# Best-Response Multiagent Learning in Non-Stationary Environments

Michael Weinberg Jeffrey S. Rosenschein
School of Engineering and Computer Science
Hebrew University
Jerusalem, Israel
{mwmw,jeff}@cs.huji.ac.il

## Abstract

*This paper investigates a relatively new direction in Multiagent Reinforcement Learning. Most multiagent learning techniques focus on Nash equilibria as elements of both the learning algorithm and its evaluation criteria. In contrast, we propose a multiagent learning algorithm that is optimal in the sense of finding a best-response policy, rather than in reaching an equilibrium. We present the first learning algorithm that is provably optimal against restricted classes of non-stationary opponents. The algorithm infers an accurate model of the opponent's non-stationary strategy, and simultaneously creates a best-response policy against that strategy. Our learning algorithm works within the very general framework of $n$-player, general-sum stochastic games, and learns both the game structure and its associated optimal policy.*

## 1. Introduction

Multiagent learning has been a research area of much interest over the last decade [13, 11]. The progress made on reinforcement learning [14], in particular, has opened the way for designing autonomous agents capable of acting in unknown environments by exploring different possible actions and their consequences. However, a major drawback of reinforcement learning has been its assumption of a stationary underlying environment. A more complex environment, inhabited by autonomous, self-interested agents acting to maximize their own payoffs, cannot be treated in this manner.

We present NSCP-learner[1] — the first learning algorithm that is provably optimal against restricted classes of non-stationary opponents. The algorithm infers a provably accurate model of the opponent's non-stationary strategy, and at the same time designs a best-response policy against that strategy. The algorithm works within the very general framework of $n$-player, general-sum stochastic games, and learns both the game structure and its associated optimal policy. We provide both theoretical and experimental results of our algorithm.

In contrast to previous work in multiagent learning, we eschew Nash equilibria as a relevant evaluation criterion for an agent's learned policy. Instead, we embrace the so-called "AI Agenda" [12] and its goal of best-response policies against specific classes of opponents.

### 1.1. Reinforcement Learning in Multiagent Environments

We here briefly discuss previous research on reinforcement learning (RL), and its relevance to multiagent environments.

Most work in reinforcement learning has been of the single-agent type. In single-agent RL, we are concerned with a lone agent operating in an unknown environment so as to maximize its payoffs. The environment is usually modeled as a Markov Decision Process (MDP), with the agent as the controller of the process. The first, and most well-known, algorithm for single-agent RL is Q-learning [16]. Q-learning acts in an unknown environment (in the sense of rewards and transitions), and has been proven to converge to an optimal policy that maximizes the expected reward.

However, a major drawback of reinforcement learning is its assumption that the underlying environment is stationary. This assumption may be inappropriate for more complex environments, inhabited by autonomous, self-interested agents. In fact, if several agents co-inhabiting an environment all use the Q-learning algorithm concurrently, none converges to an optimal policy.

It is worth noting, however, that if all (other) agents in an environment use only stationary policies, then the (lone) Q-learner would, in fact, converge to an optimal policy. In this simplified case, the other agents may be treated simply

1 NSCP stands for Non-Statonary Converging Policies


AAMAS'04, July 19-23, 2004, New York, New York, USA.

as part of the environment, and since this environment can still be modeled as an MDP, the Q-learning convergence assumptions hold. The algorithm we present in this paper is more flexible, in the sense that it is proved to converge to an optimal policy even if the environment is inhabited by certain types of non-stationary agents.

Multiagent reinforcement learning is the attempt to extend RL techniques to the setting of multiple agents. The theoretical framework in which multiagent RL takes place is either matrix games or stochastic games; matrix games are games with one state, and stochastic games are games with multiple states.

Littman [8] introduced the Mini-Max Q-learning algorithm, which was proven to converge to the game-theoretic optimal value for zero-sum (purely competitive) two-player games. Claus and Boutilier [3] investigated the convergence to Nash equilibrium of JALs (Joint Action Learners) in the setting of common-payoff (team) games. Hu and Wellman [7] introduced the Nash-Q algorithm which works for general-sum games (when the payoffs of agents are not correlated), and under certain assumptions converges to Nash equilibrium.

Littman [9], in turn, noticed that the assumptions of Nash-Q are quite restrictive, and reinterpreted the algorithm as the Friend-or-Foe (FOF) algorithm. The FOF-learner treats each other agent as either friend or foe, and the algorithm converges to Nash equilibrium for special cases of stochastic games. CE-Q [5] is similar to Nash-Q, but uses the value of correlated equilibria instead of Nash equilibria.

As can be seen, the goal of most multiagent learning algorithms mentioned above is to converge to a (Nash) equilibrium. However, this approach is problematic on many grounds [12]. It is not clear what justifies the focus on equilibrium: equilibrium may identify a point at which learning should stop, but it does not necessarily have to be the goal of the learning process. Another problem with this approach is the potential existence of multiple equilibrium points. In these cases, the learning agents need some kind of an oracle in order to coordinate their choice of a unique equilibrium point.

Shoham has proposed the so called "AI Agenda" as a more appropriate direction for the development of the field of multiagent learning. The "AI Agenda" asks what is the best learning strategy for an agent, given a fixed class of other agents in the game. In other words, it asks how to design an optimal agent for a given environment inhabited by certain types of other agents. The goal of the learning process is to get an 'optimal' payoff in the presence of other agents, and not to consider equilibrium as necessary or even relevant.

Following this line, Hu [6] introduced the Best-Response algorithm for multiagent learning. It was proven to converge to a best response policy in the presence of a stationary opponent in general-sum, two-player stochastic games. Similarly, Conitzer and Sandholm introduced AWESOME [4], a learning algorithm for repeated matrix games that learns to play optimally against stationary opponents, and converges to a Nash equilibrium in self-play.

Bowling and Veloso [2] proposed criteria for multiagent learning algorithms: learning should (1) always converge to a stationary policy, and (2) only terminate with a best-response to the play of other agents. While our algorithm satisfies these requirements, it does not satisfy another criterion mentioned by Conitzer and Sandholm [4]: convergence to Nash equilibrium in self-play. While this last criterion is quite desirable, we argue that it might be overly strict when designing best-response learning algorithms against fixed classes of opponents — the problem of designing best-response algorithms for even simple classes of agents is hard enough. We discuss this issue at greater length below in Section 6.

Our NSCP learning algorithm continues the research direction of exploring best-response learning algorithms, and derives best-response policies against a restricted class of *non-stationary* opponents. As mentioned above, it works in the general framework of $n$-player, general-sum stochastic games.

## 2. Theoretical Framework and Definitions

We will consider multiagent reinforcement learning in the most generic model, namely general-sum stochastic games. As background for our presentation, we present in this section definitions of a stage game, stochastic game, strategy, and stationary/non-stationary policy in a game.

**Definition 1** *An N-player* stage game (or matrix game) *is a tuple* $\langle n, A_{1..n}, R_{1..n} \rangle$*, where* $n$ *is the number of agents,* $A_i$ *is the discrete action space available to agent* $i$*, and* $R_i$ *is the reward function for agent* $i$ *—* $R_i : A_1 \times \cdots \times A_n \rightarrow \Re$.

**Definition 2** *A* strategy $S$ *in a stage game is a probability distribution that assigns probability* $p_i$ *to each possible action* $a_i$ *—* $S : A \rightarrow [0, 1]$.

**Definition 3** *The distance D between two stage game strategies* $S_1$ *and* $S_2$ *is the distance between the probability vectors of the strategies.*

$$D(S_1, S_2) = |S_1 - S_2|_1$$

We now define a *stochastic game*, which is an extension of the stage game to multiple stages.

**Definition 4** *An N-player* stochastic game *is a tuple* $\langle n, S, A_{1..n}, T, R_{1..n} \rangle$*, where* $n$ *is the number of*


AAMAS'04, July 19-23, 2004, New York, New York, USA.

*agents, $S$ is the discrete state space, $A_i$ is the discrete action space available to agent $i$, $T$ is the transition function $T : S \times A_1 \times \cdots \times A_n \times S \to [0, 1]$ and $R_i$ is the reward function for agent $i$ — $R_i : S \times A_1 \times \cdots \times A_n \times S \to \Re$.*

A strategy for a stochastic game assigns a stage-game strategy to each possible stage (state) in the game. A policy for the stochastic game is thus a sequence of strategies.

**Definition 5** *A* policy $\pi$ *in a stochastic game is a (possibly infinite) sequence of strategy rules $\pi = (\pi_0, ..., \pi_t, ...)$. $\pi_t$ is called the strategy rule at time $t$ and assigns probability $p_i$ to action $a_j$ at state $s_k$ — $\pi_t : S \times A \to [0, 1]$.*

**Definition 6** *A policy $\pi = (\pi_0, ..., \pi_t, ...)$ is called* stationary *if the strategy rules are independent of time — $\forall t : \pi_t = \pi_{t-1}$. A policy is called* non-stationary *if the strategy rules change over time — $\exists t_i, t_j$ such that $\pi_{t_i} \neq \pi_{t_j}$.*

**Definition 7** *The distance between two strategy rules $\pi_{t_i}$ and $\pi_{t_j}$ is defined as*

$$\left|\pi_{t_i} - \pi_{t_j}\right| = \sum_{s \in S} D(\pi_{t_i}(s), \pi_{t_j}(s))$$

*where $\pi_{t_k}(s)$ denotes the probability vector over actions played in state $s$ by the decision rule $\pi_{t_k}$.*

**Definition 8** *We define an* epoch *as a period of rounds during which the opponents' strategies do not change.*

We do not restrict the length of the epoch (e.g., it can be 1), but we assume that we do know its length $L$.

# 3. NSCP: Best-Response Q-Learning Algorithm for Non-Stationary Policies with a Limit

In this section, we present a Q-learning-like learning algorithm which is provably optimal in environments inhabited by agents whose policies are non-stationary but have a limit. In the following subsections, we provide a brief review of single-agent Q-learning, formally define the class of non-stationary agents that have a limit, and then present our NSCP-learner and prove its convergence.

## 3.1. Single Agent Q-learning

In single-agent Q-learning [16], the goal of the agent is to learn the optimal Q-values, defined as

$$Q_*(s, a) = r(s, a) + \beta \cdot \sum_{s'} p(s'|s, a) \cdot v(s', \pi_*)$$

where

$$v(s', \pi_*) = \max_a Q_*(s, a)$$

The agent starts with arbitrary Q-values, and updates them as follows:

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

Under standard RL assumptions, the sequence $Q_t$ converges to $Q_*$ with probability 1, and the optimal policy is simply taking the action that maximizes $Q_t(s, a)$ at any state $s$.

## 3.2. Non-Stationary Policies with a Limit

Our algorithm works with any number of agents that come from the class of agents whose policies have a limit. More formally, the probability that the strategy rule would be far away from the limit gets smaller as $t$ gets larger.

**Definition 9** *A non-stationary policy $\pi = (\pi_1, \ldots, \pi_t, \ldots)$ has a limit if it satisfies*

$$\exists \pi_* \quad s.t. \quad \forall \epsilon > 0 \quad \lim_{t \to \infty} P(|\pi_t - \pi_*| > \epsilon) = 0$$

Note that although we do require the opponents' strategies to converge in probability, this is different from players that play changing strategies but eventually settle down to stationary strategies after a constant, finite number of moves. Our opponents may *never* settle down to fixed, stationary strategies, but rather continue with decreasing changes forever. In addition, we do not require or assume that the limit $\pi_*$ is known to our algorithm.

## 3.3. Modeling Policies with a Limit

We here describe how the approximated model $\hat{\pi} = (\hat{\pi}^i_1, \ldots \hat{\pi}^i_t, \ldots)$ is derived based on observations of agent $i$'s actions taken according to his real policy $\pi = (\pi^i_1, \ldots \pi^i_t, \ldots)$. Our goal is to have an accurate model of the other agent's policy; in other words, we want our model sequence to converge to the real sequence: $|\hat{\pi}_t - \pi_t| \to 0$ as $t \to \infty$.

When another agent plays a stationary strategy, it is obvious that algorithms such as fictitious play converge to the agent's stationary strategy, since they get more and more samples from the same fixed distribution. However, we allow the strategy of the opponent to change after $L$ moves (an epoch), and we never get more than $L$ samples from the same distribution. Thus, it is not obvious that fictitious play would satisfy $|\hat{\pi}_t - \pi_t| \to 0$ as $t \to \infty$ — but, in fact, it does.

The intuition is that we get $L$ samples of each opponent's strategy rule, and since the distance between the strategy rules gets smaller, the $L$ samples of the previous rule may be used with increasingly good accuracy to model the current rule as well. Thus, as $t \to \infty$, we can use more and more


AAMAS'04, July 19-23, 2004, New York, New York, USA.

samples to model the current strategy rule since the samples from previous epochs describe the current strategy rule increasingly well (because the distance between the strategy rules is getting smaller).

Let $t$ denote the number of the epoch and $n_t(s,a)$ the number of times action $a$ was chosen in state $s$ during epoch $t$. We then define $\hat{\pi}_{t+1}$ as

$$\hat{\pi}_{t+1}(s,a) = \frac{\frac{n_1(s,a)}{L} + \ldots + \frac{n_t(s,a)}{L}}{t}$$

**Lemma 1**

$$\lim_{t\to\infty} P(|\hat{\pi}_t - \pi_t| > \epsilon) = 0$$

**Proof:** $\hat{\pi}_t$ and $\pi_t$ are defined over all states and all actions and assign probability $p$ to action $a$ at state $s$. Let $p_t = \pi_t(s,a)$ , $p_* = \pi_*(s,a)$ and $\hat{p}_t = \hat{\pi}_t(s,a)$. We know that $\lim_{t\to\infty} P(|p_t - p_*| > \epsilon) = 0$ and we have to show that $\lim_{t\to\infty} P(|\hat{p}_t - p_t| > \epsilon) = 0$.

Let $X$ be a random variable defined as:

$$X = \begin{cases} 1 & \text{if action } a \text{ was chosen in state } s \\ 0 & \text{otherwise} \end{cases}$$

In epoch $t$ the probability to play action $a$ in state $s$ is $p_t$, so $E(X) = 1 \cdot p_t + 0 \cdot (1 - p_t) = p_t$. For epoch $t$ we define $\tilde{p}_t$ to be the proportion of action $a$ in state $s$ during the epoch:

$$\tilde{p}_t = \frac{X_1 + \ldots + X_L}{L}$$

and our estimator of $p_t$ at epoch $t+1$ is defined as:

$$\hat{p}_{t+1} = \frac{\tilde{p}_1 + \ldots + \tilde{p}_t}{t}$$

The expected value of $\tilde{p}_t$ is

$$\begin{aligned} E(\tilde{p}_t) &= E\left(\frac{X_1 + \ldots + X_L}{L}\right) \\ &= \tfrac{1}{L} \cdot (E(X_1) + \ldots + E(X_L)) \\ &= \tfrac{1}{L} \cdot L \cdot p_t = p_t \end{aligned}$$

When $t \to \infty$ we have $p_t \to p_*$ so $\lim_{t\to\infty} E(\tilde{p}_t) = p_*$.

Since $p_*$ is the expected value of $\tilde{p}_t$, from the central limit theorem we get

$$\lim_{t\to\infty} P\left(\left|\frac{\tilde{p}_1 + \ldots + \tilde{p}_t}{t} - p_*\right| > \epsilon\right) = 0$$

Replacing the fraction by $\hat{p}_t$ we get

$$\lim_{t\to\infty} P(|\hat{p}_t - p_*| > \epsilon) = 0$$

and because it is given that

$$\lim_{t\to\infty} P(|p_t - p_*| > \epsilon) = 0$$

we get as needed that

$$\lim_{t\to\infty} P(|\hat{p}_t - p_t| > \epsilon) = 0$$

and the proof is completed.


AAMAS'04, July 19-23, 2004, New York, New York, USA.


### 3.4. Generalization of Q-learning

In our scenario, the agent's payoff depends not only on the actions it takes, but on the actions of other agents as well. Since the other agents are not assumed to be stationary, we cannot adopt the common approach, which assumes that other (stationary) agents are part of the extended environment, and the learning problem is reduced to learning in an MDP instead of in a stochastic game. Therefore, when talking about an optimal policy for our agent, we have to explicitly take into account the strategies of other agents in the environment. We define the optimal Q-values for our agent, with respect to the other $n-1$ agents in the environment, as:

$$\begin{aligned} &Q_*(s,a^1,\ldots,a^n) = r(s,a^1,\ldots,a^n) + \\ &\quad \beta \cdot \textstyle\sum_{s'} p(s'|s,a^1,\ldots,a^n) \cdot v(s',\pi^1_{br},\pi^2_*\ldots\pi^n_*) \end{aligned}$$

where $v(s',\pi^1_{br},\pi^2_*\ldots\pi^n_*)$ is our agent's total discounted reward over infinite periods starting from $s'$, given that the other agents follow strategies $(\pi^2_*,\ldots\,\pi^n_*)$ (the limits), and $\pi^1_{br}$ is the best response strategy to $(\pi^2_*,\ldots\,\pi^n_*)$.

Our learning algorithm maintains a table of Q-values $Q(s,a^1,\ldots\,a^n)$ for each state $s$ and joint action $(a^1,\ldots\,a^n)$. The Q-values are updated using the following rule.

$$\begin{aligned} &Q_{t+1}(s,a^1,\ldots,a^n) = (1-\alpha_t)\cdot Q_t(s,a^1,\ldots,a^n) + \\ &\quad \alpha_t \cdot [r_t + \beta \cdot \max_{\pi^1_{br}} \textstyle\sum_{a^1}\sum_{a^2}\ldots\sum_{a^n} \pi^1_{br}(s',a^1) \cdot \\ &\qquad \cdot \textstyle\prod_{i=2}^{n} \hat{\pi}^i_t(s',a^i) \cdot Q_t(s',a^1,\ldots,a^n)] \end{aligned}$$

where $\hat{\pi}^i_t$ is our agent's model of opponent agent $i$ and $\pi^1_{br}$ is our agent's best-response strategy rule against $(\hat{\pi}^2_t\ldots\hat{\pi}^n_t)$. Our agent uses this strategy rule for choosing the next action in the game.

Our proof of convergence is based on the following lemma by Szepesvari & Littman [15]. Before we get to the formal proof, we first explain some notations used in Lemma 2. $\{Q\}$ denotes a set of all Q-functions $Q : S \times A_1 \times \ldots \times A_n \to \Re$ and $P_t$ maps each Q-function to some other Q-function. The norm $\|\cdot\|$ of a Q-function is the supremum norm defined as

$$\|Q\| = \sup_{(s,a_1,\ldots,a_n)} Q(s,a_1,\ldots,a_n) < \infty$$

**Lemma 2** *Assume that $\alpha_t$ satisfies the typical assumptions for Q-learning, and the mapping $P_t : \{Q\} \to \{Q\}$ satisfies the following condition: there exists a number $0 < \gamma < 1$, and a sequence $\lambda_t > 0$ converging to zero with probability 1, such that $\|P_tQ - P_tQ_*\| \le \gamma \cdot \|Q - Q_*\| + \lambda_t$ for all $Q \in \{Q\}$ and $Q_* = E[P_tQ_*]$, then the iteration defined by*

$$Q_{t+1} = (1-\alpha_t)\cdot Q_t + \alpha_t \cdot [P_tQ_t]$$

*converges to $Q_*$ with probability 1.*

**Proof:** Corollary 5 by Szepesvari & Littman [15].

**Lemma 3** *Our update rule of Q-values converges to the optimal $Q_*$-values with probability 1.*

**Proof:** Our proof is based on Lemma 2. We first show that our update rule satisfies $\|P_tQ - P_tQ_*\| \leq \gamma \cdot \|Q - Q_*\| + \lambda_t$. We will use the following definition, to make the proof presentation more concise.

$$BR(\pi \in \{\pi_*, \pi_t, \hat{\pi}_t\}, Q \in \{Q_t, Q_*\}) = \max_{\pi^1_{br}} \sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi^1_{br}(s', a^1) \cdot \prod_{i=2}^{n} \pi^i(s', a^i) \cdot Q(s', \vec{a})$$

This denotes our best-response reward with respect to other agents' policy $\pi_*, \pi_t$ or $\hat{\pi}_t$ and Q-function $Q_t$ or $Q_*$. For example, $BR(\hat{\pi}_t, Q_t)$ is the maximum discounted reward that we would get from state $s'$ with regard to our current state-action evaluation $Q_t(s', a^1, \ldots, a^n)$ and our models of the other agent's real strategies $\hat{\pi}^i_t$.
In our definition,

$$P_tQ_t(s, a^1, \ldots, a^n) = r_t + \beta \cdot BR(\hat{\pi}_t, Q_t).$$

We also have

$$P_tQ_*(s, a^1, \ldots, a^n) = r_t + \beta \cdot BR(\pi_*, Q_*).$$

The proof: since Q-functions are defined over a finite set of state-actions tuples, $sup_{(s,\vec{a})}Q = max_{(s,\vec{a})}Q$

$$\|P_tQ_t - P_tQ_*\| = \max_{(s,\vec{a})} |P_tQ_t(s, \vec{a}) - P_tQ_*(s, \vec{a})|$$

and

$$\begin{aligned} &|P_tQ_t(s, \vec{a}) - P_tQ_*(s, \vec{a})| \\ &= |(r_t + \beta \cdot BR(\hat{\pi}_t, Q_t) - (r_t + \beta \cdot BR(\pi_*, Q_*)| \\ &= \beta \cdot |BR(\hat{\pi}_t, Q_t) - BR(\pi_*, Q_*)| \end{aligned}$$

Since $\hat{\pi}^i_t(s', a^i)$ converges in probability to $\pi^i_t(s', a^i)$, we replace each $\hat{\pi}^i_t(s', a^i)$ with $\pi^i_t(s', a^i) + \hat{\epsilon}^i_t$, and we get that the above term is less than

$$\begin{aligned} &\leq \beta \cdot |BR(\pi_t, Q_t) - BR(\pi_*, Q_*) + \\ &\quad \max_{\pi^1_{br}} \sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi^1_{br}(s', a^1) \cdot \\ &\quad \prod_{i=2}^{n} \hat{\epsilon}^i_t(s', a^i) \cdot Q_t(s', \vec{a})| \\ &\leq \beta \cdot |BR(\pi_t, Q_t) - BR(\pi_*, Q_*)| + \\ &\quad \beta \cdot |\max_{\pi^1_{br}} \sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi^1_{br}(s', a^1) \\ &\quad \cdot \prod_{i=2}^{n} \hat{\epsilon}^i_t(s', a^i) \cdot Q_t(s', \vec{a})| \\ &= |BR(\pi_t, Q_t) - BR(\pi_*, Q_*)| + \lambda^1_t \end{aligned}$$

where $\lambda^1_t = |\max_{\pi^1_{br}} \sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi^1_{br}(s', a^1) \cdot \prod_{i=2}^{n} \hat{\epsilon}^i_t(s', a^i) \cdot Q_t(s', \vec{a})|$. Since $\forall i \quad \hat{\pi}^i_t \to \pi^i_t$ with probability 1 as $t \to \infty$, we get that $\forall i \quad \hat{\epsilon}^i_t \to 0$ as $t \to \infty$, and then $\lambda^1_t \to 0$ with probability 1 as $t \to \infty$.

Now we look at $|BR(\pi_t, Q_t) - BR(\pi_*, Q_*)|$. Since $\pi^i_t(s', a^i)$ converges in probability to $\pi^i_*(s', a^i)$, we replace each $\pi^i_t(s', a^i)$ with $\pi^i_*(s', a^i) + \epsilon^j_t$, and we get that $|BR(\pi_t, Q_t) - BR(\pi_*, Q_*)|$ is less than

$$|BR(\pi_*, Q_t) - BR(\pi_*, Q_*)| + \lambda^2_t$$

where $\lambda^2_t = |\max_{\pi^1_{br}} \sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi^1_{br}(s', a^1) \cdot \prod_{i=2}^{n} \epsilon^i_t(s', a^i) \cdot Q_t(s', \vec{a})| \to 0$ with probability 1. Now assume (without loss of generality[2]) that $BR(\pi_*, Q_t) \geq BR(\pi_*, Q_*)$ and that $\pi^1_{br}$ maximizes $BR(\pi_*, Q_t)$. We put $\pi^1_{br}$ in $BR(\pi_*, Q_*)$ and get

$$\begin{aligned} &BR(\pi_*, Q_t) - BR(\pi_*, Q_*) \leq \\ &\sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi^1_{br}(s', a^1) \cdot \prod_{i=2}^{n} \pi^i_*(s', a^i) \\ &\cdot Q_t(s', \vec{a}) \\ &- \sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi^1_{br}(s', a^1) \cdot \prod_{i=2}^{n} \pi^i_*(s', a^i) \\ &\cdot Q_*(s', \vec{a}) \\ &= \sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi^1_{br}(s', a^1) \cdot \prod_{i=2}^{n} \pi^i_*(s', a^i) \\ &\cdot (Q_t(s', \vec{a}) - Q_*(s', \vec{a})) \\ &\leq \sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi^1_{br}(s', a^1) \cdot \prod_{i=2}^{n} \pi^i_*(s', a^i) \\ &\cdot \left[\max_{(s', \vec{a})} (Q_t(s', \vec{a}) - Q_*(s', \vec{a}))\right] \\ &= \left[\max_{(s', \vec{a})} (Q_t(s', \vec{a}) - Q_*(s', \vec{a}))\right] \cdot \\ &\cdot \sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi^1_{br}(s', a^1) \cdot \prod_{i=2}^{n} \pi^i_*(s', a^i) \\ &= \max_{(s', \vec{a})} (Q_t(s', \vec{a}) - Q_*(s', \vec{a})) \cdot 1 \\ &\leq \max_{(s', \vec{a})} |Q_t(s', \vec{a}) - Q_*(s', \vec{a})| \\ &= \|Q_t - Q_*\| \end{aligned}$$

In conclusion, we get that $\|P_tQ - P_tQ_*\| \leq \beta \cdot \|Q_t - Q_*\| + \lambda^1_t + \lambda^2_t$ and $\lambda_t = (\lambda^1_t + \lambda^2_t) \to 0$ with probability 1 as $t \to \infty$, so the first condition of Lemma 1 is satisfied.

We now show that $Q_* = E[PQ_*]$. By the definition of $Q_*$ we have:

$$\begin{aligned} &Q_*(s, a^1, \ldots, a^n) = r(s, a^1, \ldots, a^n) + \\ &\quad \beta \cdot \sum_{s'} p(s'|s, a^1, \ldots, a^n) \cdot v(s', \pi^1_{br}, \pi^2_* \ldots \pi^n_*) \\ &= r(s, a^1, \ldots, a^n) + \beta \cdot E_{s'}[v(s', \pi^1_{br}, \pi^2_* \ldots \pi^n_*)] \end{aligned}$$

On the other side, we have:

$$\begin{aligned} &E[PQ_*(s, a^1, \ldots, a^n)] = \\ &\quad E[r(s, a^1, \ldots, a^n) + \beta \cdot BR(\pi_*, Q_*)] \\ &= r(s, a^1, \ldots, a^n) + \beta \cdot E_{s'}[BR(\pi_*, Q_*)] \end{aligned}$$

2 If $BR(\pi_*, Q_*) \geq BR(\pi_*, Q_t)$, we put the $\pi^1_{br}$ that maximizes $BR(\pi_*, Q_*)$ in $BR(\pi_*, Q_t)$


AAMAS'04, July 19-23, 2004, New York, New York, USA.

Since $BR(\pi_*, Q_*)$ is the maximum discounted reward with regard to $\pi_*^i$ and $Q_*$, it is actually equal to $v(s', \pi_{br}^1, \pi_*^2 \ldots \pi_*^n)$, and thus we get that $Q_* = E[PQ_*]$. The proof is completed.

### 3.5. NSCP-learner Skeleton

In this subsection, we describe the full algorithm, combining the modeling and Q-value learning algorithms.

1. Init: $\forall s \in S$ and $a \in A$ and $i \in N$, $\quad \hat{\pi}_0^i(s,a) = \frac{1}{|A|}$
2. $\forall s \in S$ and $a \in A$ and $i \in N$, initialize $Q_0(s, a^1, \ldots, a^n)$ to any value
3. For $k = 1$ to $L$ do (during epoch)
   (a) Observe the actions taken by other agents $(a_t^2, \ldots, a_t^n)$, new state $s'$, and reward $r_t$
   (b) Update the other agents' models $(\hat{\pi}_t^2, \ldots \hat{\pi}_t^n)$
   (c) Select the best-response strategy $\pi_{br}^1$ that maximizes
   $$\sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi_{br}^1(s', a^1) \cdot \cdot \prod_{i=2}^n \hat{\pi}_t^i(s', a^i) \cdot Q_t(s', \vec{a})]$$
   (d) Update Q-values using the following rule:
   $$Q_{t+1}(s, a^1, \ldots, a^n) = (1 - \alpha_t) \cdot Q_t(s, \vec{a}) + \alpha_t \cdot [r_t + \beta \cdot \textstyle\sum_{a^1} \sum_{a^2} \cdots \sum_{a^n} \pi_{br}^1(s', a^1) \cdot \cdot \prod_{i=2}^n \hat{\pi}_t^i(s', a^i) \cdot Q_t(s', \vec{a})]$$
   (e) Choose the next action for state $s'$ according to $\pi_{br}^1$
4. Go to 3

## 4. Example of a Non-Stationary Opponent

We here describe an example algorithm for a non-stationary opponent. The algorithm may adapt, and chooses the next strategy according to previous history. Theoretically, the algorithm never settles down to a fixed stationary strategy.

1. Initialize: $\forall s \in S$ and $a \in A$, $\quad \pi_0(s,a) = \frac{1}{|A|}$
2. Initialize: $\epsilon_0 = 1, t = 0$
3. Play epoch $t$ using strategy $\pi_t$ and observe history $H_t$
4. Choose strategy for next epoch: $\pi_{t+1} = argmax_\pi(Reward(H_t, \pi) \quad | \quad |\pi - \pi_t| < \epsilon_t)$
5. $\epsilon_{t+1} = \frac{\epsilon_t}{2}$
6. $t = t + 1$
7. Go to 3


AAMAS'04, July 19-23, 2004, New York, New York, USA.


$Reward(H_t, \pi)$ denotes the reward that the agent would have received if he had played the strategy $\pi$ in the previous epoch. This algorithm uses the following policy: the strategy for the first epoch is, "for every state, use the uniform distribution over actions." After it observes the history of epoch $t$, it chooses for the next epoch the strategy that maximizes the reward for the previous epoch. In this manner, the algorithm may adapt to the game history. However, there is a restriction on the set of strategies from which the next strategy is chosen: the next strategy has a distance of at most $\epsilon_t$ from the previous strategy, and $\epsilon_t$ is getting smaller after each epoch. Note that there is no finite time $t$ when $\epsilon_t$ becomes equal to zero, so theoretically the opponent may always choose $\pi_{t+1} \neq \pi_t$. It is clear that this algorithm changes the agent's strategy, so the agent's policy is not stationary.

**Proposition 1** *The NSCP-learner would find the optimal policy against the non-stationary opponent described above.*

**Proof:** The proof is based on the following lemma. All we have to show is that the sequence of strategies chosen by the opponent's algorithm has a limit.

**Lemma 4** *(Cauchy Criterion) The sequence $x_n$ converges to some value if and only if the following holds: for every $\epsilon > 0$ we can find $K$ such that $|x_n - x_m| < \epsilon$ whenever $n, m > K$.*

Since $|\pi_{t+1} - \pi_t| < \epsilon_t$ and $\epsilon_t \to 0$, it is clear that the Cauchy Criterion is satisfied and the opponent's strategy sequence has a limit.

The fact that this non-stationary opponent may also learn, and its converging strategy depends on the behavior of our NSCP-learner, may at first seem confusing. It creates a "learning loop" in which both agents eventually converge. The opponent agent's strategy converges since its learning rate is forcibly decreased. The NSCP-learner converges since it did not make any assumptions about the reason for the opponent's convergence and the actual limit to which it converges. Thus, either the opponent agent tries to adapt or it chooses its actions independently of the NSCP-learner's actions — if its strategy converges in the limit, NSCP-learner would find the best response policy to that strategy.

## 5. Experimental Results

In this section, we present several experimental results regarding our algorithm.

### 5.1. Pursuit Problem

Experimental results were obtained in the pursuit domain [1], which has been widely examined in multiagent

system research. We used the following variant of the pursuit problem:

- A $5 \times 5$ grid world with 1 predator agent and 1 prey agent. The grid world with the agents and the directions in which they can move is shown in Figure 5.1.
- The goal of the predators is to hunt the prey and the goal of the prey is to escape from the predators. Our predator agent gets a reward of 1 when it captures prey.
- The game is divided into discrete time steps, at which both agents simultaneously choose and perform one of the following actions: $A = \{Left, Right, Down, Up, Stay\}$.
- The state of the environment perceived by both agents is the location of both agents in the grid world. For example, the state in Figure 5.1 is represented as $s = \{(0, 2); (3, 3)\}$.
- The game is finished when the predator is in the same location as the prey.
- At each time step, the prey agent has a small constant probability of choosing $Stay$. This models the situation in which a predator is a bit faster than the prey.

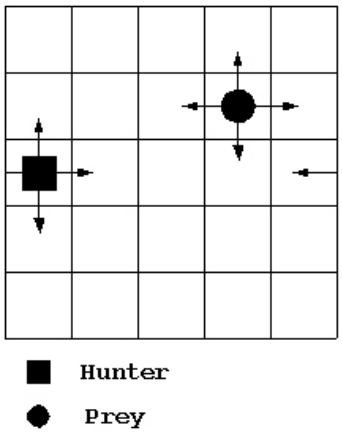


**Figure 1. A pursuit problem in a grid world: arrows denote the directions in which hunter/prey can move**

### 5.2. Experiments

In our experiments we compared our NSCP-learner algorithm with the usual Q-learning algorithm. The comparison is in terms of the number of steps until the prey is caught, as a function of the number of learning iterations. In each game the predator and the prey are positioned in the predefined positions and the game runs until the prey is caught. Every such game constitutes one learning iteration (or episode) and the learned Q-values were passed from iteration to iteration.

The prey agent has the following non-stationary policy: it starts with a strategy of moving $Left$ with probability 1 and zero probability for all other actions. The probability mass slowly moves towards 1 on choosing $Up$ and zero for all other actions. Although this policy is very simple and does not change according to the history of the game, it is still non-stationary.

We ran the game 2000 times during which Q-learner and NSCP-learner were learning. The prey changed his policy during the first 1000 games and kept a constant policy during the last 1000 games. We then repeated this experiment 10 times in order to obtain the average time needed to catch prey as a function of the number of learning episodes. Figure 5.2 shows the comparison between Q-learning and NSCP-learning. In general, for both algorithms it took more moves to catch the prey when the latter's strategy was changing. However, NSCP-learner outperformed Q-learner in both periods: when the prey's strategy was changing (during the first 1000 games), and when it stopped changing (during the last 1000 games).

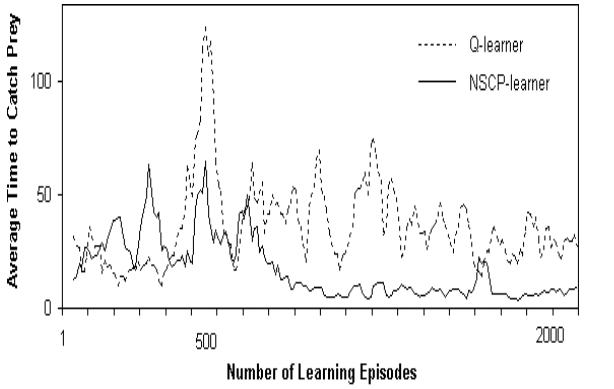


**Figure 2. Q-learner vs. NSCP-learner in presence of a non-stationary agent**

## 6. Best-Response for Other Agent Classes: a Discussion

We here briefly discuss the development of a Best-Response learning algorithm for richer classes of agents. Although our learning algorithm is optimal for environments with any number of agents with any converging policies, it still covers a very restricted class of agents.


AAMAS'04, July 19-23, 2004, New York, New York, USA.

It is important to mention that finding the best-response strategy for any given opponent is a very hard problem. Nachbar [10] shows that even in two-player stochastic games (and in particular, in the discounted repeated game of Prisoner's Dilemma), there exist strategies for which no best response is implementable by a Turing machine. It means that we should not hope to find a single best-response algorithm which would be optimal for all classes of opponent agents.

Thus, we return to the discussion about agent classes. When we speak about agents with converging policies, we define the notion of best-response with regard to the limit of the policy. However, this is impossible when we start to speak about agents with non-converging policies. These agents constantly change their strategy, and never converge to a fixed stationary strategy (for example, because their learning process never terminates). This scenario is realistic in a rich, dynamic environment inhabited by autonomous, selfish learning agents. In this case, our learning agent should not converge to a stationary policy, but rather should be ready to adapt indefinitely to new agents and new occurrences in the environment. The learning process should try to track a moving learning goal, and for this to be meaningful we have to define new evaluation criteria for such learning processes.

## 7. Conclusions and Future Work

We presented a multiagent learning algorithm for general-sum stochastic games which is optimal (best-response) in the presence of other agents with converging policies. The algorithm explicitly models the other agents in the environment and learns a best-response policy which is proved to be optimal with regard to the limits of the other agents' policies. We also showed experimental results comparing our learning algorithm to Q-learning.

One direction for future work includes further investigation of this algorithm, performing intensive experiments in additional frameworks and in the presence of other non-stationary agents. Another direction is defining new meaningful classes of agents and designing best-response learning algorithms for these classes. In addition, new evaluation criteria should be defined for algorithms that do not converge to a stationary strategy, but view the learning task as a constantly moving target.

## References


[1] M. Benda, V. Jagannathan, and R. Dodhiawala. On optimal cooperation of knowledge sources — an empirical investigation. Technical Report BCS-G2010-28, Boeing Advanced Technology Center, Boeing Computer Services, Seattle, Washington, July 1986.

[2] M. Bowling and M. Veloso. Rational and convergent learning in stochastic games. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 1021–1026, Seattle, Washington, August 2001.

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

[4] V. Conitzer and T. Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 83–90, 2003.

[5] A. Greenwald and K. Hall. Correlated Q-learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 242–249, 2003.

[6] J. Hu. Best-response algorithm for multiagent reinforcement learning. 2003.

[7] J. Hu and M. Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[8] M. L. Littman. Markov games as a framework for multiagent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, New Brunswick, NJ, 1994.

[9] M. L. Littman. Friend-or-foe: Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322–328, 2001.

[10] J. H. Nachbar and W. R. Zame. Non-computable strategies and discounted repeated games. *Economic Theory*, 8:103–122, 1996.

[11] S. Sen and G. Weiss. Learning in multiagent systems. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter 6. MIT Press, Cambridge, MA, 1999.

[12] Y. Shoham, R. Powers, and T. Grenager. Multi-agent reinforcement learning: a critical survey. Technical report, Computer Science Department, Stanford University, Stanford, 2003.

[13] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. Technical Report CMU-CS-97-193, School of Computer Science, Carnegie Mellon University, 1997.

[14] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[15] C. Szepesvri and M. L. Littman. A unified analysis of value-function-based reinforcement learning algorithms. *Neural Computation*, 11:8:2017–2059, 1999.

[16] C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3):279–292, 1992.




AAMAS'04, July 19-23, 2004, New York, New York, USA.