Learning to Compete, Compromise, and Cooperate in Repeated General-Sum Games

Jacob W. Crandall CRANDALL@CS.BYU.EDU
Michael A. Goodrich MIKE@CS.BYU.EDU
Computer Science Department, Brigham Young University, Provo, UT 84602 USA

Abstract

Learning algorithms often obtain relatively low average payoffs in repeated general-sum games between other learning agents due to a focus on myopic best-response and one-shot Nash equilibrium (NE) strategies. A less myopic approach places focus on NEs of the repeated game, which suggests that (at the least) a learning agent should possess two properties. First, an agent should never learn to play a strategy that produces average payoffs less than the minimax value of the game. Second, an agent should learn to cooperate/compromise when beneficial. No learning algorithm from the literature is known to possess both of these properties. We present a reinforcement learning algorithm (M-Qubed) that provably satisfies the first property and empirically displays (in self play) the second property in a wide range of games.

1. Introduction

Artificial agents should act intelligently in complex and unknown environments, including environments influenced by other learning agents. Many of these environments can be modeled as repeated general-sum matrix games, which may be fully cooperative, fully competitive, or somewhere in between. Many learning algorithms perform well in various classes of these games, yet none of these algorithms perform well in general-sum matrix games as a whole.

One reason that learning algorithms have been unable to learn strategies that produce high average payoffs in many repeated general-sum games is that they have focused on one-shot game theoretic concepts, which has led to a focus on learning (myopic) best-response strategies. While appropriate in many games, game theory itself suggests, through the folk theorem, that these approaches do not produce satisfactory results in many other games.

The folk theorem suggests two properties that a successful learner should possess in repeated games. First, a learner should never learn a strategy that produces expected payoffs lower than the minimax value of the game. Second, a learner should learn to make and accept compromises that increase its average payoffs.

In this paper, we present a reinforcement learning algorithm (called M-Qubed) that provably satisfies the first property and empirically displays (in self play) the second property in a wide range of games.

2. Definitions

In this section, we give some definitions and notation. Let S denote the set of states in the game and let A_i denote the set of actions that agent/player i may select in each state $s \in S$. Let $a = (a_1, a_2, \ldots, a_n)$, where $a_i \in A_i$, be a *joint action* for n agents, and let $A = A_1 \times \cdots \times A_n$ be the set of possible joint actions.

A *strategy* (or *policy*) for agent i is a probability distribution $\pi_i(\cdot)$ over its action set A_i. Let $\pi_i(S)$ denote a strategy over all states $s \in S$ and let $\pi_i(s)$ (or π_i) denote a strategy in a single state s. A strategy may be a *pure strategy* (the agent selects an action deterministically) or a *mixed strategy* (otherwise). A *joint strategy* played by n agents is denoted by $\pi = (\pi_1, \ldots, \pi_n)$. Also, let a_{-i} and π_{-i} refer to the joint action and strategy of all agents except agent i.

Central to game theory is the matrix game, defined by a set of matrices $R = \{R_1, \ldots, R_n\}$. Let $R(\pi) = (R_1(\pi), \ldots, R_n(\pi))$ be the vector of expected payoffs when the joint strategy π is played. Also, let $R_i(\pi_i, \pi_{-i})$ be the expected payoff to agent i when it plays strategy π_i and the other agents play π_{-i}. A

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

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

(a) Rock, Paper, Scissors

	a	b	c
a	0, 0	0, 1	1, 0
b	1, 0	0, 0	0, 1
c	0, 1	1, 0	0, 0

(b) Shapley's Game

	a	b
a	1, -1	-1, 1
b	-1, 1	1, -1

(c) Matching Pennies

	a	b
a	2, 2	0, 0
b	0, 0	4, 4

(d) Coordination Game

	a	b
a	2, 2	3, -5
b	-5, 3	4, 4

(e) Stag Hunt

	c	d
c	3, 3	0, 5
d	5, 0	1, 1

(f) Prisoner's Dilemma

	c	d
c	3, 3	2, 3.5
d	3.5, 2	1, 1

(g) Chicken

	a	b
a	0, 3	3, 2
b	1, 0	2, 1

(h) Tricky Game

Figure 1. Payoff matrices for various games (zero-sum, cooperative, and conflicting interest).

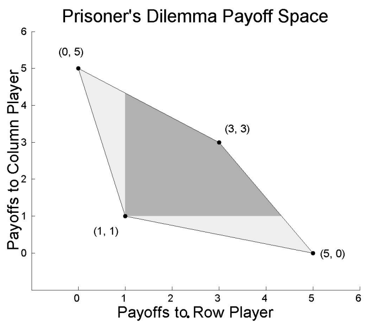


Figure 2. Payoff space of the prisoner's dilemma game shown in Figure 1(f).

stage game is a single iteration of a matrix game, and a *repeated matrix game* is the indefinite repetition of the stage game between the same agents.

Figure 1 shows several two-player matrix games. In the figure, the payoff to the row player is followed by the payoff to the column player. For example, if the row player plays `c` and the column player plays `d` in (f), then the row player gets a payoff of 0 and the column player gets a payoff of 5.

Each matrix game has certain game theoretic values. The *minimax value* for agent i is $m_i = \max_{\pi_i} \min_{a_{-i}} R_i(\pi_i, a_{-i})$. The *minimax strategy* for agent i is $\pi_i^m = \text{argmax}_{\pi_i} \min_{a_{-i}} R_i(\pi_i, a_{-i})$. A *one-shot NE* is a joint strategy π such that no agent can unilaterally change its strategy without lowering its expected payoff in the current stage. Nash (1951) showed that every n-player matrix game has at least one one-shot NE. An *NE of the repeated game* (rNE) is a joint strategy $\pi(S)$ over all states $s \in S$, such that no agent can unilaterally change its strategy in any state $s \in S$ without lowering its expected average payoff over time. Every one-shot NE is an rNE.

While matrix games do not have state, agents can benefit from using the previous w joint actions taken by the agents as state (see (Sandholm & Crites, 1995)). Unless otherwise stated, we use $w = 1$. Thus, if agent 1 plays `c` and agent 2 plays `d` at time t, then the state at time $t + 1$ (s_{t+1}) is `cd`.

We assume that an agent can observe its own payoffs as well as the actions taken by all agents. Thus, an agent can observe its own payoff matrix R_i and can calculate its minimax value m_i and minimax strategy π_i^m using linear programming. Since an agent knows its payoff matrix R_i it also knows its highest and lowest possible expected payoffs (denoted $h_\$$ and $l_\$$ respectively).

3. Motivation

Most multiagent learning algorithms to date have focused on learning a (myopic) best response to the strategies of other agents (e.g. (Fudenberg & Levine, 1998; Bowling & Veloso, 2002)). Play between learning agents using such approaches often converges to a one-shot NE. However, a major result from game theory (the folk theorem) suggests that the goal of reaching a one-shot NE may be inappropriate in repeated games. In this section, we briefly review the folk theorem (which is presented in more detail in (Gintis, 2000)) and some of its implications.

3.1. The Folk Theorem

Consider the prisoner's dilemma game shown in Figure 1(f). The joint payoff space of this game is shown in Figure 2, where the x and y axes show the payoffs to the row and column players respectively. Convex combinations of the various joint payoffs of the game form the game's payoff space (convex hull), which is depicted by the union of the shaded regions (light and dark) in the figure. By playing `d`, each player can guarantee itself a payoff at least as great as its minimax value,[1] so neither player has an incentive to receive an average payoff per stage less than 1. Thus, the darkly shaded region of the convex hull shows the set of average joint payoffs that the agents may be content to receive. The folk theorem says that any joint payoff in this region can be sustained by an rNE, provided that the players believe that play will continue with high probability after each stage of the game.

More formally, let C be the set of points in the convex hull of the game. Any joint payoff $R(\pi) = (R_1(\pi), R_2(\pi)) \in C$ such that $R_1(\pi) \geq m_1$ and $R_2(\pi) \geq m_2$ can be sustained as an rNE provided that the discount rates of the players are close to unity. The folk theorem can be extended to n-player games.

[1]In the prisoner's dilemma, the minimax solution is a one-shot NE, but this is not always true in the general case.

3.2. Implications

The folk theorem implies that, in many games, there exist rNEs that yield higher payoffs to all agents than do one-shot NEs. Thus, the one-shot NE should not be the goal of learning agents. Rather, in repeated games between other successful learning agents, a successful agent should learn to play profitable rNEs.

Since many repeated games have an infinite number of rNEs, the folk theorem does little to indicate which one the agents should play. Littman and Stone (2003) give an algorithm for computing an rNE that satisfies a set of desiderata, but how to learn this strategy is unknown. Also, an agent may want to play a more beneficial (for it) rNE if allowed by its associate(s).

Additionally, in games between learning agents, it is difficult for a learner to determine if it is playing its portion of an rNE. Because of this, we seek to design an agent that learns to play strategies that yield average payoffs that correspond to an rNE, preferably an rNE with a high payoff. Thus, a learner should possess (at the least) the following two properties.

Property 1. *(Security Property) An agent should not learn to play strategies that produce average expected payoffs below its minimax value m_i.*

An agent can easily satisfy this property by playing π_i^m regardless of what its associates do. However, such a strategy may be "irrational," since m_i may be much lower than the payoffs that an agent can realistically expect. On the other hand, m_i is the highest expected payoff that any agent can guarantee itself without some form of cooperation or compromise from its associate(s). Thus, a successful agent must learn (if possible) to influence its associate(s) to be cooperative, which brings us to the second property.

Let $\Omega_i^t(\pi_{-i}^t(S)) = \{\pi_i(S) : R_i(\pi_i(S), \pi_{-i}^t(S)) > m_i\}$ and let $\Delta_{-i}(S) = \{\pi_{-i}(S) : \Omega_i(\pi_{-i}(S)) \neq \emptyset\}$. That is, $\Omega_i^t(\pi_{-i}^t(S))$ is the set of all strategies for agent i (at time t) that will give it an expected payoff greater than its minimax value given the other agents' strategies $\pi_{-i}^t(S)$; and $\Delta_{-i}(S)$ is the set of associate strategies for which Ω_i^t is not empty. Also, let $\varphi_i^T = (a_i^0, a_i^1, \dots a_i^T)$ be an action sequence through time T for agent i, where a_i^t is agent i's action at time t. Let $\Phi_i = \{\varphi_i^T : \forall t \geq T, \pi_{-i}^t(S) \in \Delta_{-i}(S) \text{ if } \pi_i^t(S) \in \Omega_i^t(\pi_{-i}^t(S))\}$. That is, Φ_i is the set of action sequences that agent i may take that leads to it receiving an expected payoff greater than its minimax value from time T onward.

Property 2. *(Compromise/Cooperate Property) This property has two parts:* influence *and* compromise.

- Influence. *If $\Phi_i \neq \emptyset$, then an agent should play $\varphi_i \in \Phi_i$ prior to time T and $\pi_i^t(S) \in \Omega_i^t(\pi_{-i}^t(S))$ thereafter.*
- Compromise. *An agent can be* influenced *in non-competitive games.*

We refer to this property as the *compromise/cooperate (C/C) property* because an agent must offer and accept compromises to possess it. In words, *influence* says that if an agent can influence its associate(s) to play cooperatively, it will. While this may be impractical without omniscience, a successful agent should satisfy this property when associating with a large class of learning agents in a large class of games. *Compromise* says that an agent can be induced to cooperate/compromise. Therefore, an agent must be able to compromise in self play to satisfy this property.

To better motivate the implications of these properties, we give several examples from the three classes of repeated general-sum games: competitive games, cooperative games, and games of conflicting interest.

3.2.1. Competitive Games

Competitive games have no solution in which all agents receive an average payoff greater than their minimax value m_i. Figures 1(a) and (c) are examples of such games. Both of these games have a single rNE, which corresponds to each game's minimax solution. The security property implies that an agent should receive an average payoff of m_i in these games (provided, of course, that its opponent is unexploitable).

3.2.2. Cooperative Games

Figure 1(d) shows a fully cooperative game with two one-shot NEs (when the agents both play the same action) and an infinite number of rNEs. The rNE in which both agents repeatedly play `b` is the most desirable outcome. A related game is shown in Figure 1(e), in which action `b` is risky, yet still corresponds to the most profitable rNE. The C/C property implies that successful learners should learn to play this risky solution (provided that their associate will cooperate).

3.2.3. Games of Conflicting Interest

Figures 1(b), (f)-(h) show games of conflicting interest. In these games, offering and accepting compromises can produce expected payoffs that are higher than those obtained by playing a one-shot NE. These compromises often require that agents resort to alternating between "winning" and "losing." For example, in *Shapley's game*, the one-shot NE is for both agents to play completely randomly, which yields an expected payoff of $\frac{1}{3}$ to both agents. However, if the agents take turns "winning" (i.e., alternating between payoffs of 0 and 1) they both receive an average payoff of $\frac{1}{2}$.

Learning Algorithm	Security	C/C (Self Play)
Q-Learning	No	No
Fictitious Play (Fudenberg & Levine, 1998)	No	No
minimaxQ (Littman, 1994)	Yes	No
NashQ (Hu & Wellman, 1998)	No	No
FFQ (Littman, 2001)	Yes	In cooperative games
WoLF-PHC (Bowling & Veloso, 2002)	No	No
Satisficing Learning (Crandall & Goodrich, 2004)	No	In most non-competitive games
GIGA-WoLF (Bowling, 2004)	Yes	No

Table 1. Property characteristics of typical learners.

In *Chicken*, there are two pure strategy one-shot NEs which yield the payoffs $(3.5, 2)$ and $(2, 3.5)$. While there may be situations in which an agent should repeatedly accept the payoff of 2 (its minimax value), many situations exist in which a successful learner should enforce a mutually beneficial compromise (if it cannot get 3.5), such as both agents playing c.

3.3. Measuring Existing Algorithms

We know of no algorithm in the literature that possesses both properties. Table 1 lists typical learners and indicates which of the two properties they possess.[2] The *C/C* property is shown for self-play only.

4. The M-Qubed Algorithm

In this section, we describe a learning algorithm that (provably) satisfies the *security* property and appears (empirically) to possess the *C/C* property in self play. We call this algorithm M-Qubed (*M*ax or *M*ini*m*ax *Q*-learning = M^3-Q = M-Qubed).

Like Q-learning (Watkins, 1989), M-Qubed estimates the value of a state-action pair and generates a policy using this estimate. We describe the algorithm below.

4.1. Q-update

In Q-learning, the quality of a state-action pair $Q(s, a_i)$ is estimated by iteratively updating Q-values using the following equation:

$$Q_{t+1}(s, a_i) = (1 - \alpha)Q_t(s, a_i) + \alpha\left(r_i^t + \gamma V_t(s')\right), \quad (1)$$

where s is the current state, a_i is the action taken at time t, r_i^t is the reward receive at time t, and $V_t(s')$ is the estimated value of the next state s', given by $V_t(s') = V_t^Q(s') = \max_{b_i} Q_t(s', b_i)$.

Watkins showed that the estimate $Q_t(s, a_i)$ approaches the true Q-values as $t \to \infty$ in stationary environments provided that each state-action pair is taken infinitely often and the learning rate α is decreased appropriately (Watkins, 1989). However, since matrix games between learning agents are non-stationary environments, various algorithms have been proposed to make Q-learning appropriate for these situations. These approaches involve a) estimating the value of a state-joint action pair (rather than a state-action pair) and b) estimating the value of the next state $V_t(s')$ with a game theoretic value (e.g., (Littman, 1994; Hu & Wellman, 1998; Littman, 2001; Greenwald & Hall, 2003).

M-Qubed updates Q-values as does Q-learning except it uses

$$V_t(s') = \sum_{b_i} \pi_i(s', b_i) Q_t(s', b_i), \quad (2)$$

where $\pi_i(s', b_i)$ is the probability that the agent believes it will play action b_i in state s'.[3] Note that (2) is equivalent to $V_t^Q(s')$ if $\pi_i(s') = \text{argmax}_{b_i} Q_t(s', b_i)$.

4.2. Generating a Strategy

A Q-learner always (unless exploring) takes the action corresponding to its highest Q-value in the current state. However, in repeated matrix games, an agent's best strategy might not be a pure strategy. The trick is to know when to play a pure strategy, and when to play a mixed strategy. We advocate that agents should act predictably (pure strategies) in games of compromise and cooperation, but unpredictably (mixed strategies) in competitive games (if π_i^m is a mixed strategy). Since it is frequently unknown *a priori* whether a game is competitive or whether associates will be inclined to cooperate, a successful learning algorithm must determine whether to act predictably or unpredictably.

In this subsection, we state two strategy rules for determining whether M-Qubed should play predictably or unpredictably. Since each rule is advantageous for a different set of circumstances, we present a third rule which mixes the two rules in a desirable fashion.

In the first strategy rule, M-Qubed determines whether it should play predictably or unpredictably by comparing its highest Q-value in the current state ($\max_{a_i} Q_t(s, a_i)$) with the discounted sum of m_i ($\sum_{t=0}^{\infty} \gamma^t m_i = \frac{m_i}{1-\gamma}$). When $\max_{a_i} Q_t(s, a_i) > \frac{m_i}{1-\gamma}$, M-Qubed plays predictably. Otherwise, it plays π_i^m.

[2]Table 1 typifies algorithms for general-sum games. A large body of literature has been dedicated to cooperation in more specific contexts, such as the prisoner's dilemma (see, for example, (Sandholm & Crites, 1995)).

[3](2) makes M-Qubed's Q-update similar to that of Sarsa (Sutton & Barto, 1998).

Thus, M-Qubed's strategy in state s is

$$\pi_i^*(s) \leftarrow \begin{cases} \text{argmax}_{a_i} Q_t(s, a_i) & \text{if } \max_{a_i} Q_t(s, a_i) > \frac{m_i}{1-\gamma} \\ \pi_i^m & \text{otherwise} \end{cases} \quad (3)$$

Unfortunately, an agent may be exploited when using this rule since its Q-values might not reflect its actual payoffs when the environment is non-stationary.

A second rule for determining whether to play predictably or unpredictably is to compare the agent's average payoff $\mu_i^t = \frac{1}{t}\sum_{j=1}^{t} r_i^j$ with m_i. This gives

$$\pi_i^*(s) \leftarrow \begin{cases} \text{argmax}_{a_i} Q_t(s, a_i) & \text{if } \frac{\mu_i^t}{1-\gamma} > \frac{m_i}{1-\gamma} \\ \pi_i^m & \text{otherwise} \end{cases} \quad (4)$$

While safe, μ_i^t reacts too slowly. Thus, an agent using rule (4) might a) continue to play π_i^m when a more profitable strategy can be learned otherwise or b) be slow to play π_i^m when it is being temporarily exploited.

Thus, (3) allows an agent to explore possible improvements (C/C), while (4) provides an agent with security. A balance may be obtained by learning which of the two techniques is more successful. Let β_i^t be the probability that M-Qubed uses (4) at time t, and let $1 - \beta_i^t$ be the probability that it uses (3). Thus,

$$\pi_i^*(s) \leftarrow \begin{cases} \text{Rule (4)} & \text{with probability } \beta_t \\ \text{Rule (3)} & \text{with probability } 1 - \beta_t \end{cases} \quad (5)$$

β_i^t is determined using $\acute{\beta}_i^t$. Initially, M-Qubed optimistically sets $\acute{\beta}_i^0 = 0$, then updates it using

$$\acute{\beta}_i^{t+1} \leftarrow \begin{cases} \acute{\beta}_i^t + \lambda(r_i^t - m_i) & \text{Rule (4)} \\ \acute{\beta}_i^t - \lambda(r_i^t - m_i) & \text{Rule (3)} \end{cases} \quad (6)$$

where λ is some constant. In words, $\acute{\beta}_i^t$ determines which rule is more successful by increasing (toward rule (4)) if rule (4) yields payoffs greater than m_i or if rule (3) yields payoffs lower than m_i, and decreasing (toward rule (3)) otherwise. β_i^t is $\acute{\beta}_i^t$ constrained to the interval $[0, 1]$, or $\beta_i^t = \min(1, \max(0, \acute{\beta}_i^t))$.

4.3. Adding Exploration

Q-learners typically explore their environment using ε-greedy or Boltzmann exploration. In repeated matrix games, an alternate exploration method is to set initial value estimates optimistically (as suggested by Stimpson et al. (2001)) and let the learning process dictate exploration. M-Qubed does this by initializing each Q-value to its highest possible expected discounted reward, $\frac{h_\$}{1-\gamma}$. In this way, an agent uses a relaxation search to find a strategy that yields an expected discounted reward at least as high as its highest Q-value.

```
1) $\alpha, \eta \in [0,1]$, $\acute{\beta}_i = \beta_i = 0$.
2) Estimate $h_\$$, $l_\$$, $m_i$, and $\pi_i^m$ during $n$ observations.
3) Initialize $\pi_i(s, a_i) \leftarrow \frac{1}{|A_i|}$, $Q(s, a_i) \leftarrow \frac{h_\$}{1-\gamma}$
4) Repeat,
   a) Take action $a_i$ according to $\pi_i(s)$
   b) Observe reward $r$ and next state $s'$:
      i)   Update estimates of $m_i$, $\pi_i^m$, $h_\$$, and $l_\$$
      ii)  Update $\pi_i(s')$ using Eq. (7)
      iii) Update $Q(s, a_i)$ using Eqs. (1), (2)
      iv)  Update $\acute{\beta}_i$, $\beta_i$ using Eq. (6)
```

Table 2. The M-Qubed algorithm for player i.

Name	**Type**	**Parameter details**
M-Qubed	M-Qubed	$\alpha = 0.1$, $\gamma = 0.95$, $\eta = 0.04\left(\frac{20000}{20000+t}\right)$, $\lambda = \frac{0.01\alpha}{h_\$ - l_\$}$
QL	Q-learner	$\alpha = 1/(10 + 0.01\kappa_s^{a_i})$, $\gamma = 0.95$, ε-greedy w/ $\varepsilon = \max(0.2 - 0.0006t, 0)$
rQL	Q-learner	$\alpha = 1/(10 + 0.01\kappa_s^{a_i})$, $\gamma = 0.95$, ε-greedy w/ $\varepsilon = 0.1$
WoLF	WoLF-PHC	Same as QL, but with $\delta_w = 1/(10.0 + \kappa_s^{a_i})$, $\delta_l = 4\delta_w$

Table 3. Learners and their parameter values. $\kappa_s^{a_i}$ is the number of times that action a_i has been played in state s.

M-Qubed also randomizes its strategy with some small probability η if it perceives that its current strategy π_i^* keeps it in a local maximum. M-Qubed guesses that this is the case if it has not visited state s^* (a state with the highest global Q-value) in the previous $L = (\prod_i |A_i|)^w$ (the number of states in the games) stages of the game. Thus,

$$\pi_i(s, a_i) \leftarrow \begin{cases} \pi_i^*(s, a_i), & \text{if } s^* \in \{s_{t-L}, \cdots, s_t\} \\ (1-\eta)\pi^*(s, a_i) + \eta\frac{1}{|A_i|}, & \text{otherwise} \end{cases} \quad (7)$$

where $\pi_i^*(s, a_i)$ is obtained from (5) (unless stated otherwise). η is slowly decayed to 0 as $t \to \infty$.

Because M-Qubed can cease to explore, it can learn strategies that are not as desirable as it otherwise might have learned. However, we will show in the next section that continuing to explore in repeated games can be more costly than ceasing exploration.

5. Results

In this section, we prove that M-Qubed (summarized in Table 2) satisfies the security property and show that it displays the *C/C* property in a wide range of games (in self play). Unless stated otherwise, the parameter values used are as shown in Table 3.

5.1. Security Property

Before giving the main theorem, we give two lemmas related to the random walk of $\acute{\beta}_i^t$ since its behavior relates closely to the performance of M-Qubed. The first lemma shows that the interval $(0, 1)$ is a reflecting boundary for $\acute{\beta}_i^t$ when M-Qubed is being exploited. The second lemma states a property of reflected random walks.

Let $l_i^t = (m_i - r_i^t)$ be agent i's loss at time t, and let $L_i^t = \sum_{j=0}^{t-1} l_i^j$ be its accumulated loss prior to time t. Let $t \in [\tau, \rho]$ be an interval such that $\acute{\beta}_i^t \in (0, 1)$.

Lemma 1. *$E[\acute{\beta}_i^\rho - \acute{\beta}_i^\tau] > 0$ if M-Qubed is exploited during time $t \in [\tau, \rho]$.*

Proof. Let ξ_3 be the subset $t \in [\tau, \rho]$ in which M-Qubed uses strategy rule (3), and let ξ_4 be the subset $t \in [\tau, \rho]$ in which it uses strategy rule (4). Then M-Qubed's expected loss during $t \in [\tau, \rho]$ is

$$E[L_i^\rho - L_i^\tau] = \sum_{t=\tau}^{\rho} E[l_i^t] = \sum_{t \in \xi_3} E[l_i^t] + \sum_{t \in \xi_4} E[l_i^t]$$

When $\mu_i^t \leq m_i$, then $\sum_{t \in \xi_4} E[l_i^t] \leq 0$ since M-Qubed plays π_i^m.[4] Thus, the expected loss becomes $E[L_i^\rho - L_i^\tau] \leq \sum_{t \in \xi_3} E[l_i^t]$. Since $\acute{\beta}_i^t$ increases whenever $r_i^t < m_i$ and M-Qubed plays rule (3), $E[\acute{\beta}_i^\rho - \acute{\beta}_i^\tau] > 0$ if M-Qubed is exploited in $t \in [\tau, \rho]$. □

We state the next lemma without proof. Let W be a random walk in which $E[W_t - W_{t+1}] \geq 0$ provided that it encounters no boundaries. Let there be a reflecting boundary such that whenever $W_t < b$, $E[W_{t+T}] \geq b$ for some $T > 0$. Also, let $Pr(\cdot)$ denote a probability.

Lemma 2. *$Pr(W_t < b) \to 0$ as $t \to \infty$.*

Theorem 1. *M-Qubed satisfies the security property (in the limit) in all repeated general-sum matrix games, regardless of its associate(s).*

Proof. We will show that $\lim_{t\to\infty} L_i^t/t \leq 0$ in the worst case. The worst case can occur when playing against an omniscient opponent, a *seer agent*, which knows everything about M-Qubed including its strategy, but not its actual actions (*a priori*).

If $L_i^t \leq 0$ for all $t \geq T$, then $\lim_{t\to\infty} L_i^t/t \leq 0$. Otherwise, we must address three cases when $\mu_i^t < m_i$.

Case 1: $\acute{\beta}_i^t \leq 0$ for all $t \geq T$. Since $\beta_i^t = 0$ when $\acute{\beta}_i^t \leq 0$, M-Qubed uses strategy rule (3) with probability 1. Thus, (6) assures that $\acute{\beta}_i^t = \acute{\beta}_i^T - \lambda(L_i^T - L_i^t)$, so agent i's loss during $t \geq T$ is $L_i^t - L_i^T = \frac{\acute{\beta}_i^t - \acute{\beta}_i^T}{\lambda}$. Since $\acute{\beta}_i^t \leq 0$, $L_i^t - L_i^T \leq \frac{-\acute{\beta}_i^T}{\lambda}$ (a constant), so $\lim_{t\to\infty} L_i^t/t \leq 0$.

[4]Since $\eta \to 0$ as $t \to \infty$ we can assume $\eta = 0$.

Case 2: $\acute{\beta}_i^t \geq 1$ for all $t \geq T$. Using similar logic as case 1 (using (4) rather than (3)), $\lim_{t\to\infty} L_i^t/t \leq 0$.

Case 3: $\acute{\beta}_i^t \in (0, 1)$ for some arbitrary time interval(s) $t \in [\tau, \rho]$. Lemma 1 shows that $E[\acute{\beta}_i^\rho - \acute{\beta}_i^\tau]$ is proportional to M-Qubed's losses during the time $t \in [\tau, \rho]$. This means two things. First, a seer agent cannot increase M-Qubed's expected accumulated loss by more than a constant when $\acute{\beta}_i^t \in (0, 1)$. Second (in conjunction with cases 1 and 2), M-Qubed's expected accumulated loss can increase by no more than a constant except when $\acute{\beta}_i^t$ repeatedly crosses 0 or 1. Each time $\acute{\beta}_i^t$ crosses 0 or 1, M-Qubed can experience a small loss (bounded by $(m_i - l_\$)$). However, since M-Qubed uses rule (3) when $\acute{\beta}_i^t \leq 0$, $\acute{\beta}_i^t \leq 0$ is a reflecting boundary. Since $\acute{\beta}_i^t$ also satisfies the preconditions of Lemma 2 it follows from Lemma 2 that $Pr(\acute{\beta}_i^t < 0) \to 0$ as $t \to \infty$. Thus, $\acute{\beta}_i^t$ crosses 0 increasingly less frequently as $t \to \infty$. A similar argument can be made for $\acute{\beta}_i^t$ crossing 1 since Lemma 1 shows that $(0, 1)$ is also a reflecting boundary (thus, $\acute{\beta}_i^t < 1$ is a reflecting boundary). Therefore, $\lim_{t\to\infty} L_i^t/t \leq 0$. □

Figure 3(a) shows the average payoffs of various learners against a seer agent in Matching Pennies. Since a Q-learner (unless exploring) plays the action corresponding to its max Q-value, a seer agent exploits it indefinitely. Also, if M-Qubed uses only strategy rule (3) (labeled M-Qubed(3)), then it may be exploited somewhat. However, if M-Qubed uses strategy rules (4) or (5) (labeled M-Qubed (4) and (5)), it learns to play π_i^* and, therefore, cannot be exploited. Figure 3(b) shows M-Qubed's average payoffs in self play.

5.2. Cooperation and Compromise Property

In this subsection, we present empirical results showing that M-Qubed displays (in self play) the C/C property in many general sum matrix games. In doing so, we compare its performance with the self play of other learning agents (see Table 3) in a number of games.

Prisoner's Dilemma. Figure 3(c) shows average payoffs for the Prisoner's Dilemma. M-Qubed learns to always cooperate, resulting in a payoff of 3 to both agents. The other learners learn strategies that produce inferior payoffs. Note, however, that QL, which stops exploring, is more successful than rQL, which continues exploring. WoLF-PHC generally learns to defect since it plays a stochastic (unpredictable) policy while learning.

Figures 4(a) and (b) show the average payoffs to agent 1 when M-Qubed agents use different values of α and γ. The parameter values of the two agents are shown along the x and y-axes respectively. Lighter colors indicate that agent 1 received high payoffs. The figures show that M-Qubed is more successful when using low values of α and high values of γ.

Chicken. M-Qubed also learns mutual cooperation in Chicken, giving it the highest average payoffs of the learners (see Figure 3(f)).

Shapley's Game. Many learning algorithms do not converge in Shapley's game (Fudenberg & Levine, 1998; Bowling, 2004). However, M-Qubed does converge to a cooperative solution which yields an average payoff of $\frac{1}{2}$ to each agent (see Figure 3(d)). The other learners play strategies that yield only slightly lower average payoffs, but do not typically converge. Again, QL outperforms rQL since QL stops exploring.

Staghunt. While all the other learners typically learn to play a in Staghunt, M-Qubed learns to play b, which results in a significantly higher payoff (see Figure 3(e). Figures 4(c) shows the results of varying α, which again shows that lower values of α are more successful.

Tricky Game. Figures 3(g) and (h) show payoffs to the row and column players respectively. Again, M-Qubed outperforms the other agents and QL outperforms rQL

6. Summary and Discussion

We have advocated that learners in repeated games should a) never learn to receive payoffs below the minimax value of the game and b) have the ability to cooperate/compromise. No algorithm from the literature is known to satisfy both of these properties simultaneously. We presented an algorithm (M-Qubed) that provably satisfies the security property and displays the C/C property in self play in many games. In these games, M-Qubed learns the solutions proposed in (Littman & Stone, 2003), which are pareto efficient. These results support the hypothesis that learning agents should act predictably in non-competitive games, and unpredictably otherwise.

M-Qubed has a number of weaknesses, one of which is that it can be "bullied" by more intelligent agents (though it still gets a payoff as great as its minimax value). Additionally, M-Qubed does not have the C/C property when associating with many other learners. We believe that future learning algorithms should use principles such as reputation, teaching, etc. in order to satisfy the C/C property with a larger class of learners.

Acknowledgments

We thank Michael Bowling and Rich Sutton for helpful discussions about the algorithm presented herein.

References

Bowling, M. (2004). Convergence and no-regret in multiagent learning. *Advances in Neural Information Processing Systems.*

Bowling, M., & Veloso, M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence, 136(2)*, 215–250.

Crandall, J. W., & Goodrich, M. A. (2004). Learning near-pareto efficient solutions with minimal knowledge requirements using satisficing. *AAAI Spring Symp. on Artificial Multiagent Learning.*

Fudenberg, D., & Levine, D. K. (1998). *The theory of learning in games.* The MIT Press.

Gintis, H. (2000). *Game theory evolving: A problem-centered introduction to modeling strategic behavior.* Princeton, New Jersey: Princeton University Press.

Greenwald, A., & Hall, K. (2003). Correlated-q learning. *Proc. of the 20th Intl. Conf. on Machine Learning.*

Hu, J., & Wellman, M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. *Proc. of the 15th Intl. Conf. on Machine Learning.*

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. *Proc. of the 11th Intl. Conf. on Machine Learning.*

Littman, M. L. (2001). Friend-or-foe: Q-learning in general-sum games. *Proc. of the 18th Intl. Conf. on Machine Learning.*

Littman, M. L., & Stone, P. (2003). A polynomial-time nash equilibrium algorithm for repeated games. *ACM Conf. on Electronic Commerce.*

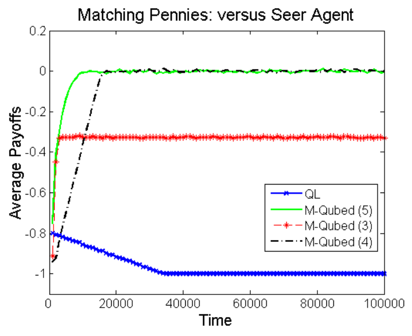
Nash, J. F. (1951). Non-cooperative games. *Annals of Mathematics, 54*, 286–295.

Sandholm, T. W., & Crites, R. H. (1995). Multiagent Reinforcement Learning in the Iterated Prisoner's Dilemma. *Biosystems, Special Issue on the Prisoner's Dilemma.*

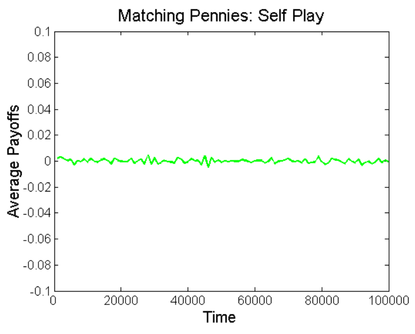
Stimpson, J. R., Goodrich, M. A., & Walters, L. C. (2001). Satisficing and learning cooperation in the prisoner's dilemma. *Proc. of the 17th Intl. Joint Conf. on Artificial Intelligence.*

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction.* The MIT Press.

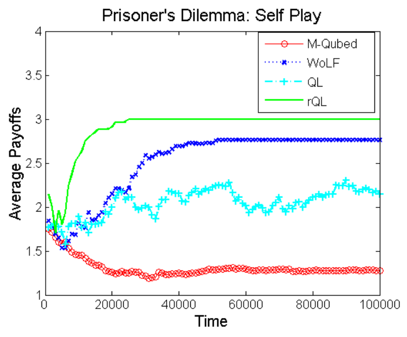
Watkins, C. (1989). *Learning from delayed rewards.* Doctoral dissertation, University of Cambridge.



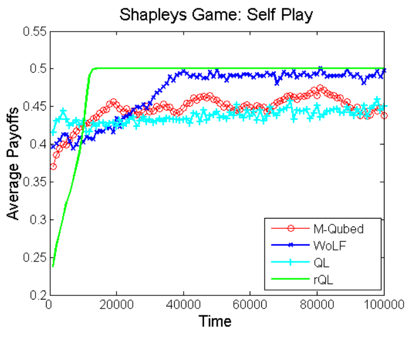
(a)



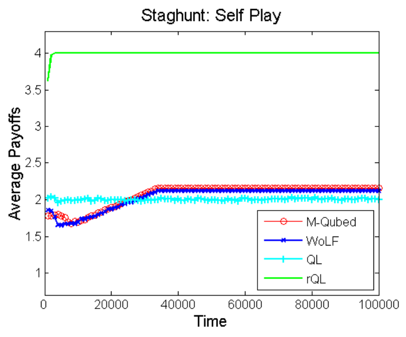
(b)



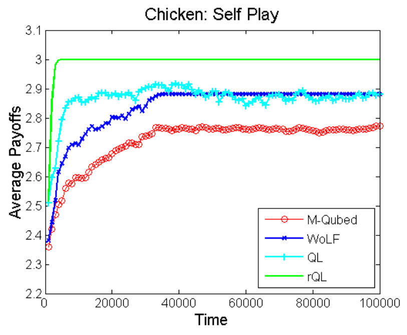
(c)



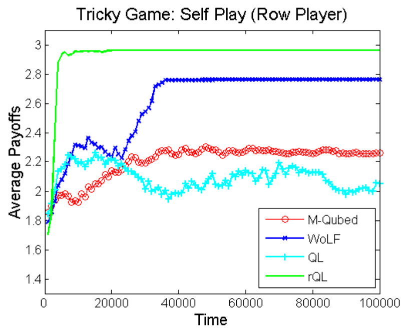
(d)



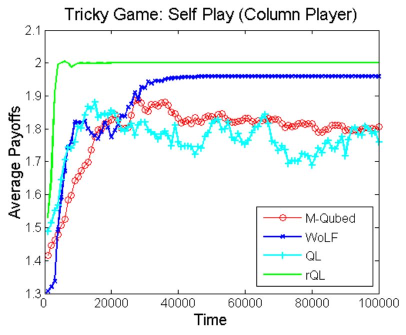
(e)



(f)

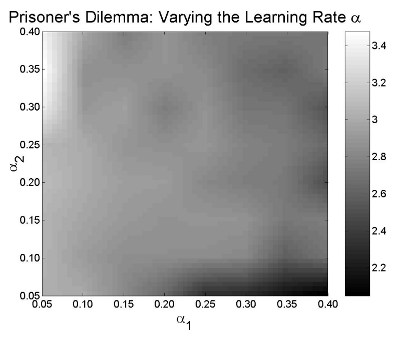


(g)

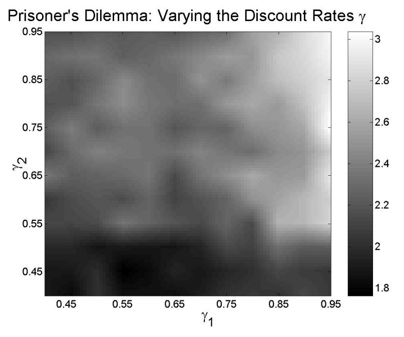


(h)

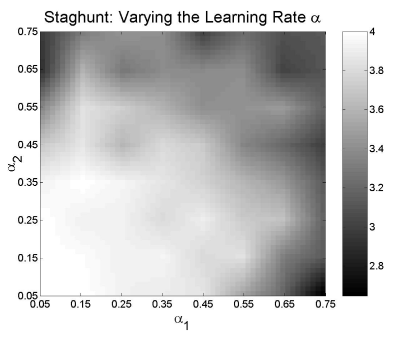
Figure 3. Plots showing the performance of several learning algorithms in many of the matrix games shown in Figure 1. Each plot shows the sliding average (mean of 50 trials) obtained by the agents over time. (a) learners vs. seer agent in matching pennies. For M-Qubed (4), m_i in (4) is replaced by the function $f(t) = \min(l_\$ + \frac{t(m_i - l_\$)}{30000}, m_i)$. (b) M-Qubed in self play in matching pennies. (c), (d), (e), (f), (g), (h) Average payoffs to agents in self play in the prisoner's dilemma, Shapley's game, staghunt, chicken, and tricky game. Unless stated otherwise, parameter values are as given in Table 3.



(a)



(b)



(c)

Figure 4. Average payoffs to player 1 of learned strategies for M-Qubed in self play when parameter values are varied by the agents as shown along the axes.