequences $\{Q_t^1, Q_t^2\}$, updated by*

$$Q_{t+1}^k(s,a^1,a^2) = (1-\alpha_t)Q_t^k(s,a^1,a^2) + \alpha_t[r_t^k + \beta\pi^1(s')Q_t^k(s')\pi^2(s')] \quad (22)$$

where $k = 1, 2$, and $(\pi^1(s'), \pi^2(s'))$ is a mixed strategy Nash equilibrium for the bimatrix game $(Q_t^1(s'), Q_t^2(s'))$. The sequences $\{Q_t^1, Q_t^2\}$ converge to the Nash equilibrium Q values (Q_*^1, Q_*^2), with elements of Q_*^1 and Q_*^2 defined as in (14) and (15).

Proof. By Lemma 3, $\parallel P_t^k Q^k - P_t^k Q_*^k \parallel \leq \beta \parallel Q^k - Q_*^k \parallel$.

From Lemma 1, the sequence

$$Q_{t+1}^k(s,a^1,a^2) = (1-\alpha_t)Q_t^k(s,a^1,a^2) + \alpha_t[r_t^k + \beta\pi^1(s')Q^k(s')\pi^2(s')]$$

converges to

$$\begin{aligned} &E(r_t^k + \beta\pi^1 Q^k(s')\pi^2) = \\ &\quad \textstyle\sum_{s'} P(s'|s,a^1,a^2)\Big(r^k(s,a^1,a^2) + \beta\pi^1(s')Q^k(s')\pi^2(s')\Big). \end{aligned}$$

Define T^k as

$$(T^k Q^k)(s, a^1, a^2) = \sum_{s'} P(s'|s, a^1, a^2)\Big(r^k(s, a^1, a^2) + \beta\pi^1(s')Q^k(s')\pi^2(s')\Big)$$

From above, the sequence $\{Q_t^k\}$ converges to $T^k Q^k$. It is easy to show that T^k is a contraction mapping. To see this, rewrite T^k as

$$T^k Q^k(s) = \sum_{s'} P(s'|s, a^1, a^2) P_t Q^k(s).$$

Since P_t is a contraction mapping of Q^k and $P(s'|s, a^1, a^2) \geq 0$, T^k is also a contraction mapping of Q^k. We proceed to show that Q_*^k defined in (12) is the fixed point of T^k. From the definition of T^k, we have

$$\begin{aligned}(T^k Q_*^k)(s, a^1, a^2) &= \sum_{s'} P(s'|s, a^1, a^2)\Big(r^k(s, a^1, a^2) + \beta\pi_*^1(s')Q_*^k(s')\pi_*^2(s')\Big) \\ &= r^k(s, a^1, a^2) + \sum_{s'} P(s'|s, a^1, a^2)\beta\pi_*^1(s')Q_*^k(s')\pi_*^2(s')\end{aligned}$$

By Theorem 2, $\pi_*^1(s')Q_*^k(s')\pi_*^2(s') = v^k(s', \pi_*^1, \pi_*^2)$, thus $Q_*^k = T^k Q_*^k$. Therefore the sequence

$$Q_{t+1}^k(s, a^1, a^2) = (1 - \alpha_t)Q_t^k(s, a^1, a^2) + \alpha_t[r_t^1 + \beta\pi_*^1 Q_*^k(s')\pi_*^2] \tag{23}$$

converges to $T^k Q_*^k = Q_*^k$. By Lemma 2, the sequence (22) converges to Q_*^k. □

We want to point out several things. First, the convergence result does not depend on the sequence of actions taken by the agents. It requires only that every action and state have been visited infinitely often. Second, the convergence depends on certain restrictions on the bimatrix games encountered during learning. This is required because Nash equilibrium operator is generally not a contraction operator. It remains an open question to what extent we can relax the restrictions. Third, the theorem implicitly assumes that the Nash equilibrium point is either unique or has the same payoffs as all other Nash equilibria. Thus there is no equilibrium selection problem here because agents can only use one Nash value to update their Q-tables.

5 An Example

In our experiments, we want to see whether the algorithm leads to convergence after a large number of trials. Even though we have proved in theory that the multiagent Q-learning algorithm will converge to a Nash equilibrium, our theorem did not address situations where some assumptions are violated. Our experiments also serve as illustrative examples on how the multiagent Q-learning method is implemented for a specific problem.

In applying the algorithm to situations where some assumptions are violated, we have to address many different issues. Suppose there are multiple Nash equilibria with distinct values for the bimatrix games during learning, we want to see the effect of agents' equilibrium selection. If agents choose the same equilibrium every time, would that lead to convergenceΓ What if agents choose different equilibriaΓ Suppose a Nash equilibrium is unique, but it is neither a saddle point nor an optimal point for both agents, does our Q-learning still convergeΓ

We tested our learning algorithm on the game shown in Figure 3. This game is closely related to the one used by Littman [9], the grid-games studied by Sutton and Barto [19], and an example by Mitchell [12]. The main difference between our game and Littman's is that we allow agents to receive positive (or negative) payoffs at the same time while in Littman's game one agent's gain is always the other agent's loss. Sutton's and Mitchell's games have only one agent, while we have two agents.

5.1 The Grid-World Game

In a 3×3 grid world, two agents move around the grid, trying to reach a goal square. Initially, the two agents are located at the lower left corner and lower right corner respectively, and the goal is located at the upper middle cell. An agent can move only one cell at each step. There are at most 4 directions to go from a cell: *Left, Right, Up, Down.* If two agents attempt to move into the same cell (excluding the goal cell), they are bounced back to their previous cells. The game ends when at least one agent reaches the goal. The two agents can reach the goal at the same time.

At the beginning of the game, each agent does not know the position of the goalstrategy, but we assume that they know the position of the other agent. In the terminology of game theory, agents have incomplete but perfect information. By incomplete information, we mean an agent does not know

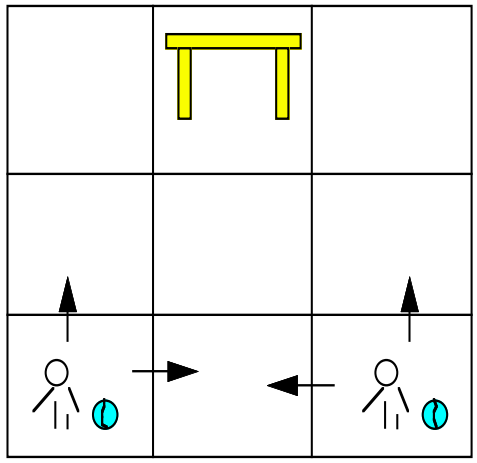


Figure 3: The Grid world game

another agent's payoff function. In the game here, an agent does not know the reward the other agent might get from their joint actions. By perfect information, we mean the current state and previous actions of all agents are observable. In the grid game, the agents know their positions and all previous movements taken by all the agents.

The task of each agent is to find its own shortest path to the goal. Since there are two agents in the system, they have to find shortest paths that do not interfere with each other. A path represents an agent's moves (actions) in different locations. Therefore a path can be viewed as an agent's stationary . A shortest path represents an optimal strategy. Two shortest paths that do not interfere with each other constitute a Nash equilibrium. When state transition is deterministic, meaning there is no uncertainty for the effect of the two agents' moves, there are 5 possible Nash equilibria in this game, as shown in Figure 4.

This grid-world game can be modeled as a stochastic game. The action space of agent i, $i = 1, 2$, is $A^i =\{$ *Left, Right, Down, Up*$\}$, with each element be an action a^i. The state space is $S = \{((1,1)(1,2)), ((1,1)(1,3)), \ldots, ((3,3)(3,2))\}$, where each state $s = (l^1, l^2)$ represents the agents' joint location. Agent i's location l^i, $i = 1$ or 2, is represented by (X, Y) coordinates, as shown in Figure 5. Since two agents cannot occupy the same position, neither can they rest in the goal position before the end of game, the number of possible joint positions is $8 \times 7 = 56$.

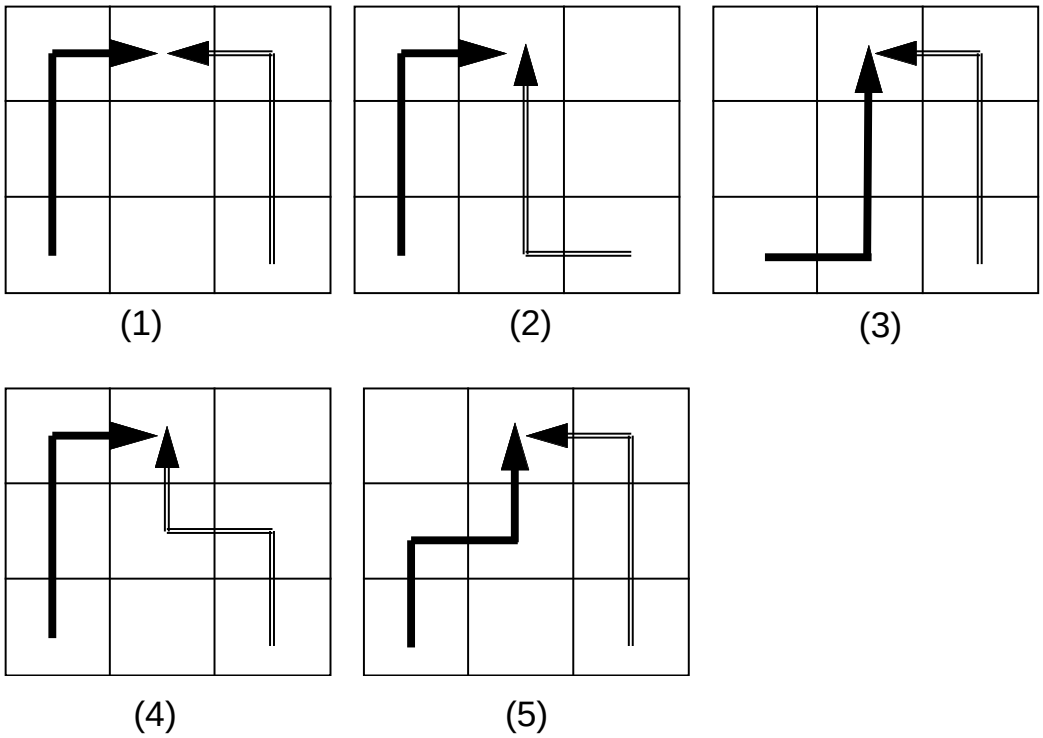


Figure 4: Nash equilibria of the grid-word game when state transition is deterministic

If an agent reaches the goal position, it scores 100 points. If it reaches other positions without colliding with the other agent, it scores 0 points. If it collides with the other agent, it scores -1 and both agents are bounced back to their previous positions. Let $L(l^i, a^i)$ be agent i's potential new location resulting from choosing action a^i in position l^i. The reward function is, for $i = 1, 2$,

$$r_t^i = \begin{cases} 100 & \text{if } L(l_t^i, a_t^i) = Goal \\ -1 & \text{if } L(l_t^1, a_t^1) = L(l_t^2, a_t^2) \neq Goal \\ 0 & \text{otherwise.} \end{cases}$$

Different state transition mappings define different equilibrium strategies. When state transition is deterministic for every state, the Nash equilibria are shown in Figure 4. We can see that there is one dominating strategy for each agent. Agent 1's dominating strategy is the left path, and agent 2's dominating strategy is the right path. Therefore it is relatively easy for a

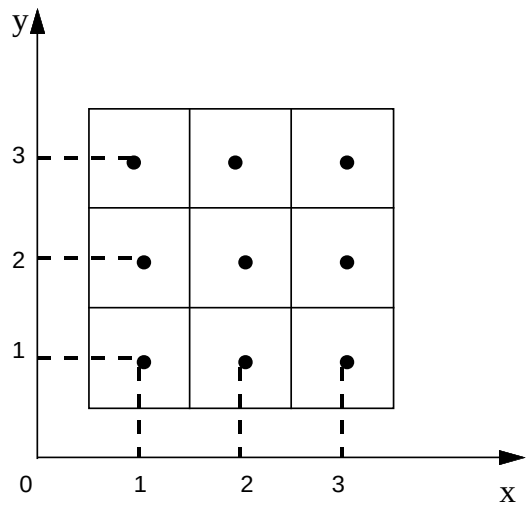


Figure 5: The coordinate for positions of agents in the grid game

learning algorithm to converge to the Nash equilibrium consisting of these two dominating strategies, which is (1) in Figure 4.

We want to test convergence in a more difficult case where the dominating Nash equilibrium is eliminated. We define the state transition to be deterministic for all states except the states with position $(1, 1)$ or $(3, 1)$. If an agent chooses *Up* from position $(1, 1)$ or $(3, 1)$, it moves up with probability 0.5 and remains in its previous position with probability 0.5. The joint probabilities of reaching new states from state ((1 1)(3 1)) and agent 1 taking *Up*, agent 2 taking *Up* are therefore

$$P\Big(((1\ 1)(3\ 1))|(1\ 1)(3\ 1), Up,\ Up\Big) = 0.25,$$
$$P\Big(((1\ 2)(3\ 1))|(1\ 1)(3\ 1), Up,\ Up\Big) = 0.25,$$
$$P\Big(((1\ 1)(3\ 2))|(1\ 1)(3\ 1), Up,\ Up\Big) = 0.25,$$
$$P\Big(((1\ 2)(3\ 2))|(1\ 1)(3\ 1), Up,\ Up\Big) = 0.25.$$

We are interested in state ((1 1)(3 1)) because it is the initial state for our game.

When agent 1 chooses *Up* and agent 1 chooses *Left* from state ((1 1)(3

1)), the probabilities for reaching the new states are:

$$P\Big(((1\ 1)(2\ 1))|(1\ 1)(3\ 1), \textit{Up, Left}\Big) = 0.5, \tag{24}$$

$$P\Big(((1\ 2)(2\ 1))|(1\ 1)(3\ 1), \textit{Up, Left}\Big) = 0.5. \tag{25}$$

Similarly, we have

$$P\Big(((2\ 1)(3\ 1))|(1\ 1)(3\ 1), \textit{Right, Up}\Big) = 0.5,$$

$$P\Big(((2\ 1)(3\ 2))|(1\ 1)(3\ 1), \textit{Right, Up}\Big) = 0.5.$$

If the two agents move from other positions, then we have to check whether they run into the same new position. If so, the agents will be bounced back to the previous positions with probability 1; if not, the agents will end up in the new positions with probability 1.

5.2 Stationary Nash Equilibrium Strategies

A stationary Nash equilibrium strategy assigns a mixed strategy to each state. For example, in state $s = ((1,1)(3,1))$, agent 1 can have a strategy $\pi^1(s) = \{0,1\}$, and agent 2 can choose $\pi^2(s) = \{0,1\}$, which are the probability distributions over each agent's available actions. In state s, the payoff matrixes for agents 1 and 2 are, respectively,

$$r^1(s) = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad r^2(s) = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}$$

where $(r^1(s), r^2(s))$ constitutes a bimatrix game. Such a game can be thought of as a simplified stochastic game with one period.

Agents' reward functions for each state can be represented as tables, as shown in Table 2 and 3.

	Right	*Up*
Left	-1,-1	0,0
Up	0,0	0,0

Table 2: Reward table in state ((1,1)(3,1))

		Agent 2			
		Left	*Right*	*Down*	*Up*
Agent 1	*Right*	100,0	100,0	100,0	100,100
	Down	0,0	0,0	0,0	100,0

Table 3: Reward table in state ((1,3)(2,2))

A pair of stationary Nash equilibrium strategies is denoted by $(\pi^1, \pi^2) = \Big((\pi^1(s^1),$
$\ldots, \pi^1(s^{56})), (\pi^2(s^1), \ldots, \pi^2(s^{56}))\Big)$, where $\pi^i(s^k) = \{\Pr(\mathit{Left}), \Pr(\mathit{Right}), \Pr(\mathit{Down}),$ $\Pr(\mathit{Up})\}$ for $i = 1, 2$ and $k = 1, \ldots, 56$.

The discount rate β is 0.99 for both agents in our game.

To see whether a pair of stationary strategies (π^1_*, π^2_*) is a Nash equilibrium, we can check whether the following two conditions are satisfied:

$$\pi^1_* = \arg\max_{\pi^1} v^1(s, \pi^1, \pi^2_*) = \arg\max_{\pi^1} \Big(\sum_{t=0}^{\infty} \beta^t E(r^1_t | \pi^1, \pi^2_*, s_0 = s) \Big) \quad (26)$$

$$\pi^2_* = \arg\max_{\pi^2} v^2(s, \pi^1_*, \pi^2) = \arg\max_{\pi^2} \Big(\sum_{t=0}^{\infty} \beta^t E(r^2_t | \pi^1_*, \pi^2, s_0 = s) \Big). \quad (27)$$

If $\pi^2_*(((1,1)(3,1))) = \{1, 0\}$, which means in state $((1,1)(3,1))$ agent 2 chooses *Left* with probability 1 and *Up* with probability 0, then agent 1's best response to this strategy is always choosing *Up*. Suppose agent 1 chooses *Right* instead, the two agent will be bounced back to their original position again and again. The accumulated discounted payoff for agent 1 is then

$$v^1(s^1, \pi^1, \pi^2_*) = -1 - 0.99 - 0.99^2 - \cdots = -100.$$

If agent 1 chooses *Up*, the probabilities of transition to new states ((1,2)(2,1)) and ((1,1)(2,1)) are given by equations (24) and (25). Thus the accumulated discounted payoff for agent 1 is

$$v^1(s^1, \pi^1, \pi^2_*) = \frac{1}{2}(0 + .99 v^1(((1,2)(2,1)), \pi^1, \pi^2_*)) + \frac{1}{2}(0 + .99 v^1(((1,1)(2,1)), \pi^1, \pi^2_*)).$$

Assuming agent 1 chooses the best-response strategies in all other states, we can propagate back the v-values, and get

$$v^1(s^1, \pi^1, \pi^2_*) = \frac{1}{2} \times (97) + \frac{1}{2} \times (0) = 48.5.$$

where $97 = 0.99 \times 0.99^2 \times 100$ is agent 1's discounted Nash equilibrium payoff at state ((1,2)(2,1)), and 0 is agent 1's discounted Nash equilibrium payoff at state ((1,1)(2,1)).

Thus agent 1's strategy of moving *Up* in state ((1,1)(3,1)) satisfies equation (26), and is part of a Nash equilibrium strategy. In this Nash equilibrium, agent 2 choose *Left* in state ((1,1)(3,1)), *Left* in state ((1,2)(2,1)) and ((1,3)(2,2)), and any action in any other state. Similarly, we can show that when agent 1 chooses *Up* in state ((1,1)(3,1)), the best response for agent 2 is moving left. Thus the strategy pair $(\pi_*^1(s^1), \pi_*^2(s^1)) = (\{0,1\}, \{1,0\})$, meaning agent 1 taking *Up* and agent 2 taking *Left* in state ((1,1)(3,1)), is part of the Nash equilibrium strategies.

Following the above reasoning, we can see that there are only two stationary Nash equilibria for the grid game, which are shown as (2) and (3) in Figure 4. These figures show that given that agent 1 chooses *Up* in state ((1,1)(3,1)), the best response for agent 2 is choosing *Left*. Therefore (1), (4) and (5) are not represent Nash equilibria because agent does not choose its best response action in those cases.

5.3 Off-line Learning

Our experiments aim to show that with enough training, our multiagent Q-learning method converges to stationary Nash equilibrium strategies. The learning results are the final Q-values for all state-action tuples. Each tuple $\langle s, a^1, a^2 \rangle$ is an entry for the bimatrix game $(Q^1(s), Q^2(s))$. For 56 states, we have 56 bimatrix games. According to Theorem 2, a Nash equilibrium for the bimatrix game $(Q^1(s), Q^2(s))$ is part of the stationary Nash equilibrium of the whole game.

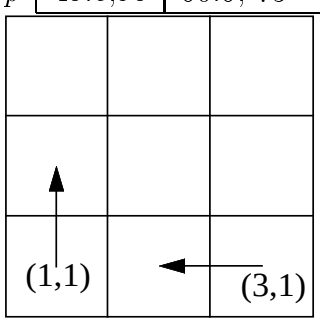
In our experiments, agents start with arbitrary Q-values. We let $Q^1(s, a^1, a^2) = 0$ and $Q^2(s, a^1, a^2) = 0$ for all s, a^1, a^2. The initial state is $((1,1)(3,1))$, which means agent 1 is in the left lower corner and agent 2 is in the right lower corner. Given the current state, agents choose their actions simultaneously. They then observe their own and the other's rewards and the next state. After that, agents update their Q-values according to (23). In the new state, agents repeat the above process. When one or both agents move into the goal position, the game re-starts with a new episode. In the new episode, each agent is randomly assigned a new position (except the goal), and the training repeats. The training stops after 5000 episodes. Each episode on average takes about 5 steps. So a complete training lasts about 25,000 steps.

The Nash equilibrium Q-values predicted by theory for state ((1,1)(3,1)) is shown in Figure 6. In this figure, (R_1, R_2) represents a pair of Nash equilibrium payoffs result from the fact that agent 1 and 2 follow their stationary Nash equilibrium strategies from state ((1,1)(3,1)). The values of R_1 and R_2 depend on which Nash equilibrium the agents choose.

Agent 1 \ Agent 2	*Left*	*Up*
Right	$-1+0.99R_1, \quad -1+0.99R_2$	$0.99 \times 99, \quad \frac{1}{2} \times 97 + \frac{1}{2} \times 0$
Up	$\frac{1}{2} \times 97 + \frac{1}{2} \times 0, 0.99 \times 99$	$\frac{1}{2} \times 97 + \frac{1}{4}R_1, \frac{1}{2} \times 97 + \frac{1}{4}R_2$

Agent 1 *Up* and agent 2 *left*

Agent 1 \ Agent 2	*Left*	*Up*
Right	47, 96	98,48.5
Up	48.5,98	60.6, 73



Agent 1 *Up* and agent 2 *left*

Agent 1 \ Agent 2	*Left*	*Up*
Right	96,47	98,48.5
Up	48.5,98	73, 60.6

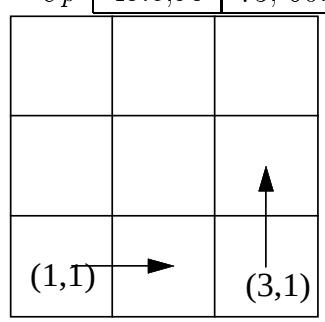


Figure 6: Nash equilibrium Q-values in state ((1,1)(3,1))

During learning, each agent maintains its own Q-values and the Q-values of its counterpart. When updating the Q-values, an agent has to choose a Nash equilibrium value from the next bimatrix game $(Q^1(s'), Q^2(s'))$, where s' is the next state. There may be multiple Nash equilibria. We assume that the agents choose the same Nash equilibrium to update their Q-values. If both agents observe the same information in the game, choosing the same Nash equilibrium to update their Q-values, and use the same updating rule, they would be able to learn the Q-values of the other agent.

Our experiments suggest that the final Q-values are sensitive to the learning rate. When the learning rate decreases too fast, $\alpha_t = 1/t$, the old results are not passed to the new one and there is very little learning. When learning

		Agent 2	
		Left	*Up*
Agent 1	*Right*	39,84	97,51
	Up	46,93	59,74

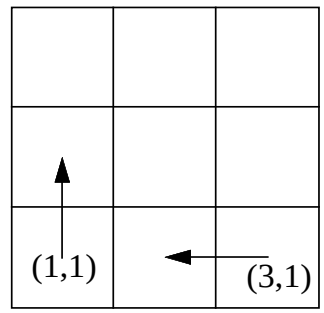


Figure 7: Final Q-values in state ((1,1)(3,1))

rate decrease too slow, $\alpha_t = 0.999^t$, the new results have little weight in the whole evaluation. We also have set learning rate to constant 0.5 and 0.9, in which the Q-learning does not converge to a Nash equilibrium. The learning rate we finally adopt is $\alpha_t(s, a^1, a^2) = \frac{1}{n_t(s,a^1,a^2)}$, where $n_t(s, a^1, a^2)$ is the number of times the tuple (s, a^1, a^2) has been visited. It is easy to show that this definition of learning rate satisfies the conditions $\sum_t \alpha_t(s, a^1, a^2) = \infty$ and $\sum_t \alpha_t^2(s, a^1, a^2) < \infty$ of Assumption 2.

When there are multiple Nash equilibria, we order them in a list. We show the experimental results for two cases. In case 1, agents always agree to choose the first Nash equilibrium; in case 2, agents always agree to choose the second Nash equilibrium if there is more than one. The results for Case 1 are shown in Figures 7, 8, and 9. The results for Case 2 are shown in Figures 10, 11, and 12.

Our experimental results show that when agents agree on their choice of Nash equilibrium of each bimatrix game during learning, their learning converges to a Nash equilibrium of the whole (stochastic) game. This means the Q-learning method still converges when there exist multiple Nash equilibria.

We further test the cases where agents disagree on the Nash equilibrium they choose. When there exist multiple Nash equilibria, if agents choose actions that belong to different Nash equilibria, the resulting joint action is not a Nash equilibrium. This is called the equilibrium selection problem. We test two cases. In case 1, agent 1 always chooses the first Nash equilibrium, agent 2 always chooses the second Nash equilibrium; in case 2, agent 1 chooses the Nash equilibrium that yields that highest expected payoff to agent 1 itself, agent 2 chooses the Nash equilibrium that yields that highest expected

Nash equilibrium Q-values

		Left	*Right*	*Up*
Agent 1	*Right*	99, 0	99, 0	97, 97
	Down	97, 97	$0.99R_1, 0.99R_2$	0, 99
	Up	99, 0	99, 0	99, 99

Agent 2

		Left	*Right*	*Up*
Agent 1	*Right*	97, 0	98, 0	84, 87
	Down	86, 86	41, 88	0, 99
	Up	99, 0	99, 0	99, 96

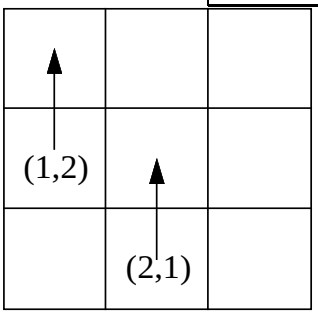


Figure 8: Final Q-values in state ((1,2)(2,1))

Nash equilibrium Q-values

Agent 1	Agent 2: *Left*	*Right*	*Down*	*Up*
Right	100, 0	100, 0	100, 0	100, 100
Down	98, 98	98, 98	98, 98	0, 100

Agent 1	*Left*	*Right*	*Down*	*Up*
Right	100,0	100,0	100,0	100,100
Down	95,95	98,98	85,85	0,100

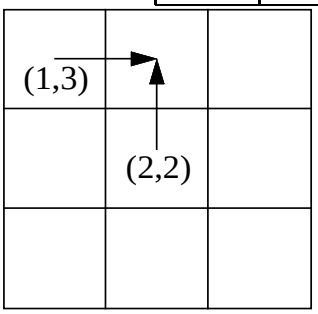


Figure 9: Final Q-values in state ((1,3)(2,2))

payoff to agent 2 itself. The results for Case 1 under state ((1, 1)(3, 1)) are shown in Figure 13. The results for state ((1,2)(2,1)) and ((1,3)(2,2)) are not shown because they have the same Nash equilibria as in previous cases. Our experiments for Case 1 show that even though agents have learned different final Q-values due to their different choices of Nash equilibria, their final Q-values are still close to each other. There is a unique Nash equilibrium for each of the final bimatrix games, and both agents agree on them.

The results for Case 2 under state ((1, 1)(3, 1)) are shown in Figure 14. The agents have learned very similar Q-values, and they are able to agree on the final Nash equilibrium.

6 Discussions on Online Learning Performance

We are interested in applying the multiagent Q-learning method to online learning. In off-line learning, action choices during learning are not important as long as agents try out all possible actions with infinitely times. In online learning, however, an agent has to choose actions at each step so as to maintain a reasonable run-time performance.

		Agent 2	
		Left	*Up*
Agent 1	*Right*	49, 82	46, 51
	Up	51, 92	62, 73

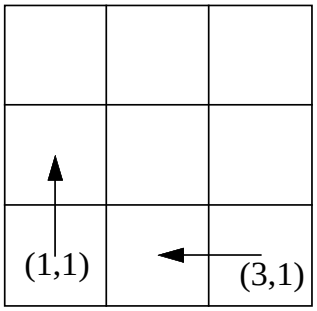


Figure 10: Final Q-values of choosing second Nash in state ((1,1)(3,1))

		Agent 2		
		Left	*Right*	*Up*
Agent 1	*Right*	96, 0	98, 0	94, 84
	Down	97, 85	50, 85	0, 99
	Up	99, 0	99, 0	99, 96

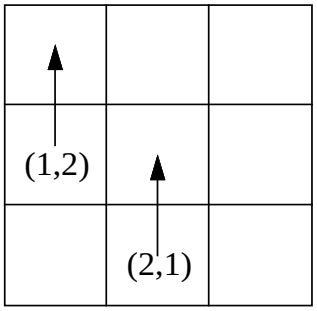


Figure 11: Final Q-values of choosing second Nash in state ((1,2)(2,1))

There are two reasons that an agent might care about its run-time performance. First, the agent is uncertain about when the game will end. Therefore it cannot solely rely on the asymptotic property of the Q-learning algorithm. By getting a reasonable payoff at each step, the agent can maintain a payoff as good as possible when the game ends. Second, In certain games, each agent has to maintain some minimum payoff above a threshold throughout the game. The payoff can be thought of as the agent's energy level or food storage level. Below the threshold, the agent will die.

In a dynamic setting, an optimal action should not only maximize an agent's current payoff, but also future payoffs. If the agent has no uncertainty of the environment, a dynamic programming method can be used to find the optimal action. When the uncertainty exists, the agent's current

Agent 1 \ Agent 2	*Left*	*Right*	*Down*	*Up*
Right	100,0	100,0	100,0	100,100
Down	98,95	98,94	98,85	0,100

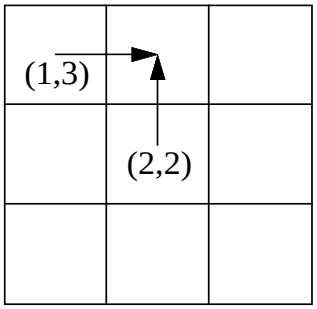
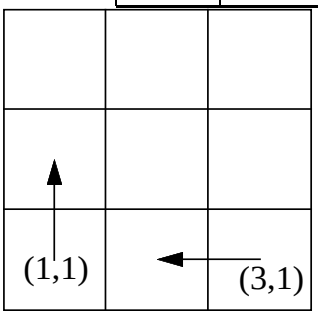


Figure 12: Final Q-values of choosing second Nash in state ((1,3)(2,2))

Agent 1's learning result

Agent 1 \ Agent 2	*Left*	*Up*
Right	39,84	97,51
Up	46,93	59,74



Agent 2's learning result

Agent 1 \ Agent 2	*Left*	*Up*
Right	49, 82	46, 51
Up	51, 92	62, 73

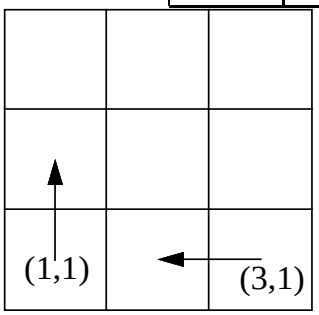
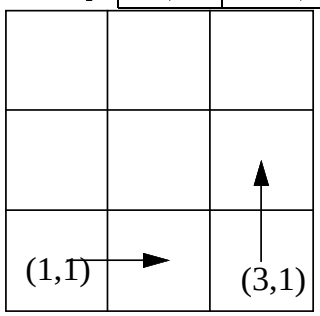


Figure 13: Final Q-values with agent 1 choosing the first Nash and agent 2 choosing the second Nash in state ((1,1)(3,1))

Agent 1's learning result

Agent 1 \ Agent 2	*Left*	*Up*
Right	84,46	92,50
Up	54,93	73,62



Agent 2's learning result

Agent 1 \ Agent 2	*Left*	*Up*
Right	83, 47	91, 50
Up	52, 93	73, 62

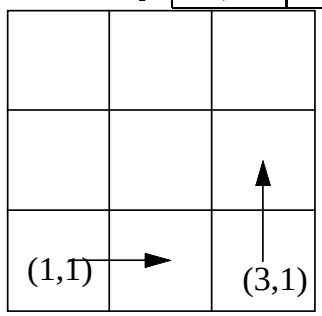


Figure 14: Different Q-values by choosing the expected Nash in state ((1,1)(3,1))

"best" action may not be the actual best. Therefore the agent has to explore the environment to get more information, while it aspires to maintain the good performance. The tradeoff between the current best action and a potential best action in the long run is the so-called exploitation vs. exploration tradeoff.

The uncertainty of the environment can be characterized by different things. In a multi-armed bandit problem, the reward of each action (arm) follows a normal distribution, whose characteristic parameters are unknown to the agent. Meuleau and Bourgine [10] extend the exploration techniques studied in multi-armed bandit problems to multi-state decision problems.

Different exploration strategies have different merits. For simplicity, we adopt ϵ-Greedy exploration strategy defined by Singh et al. [17]. In this strategy, the agent explores with probability $\epsilon_t(s)$ and choose the optimal action with probability $1 - \epsilon_t(s)$. Singh et al [17] proved that when $\epsilon_t(s) = c/n_t(s)$ with $0 < c < 1$, the learning policy satisfies the GLIE (Greedy in the Limit with Infinite Exploration) property.

The exploitation action is an action that satisfies:

$$\max_{a_k} E(Q^1(a_k, b, s)) = \max_{a_k} \Big(\sum_b \hat{P}(b) Q^1(a_k, b, s) \Big), \tag{28}$$

where $\hat{P}(b)$ is agent 1's expectation on agent 2's probability of choosing action

b. Agent 1 derives $\hat{P}(b)$ from counting agent 2's frequencies of taking different actions. This is similar as in fictitious play.

We test agent 1's performance under three different strategies. These strategies include the exploit strategy, the random (exploration) strategy, and the exploit vs. exploration strategy. In the exploit strategy, an agent tries every action at least once, then it always chooses a best response action based on criterion (28). In the pure exploration case, an agent always chooses an action randomly. In a large number of trials, this usually lead to equal chances of trying out different actions.

One of the experimental results is shown in Figures 15. The experiments show that when agent 2 uses the exploit strategy, agent 1 performs best by using exploit strategy too. This is as we expected because the exploit agent takes advantage of the fixed behavior of the other agent.

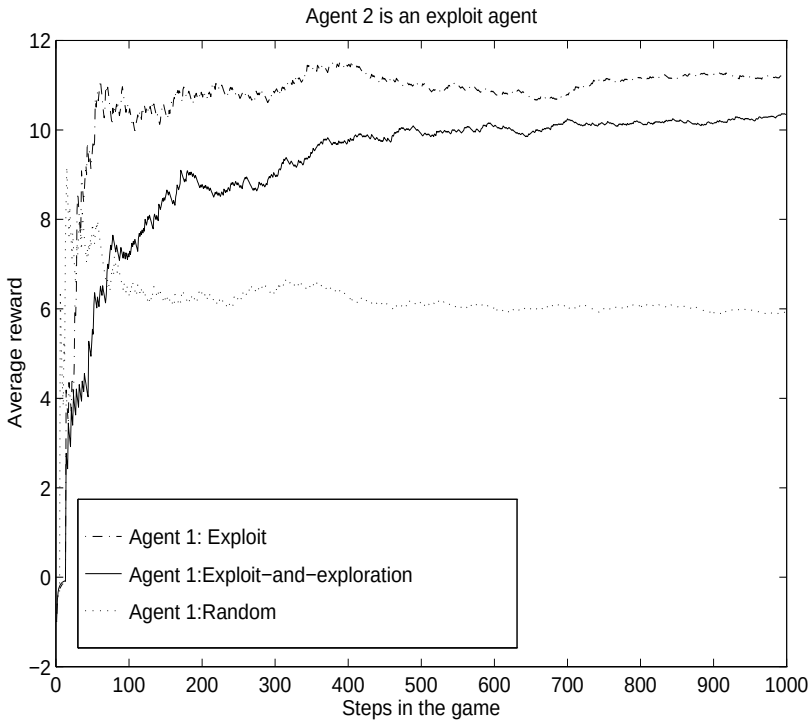


Figure 15: Agent 1's performance under different exploration strategies

7 Summary

We define a multiagent Q-learning method, and prove that it converges to a Nash equilibrium under certain assumptions. This learning method is implemented in a grid-world game. We show that agents can use this learning method to learn stationary Nash equilibrium strategies. In theory, the convergence of our learning algorithm relies on certain restrictions of the game structure during learning, which imply a unique Nash equilibrium value in each bimatrix game. In the experiments, we relax the restrictions and show that convergence is still possible when there are multiple Nash equilibria. We further discuss the application of this learning method to online settings, and provide some preliminary results.

References

[1] Tucker Balch. Learning roles: Behavioral diversity in robot teams. In Sen [16].

[2] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752, Madison, WI, 1998.

[3] Edwin De Jong. Non-random exploration bonuses for online reinforcement learning. In Sen [16].

[4] Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Process*. Springer-Verlag, 1997.

[5] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[6] John Herz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, California, 1991.

[7] Leslie Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[8] Pat Langley, Herbert Simon, Gary L. Bradshaw, and J. M. Zytkow. *Scientific Discovery: Computational Explorations of the Creative Processes.* MIT Press, Cambridge, Massachusetts, 1987.

[9] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163. New Brunswick, 1994.

[10] Nicolas Meuleau and Paul Bourgine. Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, 35(2), May 1999.

[11] Tom Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

[12] Tom Mitchell. *Machine Learning*, chapter 13. Reinforcement Learning, pages 367–390. McGraw-Hill, 1997.

[13] Kumpati S. Narendra and Mandayam A.L. Thathachar. *Learning Automata: an Introduction.* Prentice Hall, 1989.

[14] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, New York, 1994.

[15] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[16] Sandip Sen, editor. *Collected Papers from the AAAI-97 Workshop on Multiagent Learning.* AAAI Press, 1997.

[17] Satinder Singh, Tommi Jaakkola, Michael L. Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement learning algorithms. *Machine Learning.* To appear.

[18] Peter Stone. *Layered Learning in Multi-Agent Systems.* PhD thesis, Carnegie Mellon University, December 1998.

[19] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[20] Csaba Szepesvári and Michael L. Littman. A unified analysis of value-function-based reinforcement-learning algorithms. submitted for review, October 1998.

[21] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, Amherst, MA, June 1993. Morgan Kaufmann.

[22] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 3:279–292, 1992.